A Series of books in the Mathematical Sciences *Victor Klee, editor*	*М. Гэри* *Д. Джонсон*
Computers and Intractability **A Guide to the Theory of NP-Completeness**	**ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ И ТРУДНОРЕШАЕМЫЕ ЗАДАЧИ**
Michael R. Garey *David S. Johnson*	Перевод с англ. *Е.В. Левнера* и *М.А. Фрумкина*
Bell Laboratories Murray Hill, New Jersey W.H. Freeman and Company *San Francisco*, 1979	под ред. *А.А. Фридмана* Издательство «Мир» *Москва,* 1982

Монография американских учёных, посвящённая вопросам сложности решения комбинаторных задач, возникающих в дискретной оптимизации, математическом программировании, алгебре, теории чисел, теории автоматов, математической логике, теории множеств, теории графов и т.п. Книга отличается строгим и систематическим изложением теории, в приложении содержится более 300 труднорешаемых задач из различных разделов математики.

Для математиков-прикладников, аспирантов и студентов университетов.

Оглавление

ОТ РЕДАКТОРА ПЕРЕВОДА

Математике всегда было присуще стремление разрабатывать эффективные методы решения как можно более широких классов задач. Многолетний опыт развития теории дискретных и комбинаторных задач и практика их решения показали, что эти две стороны – общность метода и его эффективность – находится в известном антагонизме. Вместе с тем, очень важно знать, и в особенности это касается задач дискретной математики, можно ли в принципе надеяться на создание достаточно общих и эффективных методов или надо сознательно идти по пути разбиения задач на всё более узкие классы и, пользуясь их спецификой разрабатывать для них эффективные (хорошие) алгоритмы.

Неудачи, постигшие исследователей на этом пути, привели к необходимости анализа сложности задач. На наших глазах возникло новое и интенсивно развивающееся «сложностное» направление исследований. Число публикаций в этой области быстро растёт, так что появление настоящей книги, посвящённой вопросам сложности дискретных и комбинаторных задач, вполне закономерно. Проблематика эта весьма актуальна, поскольку такие задачи часто возникают на практике (в экономике, технике, военном деле, в исследовании операций и т.д.), а их решение средствами ЭВМ наталкивается на трудности, связанные с большими затратами машинного времени и памяти.

Предисловие авторов и вводная глава, где обсуждается содержание книги, позволяют читателю ознакомиться (на интуитивном уровне) с основными идеями, понятиями и вопросами, освещаемыми в книге. Поэтому мы, не повторяясь, заострим внимание читателя на некоторых принципиальных вопросах.

Большинство дискретных и комбинаторных задач, вообще говоря, допускает решение с помощью некоторого процесса перебора. Так, например, булевскую задачу о рюкзаке вида

$$\sum_{i=1}^{n} a_i x_i = b, x_i = 0, 1$$

можно решить, перебирая булевские n-мерные векторы (варианты). Такие задачи называются переборными. Однако число шагов переборного метода растёт экспоненциально в зависимости от размеров задачи. Для некоторых задач этого типа удаётся построить эффективные (существенно менее трудоёмкие, чем полный перебор вариантов) методы решения, (это удаётся сделать, например, для решения в Z^n систем линейных уравнений с целыми коэффициентами для потоковых задач, для нахождения паросочетаний в графе и т.д.). Однако, число таких задач невелико.

Анализ трудностей, встретившихся на пути создания эффективных методов решения дискретных задач, привел к постановке центральной теоретико-методологической проблемы всей дискретной математики – можно ли исключить перебор при решении дискретных задач? Другими словами, речь идёт о принципиальной возможности найти нужное решение (например, оптимальное), не перебирая всех или почти всех вариантов в задаче. Эта проблема имеет не только чисто математическое, но и глубокое познавательное значение. Оно заключается в том, что при поиске эффективных точных методов решения широкого класса дискретных задач надо учитывать возможность отсутствия таких методов и, следовательно, уметь вовремя ограничить «аппетит», признав что существуют *«труднорешаемые задачи»*. До настоящего времени эта проблема остаётся открытой.

Феномен труднорешаемых задач не нов для математики. После уточнения понятия алгоритма было обнаружено наличие алгоритмически неразрешимых проблем. В качестве наиболее известного примера укажем десятую проблему Гильберта: «*Существует ли алгоритм, который по данному многочлену* $p(x_1, x_2, ..., x_n)$ *с целыми коэффициентами распознает, имеет ли уравнение* $p = 0$ *решение в целых числах*». Как показал *Ю.В. Матиясевич* [376], такого алгоритма в принципе не существует.

В переборных задачах, как правило, имеется конечное множество вариантов, среди которых нужно найти решение. Так, в задаче о рюкзаке решение отыскивается среди 2^n булевских векторов длины n, и перебрав это экспоненциальное множество векторов, мы обязательно решим задачу. Но с ростом n число векторов быстро растёт и задача становится «труднорешаемой», т.е. практически неразрешимой. Поэтому в конечной области аналогом алгоритмической неразрешимости является перебор экспоненциального числа вариантов, а аналогом алгоритмической разрешимости – существование алгоритма существенно более экономичного, чем перебор. Стало общепринятым считать переборную задачу решаемой эффективно, если имеется алгоритм, решающий её за время, ограниченное полиномом от «размера задачи».

В настоящей книге систематически излагается концепция эффективной (полиномиальной) сводимости переборных задач, развитая в работах *С. Кука* [91], *Р. Карпа* [280], *Л. Левина* [340], а также освещаются более поздние результаты в этой области. Роль основного инструмента здесь играет понятие *сводимости* задач, возникшее в математической логике. Говорят, что переборная задача Π_1 сводится к переборной задаче Π_2, если метод решения задачи Π_2 можно преобразовать в метод решения задачи Π_1. Сводимость называется *полиномиальной*, если подобное преобразование можно осуществить за полиномиальное время. Главными объектами теории являются: класс NP всех переборных задач и класс P переборных задач, разрешимых за полиномиальное время на машине Тьюринга. Очевидно, что $P \subseteq NP$. Центральным в теории сводимости является вопрос о том, *совпадают ли классы* P и NP – такова математическая формулировка проблемы освобождения от необходимости перебора.

В классе NP выявлены так называемые универсальные (или NP-полные) задачи, к которым полиномиально сводится любая задачи из NP. В этом смысле универсальные задачи как бы дают эталон сложности. В настоящее время установлена универсальность многих задач, эквивалентных между собой относительно полиномиальной сводимости. В книге приведено много примеров NP-полных задач. Если бы удалось доказать, что хотя бы какая-то одна NP-полная задача принадлежит классу P, то тем самым было бы доказано, что $P = NP$, и можно было бы в принципе надеяться на построение эффективных алгоритмов для различных классов дискретных задач. Если же классы P и NP различны, то придётся разрабатывать эффективные алгоритмы для всё более узких классов задач. Однако возможна и такая ситуация, что гипотезу $P \neq NP$ (принимаемую многими математиками) нельзя ни доказать, ни опровергнуть (аналогично континуум-гипотезе).

Большой практический опыт решения дискретных задач даёт основание считать, что NP-полные задачи и задачи из P сильно отличаются по трудоёмкости решения, но в строгом смысле до сих пор это различие и, следовательно, различие между классами P и NP не доказано. Это, в частности, объясняется тем, что классы P и NP определяются с помощью понятия времени работы вычислительного устройства с потенциально неограниченной памятью. Однако хорошо известно, что время выполнения алгоритма на машине плохо поддаётся описанию и анализу общематематическими средствами. Следует отметить, что ещё в 1965 г. *Кобхэм* [80] попытался описать сложность вычисления функций внутренним образом, в рамках последовательности алгебраических операций, порождающих эти функции. К сожалению, эта работа оказалась в тени, поскольку в то время не было ещё теории сводимости переборных задач.

Излагаемая в книге и ставшая уже классической теория полиномиальной сводимости была построена для задач распознавания свойств. Оказывается, можно построить аналогичную теорию (полиномиальной) сводимости для экстремальных дискретных задач [16*, 31*, 32*]. Более того, при построении указанной теории можно не пользоваться моделью вычислительного устройства, т.е. сделать её машинно независимой, а всё изложение вести в терминах вычислимых операторов над функциями [16*, 21*, 22*].

Книга Гэри и Джонсона – это, насколько нам известно, первая монография по теории полиномиальной сводимости дискретных задач. В ней систематически изложен материал, разбросанный по журнальным статьям и отчётам. Авторы не ограничились описанием концепции Кука-Карпа-Левина, а дополнили её новыми вопросами, в том числе своими собственными результатами. Особый интерес в этом смысле представляют гл. 6-7, где собран большой теоретический материал по новым направлениям исследований и намечаются «точки роста» теории. О разнообразии и богатстве материала этих глав свидетельствует краткий перечень вопросов, освещаемых в них: оценки алгоритмов, их отклонения от оптимума, оценки в худшем случае и в среднем для различных классов задач; структура класса NP, существование в NP трудноразрешаемых задач, отличных от NP-полных; полиномиальная по сложности иерархия задач; сложность задач определения числа решений без их предъявления; различные типы вычислительных ресурсов, используемых при решении задач; взаимосвязь между сложностью по времени и сложностью по памяти; задачи с логарифмической памятью, сводимость с логарифмической памятью, доказательства трудноразрешаемости задач.

Книга содержит поистине уникальное приложение – в нём собрано более

300 задач из различных областей науки: теории графов, математического программирования, теории расписаний, теории языков и автоматов, математической логики, и отдельно – открытых задач. Здесь для каждой задачи, помимо основной формулировки и источника, приводится комментарий, включающий сводку результатов о задаче и её модификациях (с указанием авторов), а также смежные вопросы. Всё это делает приложение исключительно ценным справочным пособием.

В конце книги дана обширная библиография – около 600 работ. Мы сочли целесообразным несколько расширить библиографический список, добавив ряд работ – либо вышедших позже, либо оказавшихся вне поля зрения авторов. Эти работы дают представление о новых направлениях исследований и результатах в рассматриваемой области. (Добавленные при переводе работы сведены в отдельный список и их номера помечены звёздочкой.)

Книга хорошо написана и читается достаточно легко, хотя новизна материала и, как следствие, не вполне установившаяся терминология затрудняли работу переводчиков и редактора. Неудивительно, что при большой насыщенности и разнообразии материала в книге обнаружились некоторые неточности, погрешности, опечатки, которые переводчики и редактор постарались устранить путём введения соответствующих примечаний или небольшого исправления авторского текста.

Книга предназначена для математиков, а также для всех, кто интересуется дискретными и комбинаторными задачами и их приложениями. Она отражает новое направление в развитии теории дискретных и комбинаторных задач. Внимательный читатель найдёт в ней новый взгляд на дискретные задачи, свежие идеи, подходы, много открытых вопросов, порождающих и стимулирующих интерес к самостоятельной творческой работе.

В заключении хочется поблагодарить *Ю.И. Хмелевского, М.Р. Когаловского, В.П.Гришухина, А.Л. Семёнова, С.Г. Влэдуца, А.Б. Арановича* за помощь в работе над переводом книги: их замечания и советы помогли устранить ряд неточностей и способствовали улучшению качества перевода.

А. Фридман

ПРЕДИСЛОВИЕ

Найдётся немного научных терминов, так быстро завоевавших широкую известность, как понятие «*NP*-полная задача». За короткий отрезок времени с момента возникновения этого понятия в начале 70-х годов оно стало символом громадных трудностей, с которыми всё чаще приходится сталкиваться разработчикам алгоритмов по мере того, как они берутся за решение задач постоянно возрастающей размерности и усложняющейся структуры. Многочисленные и разнообразные задачи, часто встречающиеся в математике, теоретическом программировании и исследовании операций, оказались *NP*-полными, и список таких задач пополнятся почти ежедневно. *NP*-полные задачи настолько широко распространены, что для всех, кто в своей работе соприкасается с вычислительными аспектами указанных областей, очень важно понимать смысл и значение этого понятия.

Настоящая книга задумана как подробное руководство по теории *NP*-полных задач; здесь особое внимание уделено тем понятиям и приёмам, которые представляются наиболее полезными для применения теории к практическим задачам. Книгу можно условно разбить на три части.

В первой части (гл. 1-5) изложены основы теории *NP*-полных задач. Гл. 1 представляет собой сравнительно элементарное введение в некоторые центральные понятия вычислительной сложности, и в этом контексте обсуждается значение *NP*-полных задач. В гл. 2-5 подробно изложены определения и методы доказательства, необходимые для того, чтобы глубже понять теорию и научиться ею пользоваться на практике.

Во второй части (гл. 6 и 7) дан обзор двух альтернативных направлений дальнейшего изучения. В гл. 6 внимание сосредоточено на поиске эффективных «приближённых» алгоритмов решения *NP*-полных задач, т.е. области, тесно связанной с теорией *NP*-полноты. В гл. 7 обсуждаются многочисленные теоретические вопросы сложности вычислений, многие из которых возникли в результате развития теории *NP*-полных задач, изложенной в предыдущих главах. Обе эти главы (особенно гл. 7) следует рассматривать исключительно как введение в соответствующие области. Читателю, желающему более детально ознакомиться с теми или иными вопросами, следует обратиться к указанной нами литературе.

Третья, заключительная часть книги – приложения. В нём содержится большой список *NP*-полных и *NP*-трудных задач (более 300 основных задач и большое число относящихся к ним результатов). В примечаниях к задачам обсуждаются известные результаты, касающиеся сложности подзадач и различных модификаций основной задачи.

Книгой можно пользоваться как вспомогательным материалом к курсам по построению алгоритмов, вычислительной сложности, исследованию операций или комбинаторной математике. С неё можно, например, начинать семинар по приближённым алгоритмам или вычислительной сложности для студентов старших курсов и аспирантов. Один из авторов книги (Д. Джонсон) использовал первоначальный вариант книги в качестве основы семинара аспирантов по приближённым алгоритмам. При этом гл. 1-5 были пройдены за 5 недель, а изучение гл. 6 сопровождалось широким привлечением материала, указанного в библиографическом списке. Семинар по теории сложности можно организовать аналогичным образом, приняв в качестве основы для изучения литературы гл. 7 вместо гл. 6. Эти главы можно также изучать и одновременно, на «комбинированных» семинарских занятиях.

В более общем смысле книга может служить как учебником для читателя, интересующегося теорией *NP*-полных задач, так и справочником для исследователя и практика, сталкивающегося с алгоритмами и их сложностью. К изучению *NP*-полных задач, приведённых в приложении, читатель может приступить сразу, даже не прочитав основного материала книги,– для этого требуется лишь знакомство с основными понятиями теории *NP*-полных задач. Начинающие могут достичь такого знакомства, если бегло прочтут гл. 1-5, концентрируя внимание на неформальных обсуждениях определений и методов и обращаясь к формальным рассуждениям только тогда, когда это необходимо для полной ясности. Чтобы этой книгой можно было пользоваться как справочником, в предметный указатель включено большое число терминов, а в обширной библиографии приведены номера разделов книги, в которых упоминаются соответствующие работы.

Мы признательны многим людям, помогавшим нам в работе над книгой. *Х. Габов, Л. Лэндвебер* и *Р. Тарьян* ознакомились с первоначальным её вариантом и исходя из своего опыта внесли ценные предложения. *А. Ахо, Ш. Ивен, Р. Грэхем, Г. Хант, В. Кли, А. Мейер, К. Пападимитриу, Г. Поллак, С. Сахни, Р. Сефи, Л. Стокмейер* и *Дж. Ульман* прочли рукопись всей книги или её отдельных частей и сделали конструктивные замечания. Мы не смогли бы здесь перечислить всех исследователей, которые откликнулись на нашу просьбу присылать результаты об *NP*-полноте задач и помогли нам составить обширный список *NP*-полных задач. Их имена можно найти в библиографии. Несколько наших коллег по Bell Laboratories, особенно *Б. Керниган*, оказали нам неоценимую помощь при машинном наборе книги с помощью вычислительной системы $UNIX^R$. Наконец, хотелось бы особенно отметить *Ж. Рейнболд*, которая превратила наши рукописные иероглифы в безупречные полиграфические данные, чем существенно способствовала созданию этой книги.

Муррей Хилл, Нью Джерси,
октябрь 1978

Майкл Р. Гэри
Дэвид С. Джонсон

Глава 1.

Вычислительные машины, сложность и труднорешаемые задачи

1.1. Введение

Охарактеризовать предмет этой книги, пожалуй, лучше всего можно на таком несколько экстравагантном примере.

Представьте себе, что вы, как и авторы книги, занимаетесь исследовательской работой в одной из отраслей промышленности. Однажды шеф вызывает вас к себе в кабинет и доверительно сообщает, что компания собирается приступить к производству и продаже <бандерснечей»[1] в условиях жёсткой конкуренции. По этой причине нужен хороший метод, выявляющий возможность (или невозможность) создания новых изделий – узлов бандерснеча с заданными техническими характеристиками, причём этот метод должен выдавать ответ для любого заданного набора технических характеристик. Кроме того, в случае положительного ответа требуется представить проект соответствующего бандерснеча. Так как вы отвечаете за разработку алгоритмов в фирме, то вам и предстоит отыскать требуемый метод и построить соответствующий эффективный алгоритм.

Получив в Отделе бандерснечей необходимые разъяснения по постановке задачи, вы спешите в своей кабинет, запасаетесь справочниками и с большим энтузиазмом погружаетесь в решение задачи. Спустя несколько недель кабинет завален грудами скомканных черновиков, а ваш энтузиазм заметно поубавился. Вам всё ещё не удалось придумать алгоритма существенно лучшего, чем перебор всех возможных вариантов. За подобное решение

[1] Бандерснеч, или Брандашмыг,– фантастических образ в сказке Льюиса Кэррола «Алиса в Зазеркалье», это слово точного смысла не имеет. Как сказала сама Алиса, «такие слова наводят на всякие мысли, хоть и неясно – на какие». – *Прим. перев.*

вы едва ли удостоитесь похвалы шефа, поскольку перебор всех вариантов только для одного набора технических характеристик потребует нескольких лет машинного времени, а в Отделе бандерснечей уже по тринадцати узлам бандерснеча наблюдается отставание от намеченного графика. Естественно, что у вас нет желания возвращаться в кабинет шефа со словами:

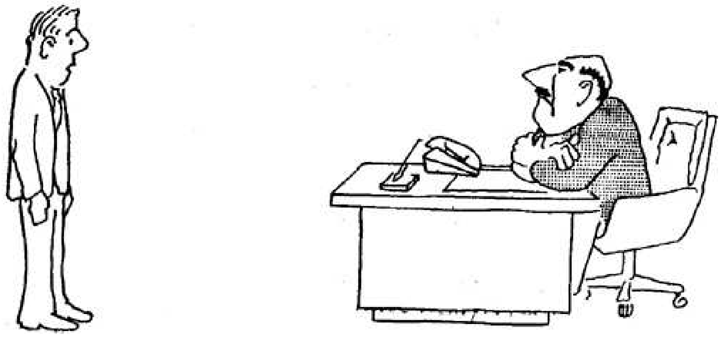


Рис. 1..1: *«Я не могу найти эффективного алгоритма, боюсь, что я для этого слишком туп»*

Для вашего положения в компании было бы гораздо лучше, если бы вы смогли доказать, что поставленная задача в принципе труднорешаема и что ни один алгоритм вообще не в состоянии решить её быстро. В этом случае вы могли бы уверенно шагнуть в кабинет шефа и заявить:

Рис. 1..2: *«Я не могу найти эффективного алгоритма, потому что такого алгоритма не существует!»*

К сожалению, доказать, что задача труднорешаема, может быть, не менее трудно, чем найти эффективные алгоритмы. Даже выдающиеся теоретики оказывались беспомощными в попытках получить подобное доказательство для часто встречающихся трудных задач. Тем не менее эта книга поможет вам найти достойный выход из положения. Теория *NP*-полных задач предлагает много простых методов доказательства того, что та или иная конкретная задача столь же трудна, как и большое число других задач признанных очень трудными и уже много лет не поддающихся усилиям специалистов.

Взяв на вооружение эти методы, вы могли бы доказать *NP*-полноту задачи о бандерснече и таким образом установить её эквивалентность всем другим трудным задачам этого класса. В этом случае можно было бы смело отправиться к шефу и сообщить:

Рис. 1..3: *«Я не могу найти эффективного алгоритма, но этого не может сделать и никто из этих знаменитостей»*

Во всяком случае, ваше начальство будет знать, что нет смысла увольнять вас и брать на работу другого специалиста по алгоритмам.

Разумеется, наше собственное начальство вряд ли одобрило бы написание этой книги, если бы её единственной целью было защитить разработчиков алгоритмов. На самом деле выявление *NP*-полноты – это обычно лишь начало работы над задачей. Проблемы отдела бандерснечей не исчезнут сразу лишь потому, что поставленная задача оказалась *NP*-полной. Однако знание этого факта действительно даёт ценную информацию о том, какие подходы к её решению окажутся наиболее перспективными.

Очевидно, что в этом случае не следует делать ставку на поиски эффективного точного алгоритма. Лучше сосредоточить своё внимание на иных, более скромных подходах. Например, вы могли бы начать разработку эффективных алгоритмов, позволяющих решать различные частные случаи поставленной общей задачи. Можно было бы заняться отысканием алгоритмов, хотя и не гарантирующих быстрого решения, однако работающих быстро в большинстве случаев. Наконец, можно даже ослабить некоторым образом постановку задачи и искать быстрый алгоритм проектирования бандерснечей, у которого заданным требованиям удовлетворяет большая часть характеристик. Короче говоря, основное предназначение теории *NP*-полных задач состоит в том, чтобы помочь разработчикам алгоритмов и направить их усилия на выбор таких подходов к решению задач, которые, вероятнее всего, приведут к практически полезным алгоритмам.

В первой главе настоящего «руководства» по теории *NP*-полных задач мы введём многие основные понятия, рассмотрим их применимость (а также выскажем некоторые предостережения) и охарактеризуем содержание остальной части книги.

1.2. Задачи, алгоритмы и сложность

Для того, чтобы в дальнейшем изучать такие понятия, как «труднорешаемые задачи» и «эквивалентные по сложности задачи», необходимо сначала договориться о значении нескольких основных терминов.

Начнём с понятия задачи. Под *массовой задачей* (или просто *задачей*) мы будем понимать некоторый общий вопрос, на который следует дать ответ. Обычно задача содержит несколько параметров, или свободных переменных, конкретные значения которых не определены. Задача Π определяется следующей информацией:

(1) общим списком её параметров,

(2) формулировкой тех свойств, которым должен удовлетворять ответ или, другими словами, решение задачи.

Индивидуальная задача I получается из массовой задачи Π, если всем параметрам задачи Π присвоить конкретные значения.

В качестве примера рассмотрим классическую задачу о коммивояжере. Параметры этой массовой задачи состоят из конечного набора «городов» $C = \{c_1, c_2, ..., c_m\}$ и расстояний $d(c_i, c_j)$ между каждой парой городов c_i, c_j из C. Решение – это такой упорядоченный набор $< c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(m)} >$ заданных городов, который минимизирует величину

$$\sum d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}).$$

Это выражение даёт длину маршрута, начинающегося в городе $c_{\pi(1)}$, проходящего последовательно через все города и возвращающегося в $c_{\pi(1)}$, непосредственно из последнего города $c_{\pi(m)}$.

Индивидуальная задача о коммивояжере, показанная на рис. 1.1, задаётся следующим образом:

$C = \{c_1, c_2, c_3, c_4\}, d(c_1, c_2) = 10, d(c_1, c_3) = 5,$

$d(c_1, c_4) = 9, d(c_2, c_3) = 6, d(c_2, c_4) = 9, d(c_3, c_4) = 3.$

Последовательность $< c_1, c_2, c_3, c_4 >$ представляет собой решение задачи, поскольку соответствующий маршрут имеет минимальную возможную длину, равную 27.

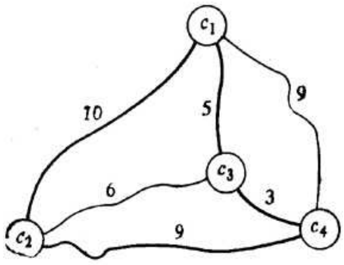


Рис. 1..4: *«Индивидуальная задача о коммивояжере и маршрут минимальной возможной длины, равной 27»*

Под алгоритмом будем понимать общую, выполняемую шаг за шагом процедуру решения задачи. Для определённости мы можем считать её программой для ЭВМ, написанной на формальном машинном языке. Будем говорить, что алгоритм решает массовую задачу П, если он применим к любой индивидуальной задаче I из П и обязательно даёт решение задачи I. Подчеркнём, что термин «решение» понимается здесь строго в соответствии с данным выше определением. Поэтому, в частности, нельзя сказать, что алгоритм «решает» задачу о коммивояжере, если он не выдаст маршрут минимальной длины хотя бы для какой-то одной индивидуальной задачи.

Вообще говоря, нам нужен наиболее «эффективный» алгоритм для решения задачи. В самом широком смысле понятие эффективности связано со всеми вычислительными ресурсами, необходимыми для работы алгоритма. Однако обычно под «самым эффективным» алгоритмом понимается самый быстрый. Поскольку ограничения по времени часто являются доминирующим фактором, определяющим пригодность конкретного алгоритма для практики, основное внимание мы сосредоточим главным образом на этом виде ресурсов.

Время работы алгоритма удобно выражать в виде функции от одной переменной, характеризующей «размер» индивидуальной задачи, т.е. объём входных данных, требуемых для описания этой задачи. Такой подход удобен, поскольку в дальнейшем сравнительная сложность задач будет оцениваться через их размеры. Часто размер задачи измеряется неформально. В задаче о коммивояжере, например, для этой цели обычно используется число городов. Однако в задаче с m городами кроме номеров этих городов на объём входной информации влияют также $m(m-1)/2$ величин, определяющих расстояния между городами, и значения этих величин. Если нам предстоит иметь дело с временны́ми характеристиками в точной математической постановке, то мы должны так определить размер задачи, чтобы все эти факторы были учтены.

Для этого обратим внимание на то, что описание индивидуальной задачи, которое мы даём в терминах входа для ЭВМ, можно рассматривать как одну конечную цепочку (или слово) символов, выбранных из конечного входного алфавита. Невзирая на то что существуют различные пути описания данной индивидуальной задачи, предположим, что заранее выбран некоторый способ и что с каждой массовой задачей связана некоторая фиксированная *схема кодирования*, которая отображает индивидуальные задачи в соответствующие цепочки символов. Входная длина индивидуальной задачи I из Π определяется как число символов в цепочке, полученной применением к задаче I схемы кодирования для массовой задачи Π. Именно это число, т.е. входная длина, и используется в качестве формальной характеристики размера индивидуальной задачи.

Например, различные конкретные задачи о коммивояжере можно описать с помощью алфавита $\{c, [,], /, 0, 1, 2, ..., 9\}$, при этом предыдущий пример будет закодирован в виде такой цепочки символов:

$$c[1]\, c[2]\, c[3]\, c[4]\, //\, 10\, /\, 5\, /\, 9\, //\, 6\, /\, 9\, //\, 3$$

Более сложные индивидуальные задачи кодируются аналогичным образом. При такой кодирующей схеме для задачи о коммивояжере входная длина в нашем примере будет равна 32.

Временна́я сложность алгоритма отражает требующиеся для его работы затраты времени. Это функция, которая каждой входной длине n ставит в соответствие максимальное (по всем индивидуальным задачам длины n) время, затрачиваемое алгоритмом на решение индивидуальных задач этой длины.

Естественно, что эта функция не будет полностью определена до тех пор, пока не зафиксирована схема кодирования, определяющая входную длину индивидуальной задачи, и не выбрано вычислительное устройство (или его модель), определяющее время работы. Однако, как будет видно из дальнейшего, подобные детали окажут незначительное влияние на различия между классами, возникающими в теории NP-полных задач. Поэтому в дальнейшем читателю рекомендуется зафиксировать мысленно какую-либо конкретную схему кодирования для каждой задачи, выбрать некоторое конкретное вычислительное устройство или его модель и рассматривать затем временну́ю сложность алгоритмов в соответствии с получающимися входными длинами и соответствующими затратами времени.

1.3. Полиномиальные алгоритмы и труднорешаемые задачи

Разные алгоритмы имеют различную временну́ю сложность, и выяснение того, какие алгоритмы «достаточно эффективны», а какие «совершенно неэффективные», всегда будет зависеть от конкретной ситуации. Однако теоретики, занимающиеся разработкой и анализом алгоритмов, предлагают для сравнения эффективности алгоритмов один простой подход, позволяющий существенно прояснить ситуацию. Речь идёт о различии между полиномиальными и экспоненциальными алгоритмами.

Будем говорить, что функция $f(n)$ есть $O(g(n))$, если существует константа c, такая, что $|f(n)| \leqslant c\,|g(n)|$ для всех значений $n \geqslant 0$. *Полиномиальным алгоритмом* (или *алгоритмом полиномиальной временно́й сложности*) называется алгоритм, у которого временна́я сложность равна $O(p(n))$, где $p(n)$ – некоторая полиномиальная функция, а n – входная длина. Алгоритмы, временна́я сложность которых не поддаётся подобной оценке, называются «экспоненциальными» (следует отметить, что это определение включает и такие функции, как $n^{\log n}$,– хотя они и не являются полиномиальными, но обычно не считаются экспоненциальными).

Различие между двумя указанными типами алгоритмов становится особенно заметным при решении задач большого размера. Таблица на рис. 1.2 позволяет сравнить скорости роста нескольких типичных полиномиальных и экспоненциальных функций. (Обратите внимание на чрезвычайно быстрый рост двух приведённых экспонент.)

Различие между полиномиальными и экспоненциальными алгоритмами проявляется ещё более убедительно, если проанализировать влияние увеличения быстродействия ЭВМ на время работы алгоритмов. Рис. 1.3 показывает, насколько увеличатся размеры задач, решаемых за один час машинного времени, если благодаря совершенствованию технологии быстродействие ЭВМ возрастёт в 100 или 1000 раз по сравнению с современными машинами. Заметим, что для функции $f(n) = 2^n$ увеличение скорости вычислений в 1000 раз приводит лишь к тому, что размер наибольшей задачи, разрешимой за один час, возрастает только на 10, в то время как для функции $f(n) = n^5$ этот размер возрастёт почти в 4 раза.

Функция временно́й сложности	Размер n					
	10	20	30	40	50	60
n	$1e^{-5}$ с.	$2e^{-5}$ с.	$3e^{-5}$ с.	$4e^{-5}$ с.	$5e^{-5}$ с.	$6e^{-5}$ с.
n^2	$1e^{-4}$ с.	$4e^{-4}$ с.	$9e^{-4}$ с.	$16e^{-4}$ с.	$25e^{-4}$ с.	$36e^{-4}$ с.
n^3	$1e^{-3}$ с.	$8e^{-3}$ с.	$27e^{-3}$ с.	$64e^{-3}$ с.	$125e^{-3}$ с.	$216e^{-3}$ с.
n^5	$1e^{-1}$ с.	3.2 с.	24.3 с.	1.7 мин.	5.2 мин.	13.0 мин.
2^n	$1e^{-3}$ с.	1.0 с.	17.9 мин.	12.7 дн.	35.7 лет	366 ст.
3^n	0.059 с.	58 мин.	6.5 лет	3855 ст.	$2e^8$ ст.	$1.3e^{13}$ ст.

Рис. 1.2. *Сравнение нескольких полиномиальных и экспоненциальных функций временно́й сложности (с. – секунд, дн. – дней, ст. – столетий).*

Размеры наибольшей задачи, разрешимой за один час			
Функция временно́й сложности	На современных ЭВМ (1979 г.)	На ЭВМ, в 100 раз более быстрых	На ЭВМ, в 1000 раз более быстрых
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31,6N_2$
n^3	N_3	$4,64N_3$	$10N_3$
n^5	N_4	$2,5N_4$	$3,98N_4$
2^n	N_5	$N_5+6,64$	$N_5+9,97$
3^n	N_6	$N_6+4,19$	$N_6+6,29$

Рис. 1.3. *Влияние технического совершенствования ЭВМ на полиномиальные и экспоненциальные алгоритмы.*

Эти таблицы демонстрируют некоторые причины, по которым полиномиальные алгоритмы обычно считаются более предпочтительными по сравнению с экспоненциальными. Эта точка зрения, различающая, с одной стороны, полиномиальные алгоритмы, а с другой стороны, экспоненциальные, является отправным пунктом в нашем определении труднорешаемых задач и теории *NP*-полных задач.

Основополагающий характер различия между этими двумя классами впервые обсуждался в работах *Кобхэма* [80] и *Эдмондса* [106]. В частности, Эдмондс отождествлял полиномиальные алгоритмы с «хорошими» алгоритмами и высказал предположение, что некоторые задачи целочисленного программирования невозможно решить такими «хорошими» алгоритмами. Согласно такой точке зрения, экспоненциальные алгоритмы не следует считать «хорошими», и чаще всего так оно и есть.

Большинство экспоненциальных алгоритмов – это просто варианты полного перебора, в то время как полиномиальные алгоритмы обычно можно построить лишь тогда, когда удаётся более глубоко проникнуть в суть решаемой задачи. Имеется широко распространённое соглашение, согласно которому задача не считается «хорошо решаемой» до тех пор, пока для неё не получен полиномиальный алгоритм. Поэтому мы будем называть задачу труднорешаемой, если для её решения не существует полиномиального алгоритма.

Конечно, это формальное определение следует рассматривать только как одну из возможных трактовок понятия «труднорешаемая задача». Различие между «эффективными» (полиномиальными) алгоритмами и «неэффективными» (экспоненциальными) алгоритмами может принять совсем иной характер, когда размеры решаемых задач невелики. Даже на рис. 1.2 функция $f(n) = 2^n$ ведёт себя лучше, чем $f(n) = n^5$ при $n \leqslant 20$. Можно легко построить ещё более яркие примеры.

Более того, известны некоторые экспоненциальные алгоритмы, весьма хорошо зарекомендовавшие себя на практике. Дело в том, что временна́я сложность определена нами как мера поведения алгоритма в *наихудшем случае*, и тот факт, что какой-то алгоритм имеет временну́ю сложность порядка 2^n, означает только, что решение по крайней мере одной индивидуальной задачи размера n требует времени порядка 2^n. На самом деле может оказаться, что большинство индивидуальных задач требует для своего решения значительно меньших затрат времени, и такого рода ситуация действительно имеет место для нескольких хорошо известных алгоритмов. Например, как было установлено в работах *Кли* и *Минти* [291], а также *Заде* [544], симплекс-метод для решения задач линейного программирования имеет экспоненциальную временну́ю сложность[2], но в то же время многочисленные свидетельства подтверждают, что этот метод хорошо работает на практике. Другой пример: алгоритмы ветвей и границ столь успешно решают задачу о рюкзаке, что многие исследователи считают эту задачу «хорошо решаемой», хотя алгоритмы ветвей и границ имеют экспоненциальную временну́ю сложность.

К сожалению, подобные примеры очень редки. Хотя экспоненциальные алгоритмы известны для многих задач, немногие из них считаются приемлемыми для практических целей. Даже при наличии успешно работающих экспоненциальных алгоритмов, упомянутых выше, исследователи не отказались от попыток найти для соответствующих задач полиномиальные алгоритмы. В действительности сам факт успешного применения экспоненциальных алгоритмов давал основание предположить, что они каким-то образом выявляют некоторые существенные особенности решаемых задач и что более глубокое их исследование может привести к дальнейшему улучшению методов. В настоящее время пока не получено удовлетворительных объяснений, почему эти алгоритмы работают успешно, и не известны методы, позволяющие заранее прогнозировать хорошую работу того или иного экспоненциального алгоритма в практической ситуации.

С другой стороны, полиномиальные алгоритмы часто позволяют делать такого рода прогнозы, поскольку полиномиальные функции значительно более адекватно оценивают время работы алгоритмов. Хотя алгоритмы, имеющие временну́ю сложность типа n^{100} или $10^{99}n^2$, не могут считаться эффективными с практической точки зрения, естественно возникающие «полиномиальные» задачи обычно требуют для своего решения (в самом худшем случае) времени порядка n^2 или n^3, причём коэффициенты при старших членах полиномов не слишком велики. Алгоритмы, обладающие такими оценками, *можно* считать «эффективными», и именно это весьма желательное свойство заставляет отдавать предпочтение полиномиальным алгоритмам как средству решения задач.

Наше определение «труднорешаемой задачи» создаёт базу для теории, обладающей значительной общностью и большими возможностями. Понятие труднорешаемой задачи оказывается, по существу, независимым от конкретной схемы кодирования и модели ЭВМ, используемых при определении временно́й сложности.

Рассмотрим сначала схемы кодирования. Предположим, например, что

[2]Аналогичный результат установлен в работе Диница и Карзанова [17*].– *Прим. перев.*

решаемая задача определена на графе $G = (V, E)$, где V – множество вершин, E – множество рёбер и каждое ребро понимается как неупорядоченная пара вершин. Условия этой задачи могут быть описаны (см. рис. 1.4) либо

Схема кодирования	Цепочка символов (слово)	Длина
Списки вершин и рёбер	$V[1]V[2]V[3]V[4](V[1]V[2])(V[2]V[3])$	36
Список соседей	$(V[2])(V[1]V[3])(V[2])()$	24
Строки матрицы инциденций	0100/ 1010/ 0010/ 0000	19

Рис. 1.4. *Описание графа* $G = (V, E)$, *где* $V = \{v_1, v_2, v_3, v_4\}$, $E = \{\{v_1,v_2\}, \{v_2, v_3\}\}$, *при трёх разных схемах кодирования.*

просто списками всех вершин и рёбер, либо с помощью матрицы инциденций графа, либо составлением для каждого узла списка всех узлов, имеющих с данным общее ребро («списки соседей»). Эти схемы кодирования для одного и того же графа могут дать входы разной длины. Однако легко проверить (см. рис. 1.5), что получаемые входы отличаются друг от друга не более чем «полиномиальным образом», т.е. любой алгоритм, имеющий полиномиальную временну́ю сложность при одной из этих схем кодирования, будет также обладать полиномиальной временно́й сложностью при всех остальных схемах.

Схема кодирования	Нижняя оценка	Верхняя оценка
Списки вершин и рёбер	$4\nu + 10e$	$4\nu + 10e + (\nu + 2e) \cdot \lceil \log_{10}\nu \rceil$
Список соседей	$2\nu + 8e$	$2\nu + 8e + 2e \cdot \lceil \log_{10}\nu \rceil$
Матрица инциденций	$\nu^2 + \nu - 1$	$\nu^2 + \nu - 1$

Рис. 1.5. *Общие оценки длины входа для трёх схем кодирования графа* $G = (V, E)$, *представленных на рис. 1.4, где* $|V| = v$, $|E| = e$. *Поскольку* $e < v^2$, *эти оценки показывают, что длины входа отличаются друг от друга не более чем полиномиальным образом* ($\lceil x \rceil$ *обозначает наименьшее целое число, не меньшее, чем* x).

В действительности стандартные схемы кодирования, используемые на практике для любой конкретной задачи, по-видимому, всегда будут отличаться друг от друга не более чем полиномиальным образом. Было бы трудно представить себе разумную схему кодирования для какой-либо задачи, которая отличалась бы более чем полиномиальным образом от стандартных схем. Хотя мы не можем формально выразить такое понятие, как «разумная схема кодирования», следующие два условия охватывают важные требования, связанные с этим понятием.

(1) код любой индивидуальной задачи I должен быть «сжатым», т.е. не содержать избыточной информации или символов;

(2) числа, входящие в условия задачи I, должны быть представлены в двоичной системе счисления (или десятичной, или восьмеричной, или иметь любое другое основание, но только не 1).

Если мы ограничимся рассмотрением только тех схем кодирования, которые удовлетворяют этим условиям, то выяснение вопроса, является ли данная задача труднорешаемой, не будет зависеть от выбора конкретной схемы кодирования.

Моделируемая машина B	Моделирующая машина A		
	1МТ	kМТ	МПД
Машина Тьюринга с 1-й лентой (1МТ)	-	$O(T(n))$	$O(T(n))\log T(n)$
Машина Тьюринга с k лентами (kМТ)	$O(T^2(n))$	-	$O(T(n))\log T(n)$
Машина произвольного доступа (МПД)	$O(T^3(n))$	$O(T^2(n))$	-

Рис. 1.6. *Время, требуемое машиной A для моделирования работы алгоритма сложности $T(n)$ на машине B (например, см. [9, 215]).*

Аналогичные замечания можно сделать относительно выбора модели ЭВМ. Все известные в настоящее время реалистические модели ЭВМ (например, одноленточные и многоленточные машины Тьюринга, машины с произвольным доступом к памяти) эквивалентны относительно полиномиальной временно́й сложности (см. рис. 1.6). Можно ожидать, что и любая иная «разумная» модель ЭВМ будет эквивалентна по сложности всем перечисленным моделям.

Под словами «разумная модель» здесь главным образом имеется в виду то, что объём работы, выполняемой машиной в единицу времени, ограничен (сверху) полиномом. Так, например, модель, обладающая способностью выполнять параллельно произвольно много операций, не будет считаться «разумной», и в действительно ни одна из существующих (или проектируемых) ЭВМ не обладает подобным свойством. Во всяком случае, если ограничиться рассмотрением стандартных моделей реальных ЭВМ, то класс труднорешаемых задач не будет зависеть от выбора конкретной модели, и такой выбор можно осуществлять, исходя из интересов дела и не уменьшая при этом сферы применимости наших результатов.

1.4. Задачи, труднорешаемость которых доказуема

После того как мы обсудили формальное содержание понятия «труднорешаемая задача», уместно дать краткий обзор современного состояния наших знаний о существовании труднорешаемых задач.

Сначала охарактеризуем различия между двумя аспектами труднорешаемости, которые отражены в нашем определении. Первый аспект, который обычно имеется в виду, состоит в том, что для отыскания решения требуется экспоненциальное время. Второй заключается в том, что искомое решение настолько велико, что не может быть представлено в виде выражения, длина которого была бы ограничена полиномом от длины входа.

Вторая ситуация возникает, например, если в задаче о коммивояжере в качестве дополнительного параметра фигурирует число B и требуется найти все маршруты длины, не превосходящей B. Легко построить индивидуальную задачу о коммивояжере, в которой имеется экспоненциальное число маршрутов длины, не превосходящей B, поэтому не существует алгоритма с полиномиальной временно́й сложностью, который все их перечисляет.

Труднорешаемостью этого вида нельзя ни в коем случае пренебрегать, и очень важно её своевременно обнаружить. В большинстве случаев, однако, её наличие ясно из постановки задачи. Этот аспект труднорешаемости обычно можно рассматривать как указание на то, что постановка задачи не реалистична, поскольку мы запрашиваем больше информации, чем когда-либо сможем использовать. В связи с этим с настоящего момента и до конца книги мы будем рассматривать только первый тип труднорешаемости. А именно будут изучаться только такие задачи, длина решения которых ограничена полиномиальной функцией от длины входной информации.

Первые результаты о труднорешаемости задач – классические результаты о неразрешимости – были получены *А. Тьюрингом*. Около 40 лет назад Тьюринг доказал, что некоторые задачи даже «неразрешимы» в том смысле, что вообще не существует алгоритма их решения. Он доказал, например, что невозможно указать алгоритм, который по произвольной программе определял бы, остановится ли эта программа на произвольно заданном входе или нет [517]. Известно большое число других неразрешимых задач. К ним относятся, в частности, задача выяснения тривиальности конечно-порождённых групп [440], десятая проблема Гильберта (разрешимость в целых числах полиномиальных уравнений) [376], а также несколько задач о покрытии плоскости равными областями [36]. Поскольку эти задачи не могут быть решены *никаким* алгоритмом, тот тем более они не разрешимы и полиномиальным алгоритмом и, значит, действительно труднорешаемы в самом сильном смысле.

Первые примеры труднорешаемых «разрешимых» задач были получены в начале 60-х годов, как один из результатов работы *Хартманиса* и *Стириза* [204] по «иерархиям» сложности. Однако эти примеры включали только «искусственные» задачи, специально построенные, чтобы обладать соответствующими свойствами. Только в начале 70-х годов *Мейеру* и *Стокмейеру* удалось показать, что некоторые «естественные» разрешимые задачи труднорешаемы.

В число этих задач вошло большое число изучавшихся ранее задач из теории автоматов, формальной теории языков и математической логики. На самом деле было показано, что эти задачи не могут быть решены за полиномиальное время даже с помощью «недетерминированного» вычислительного устройства, обладающего способностью параллельно выполнять неограниченное количество независимых вычислений. В дальнейшем будет видно, что эта «нереалистичная» модель вычислительного устройства играет важную роль в теории NP-полных задач, а её возможности будут очерчены более полно в гл. 2.

Все известные в настоящее время задачи, труднорешаемость которых доказана, попадают в один из перечисленных классов: они либо вовсе неразрешимы, либо труднорешаемы недетерминированной машиной. Однако большинство представляющихся труднорешаемыми практических задач в действительности *разрешимы* и, более того, *могут* быть решены за полиномиальное время с помощью недетерминированного вычислительного устройства. Таким образом, известные методы недостаточно сильны, чтобы установить труднорешаемость этих задач.

1.5. *NP*-полные задачи

Пока теоретики продолжают поиск более мощных методов доказательства труднорешаемости задач, параллельно ведутся работы по сравнению сложности различных задач. Как уже было сказано ранее, обнаружение таких взаимосвязей между задачами часто может дать разработчику алгоритмов полезную информацию.

Основной метод, используемый для доказательства того, что две задачи близки, состоит в «сведении» их друг к другу с помощью конструктивного преобразования, которое отображает любую индивидуальную задачу первого типа в эквивалентную ей индивидуальную задачу второго типа. Такое преобразование позволяет превратить любой алгоритм решения второй задачи в соответствующий алгоритм решения первой задачи.

Много простых примеров подобной сводимости известно уже давно. Например, *Данциг* [99] свёл несколько комбинаторных задач оптимизации к общей задаче 0-1 целочисленного программирования. *Эдмондс* [105] свёл задачи из теории графов – «найти минимальное вершинное покрытие графа» и «найти минимальное независимое множество вершин графа» – к задаче «о покрытии». *Джимпел* [173] свёл задачу о покрытии к задаче о «покрытии простыми импликантами» из теории релейно-контактных схем. *Данциг, Блаттнер* и *Рао* [100] дали описание хорошо известной сводимости задачи о коммивояжере к задаче о кратчайшем пути, в которой допускаются рёбра отрицательной длины.

Эти первые сводимости, хотя и были изолированы и относились к узкому кругу задач, предвосхитили дух результатов, которые доказываются в теории *NP*-полных задач.

Фундамент теории *NP*-полных задач был заложен в работе *С. Кука*, опубликованной в 1971 г. под названием «Сложность процедур вывода теорем» [91]. В этой короткой, но элегантной работе Кук получил несколько важных результатов.

Во-первых, он подчеркнул важность понятия «сводимость за полиномиальное время», т.е. сводимость, которая выполняется с помощью алгоритма с полиномиальной временно́й сложностью. Если одна задача сводится за полиномиальное время к другой, то любой полиномиальный алгоритм решения второй задачи может быть превращён в полиномиальный алгоритм решения первой.

Во-вторых, он обратил внимание на класс задач распознавания свойств (класс *NP*), которые могут быть решены за полиномиальное время на недетерминированном вычислительном устройстве. (Задачей распознавания свойств называется задача, решениями которой могут быть либо «да», либо «нет».) Большинство не поддающихся решению задач, которые встречаются на практике, после переформулировки их в виде задач распознавания попадают в этот класс.

В-третьих, С. Кук доказал, что одна конкретная задача из *NP*, называемая задачей о выполнимости, обладает тем свойством, что всякая другая задача из класса *NP* может быть сведена к ней за полиномиальное время[3]. Таким образом, если задача о выполнимости может быть решена за полиномиальное время, то и любая задача из класса *NP* полиномиально разрешима, а если какая-то задача из *NP* труднорешаема, то и задача о выполнимости также должна быть труднорешаемой. Таким образом, в некотором смысле задача о выполнимости – «самая трудная» в классе *NP*.

Наконец, С. Кук предположил, что и другие задачи из класса *NP* могут быть, аналогично задаче о выполнимости, самыми «трудными» представителями класса *NP*. Он показал это для следующей задачи: «Содержит ли заданный граф G полный подграф с заданным числом вершин?»

Вслед за этим *Р. Карп* опубликовал ряд результатов [280], из которых следует, что многие хорошо известные комбинаторные задачи, включая задачу о коммивояжере, будучи сформулированы в виде задачи распознавания, столь же «трудны», как задача о выполнимости. Позднее относительно широкого круга других задач было доказано, что они по трудности эквивалентны этим задачам, а сам класс эквивалентности, состоящий из «самых трудных» задач из *NP*, получил название «класс *NP*-полных задач»[4]).

Оригинальные идеи Кука оказались удивительно плодотворными. Они позволили свести много разнообразных вопросов о сложности в единый вопрос: «Верно ли, что *NP*-полные задачи труднорешаемы?» Список, включённый в приложение к настоящей книге, содержит сотни различных задач, *NP*-полнота которых установлена. По мере того как всё больше и больше задач, представляющих интерес с разных точек зрения, попадают в этот класс эквивалентности, его важность постоянно возрастает.

Вопрос о том, действительно ли *NP*-полные задачи труднорешаемы, в настоящее время считается одним из основных открытых вопросов современной математики и теоретической кибернетики. Вопреки готовности большинства специалистов считать, что все *NP*-полные задачи труднорешаемы, прогресс как в доказательстве, так и в опровержении этого далеко идущего предположения весьма незначителен. Однако, несмотря на отсутствие доказательства того, что из *NP*-полноты следует труднорешаемость, *NP*-полнота задачи означает, что для её решения полиномиальным алгоритмом требуется по крайней мере крупное открытие.

---

[3] Такие задачи называют также «универсальными».– *Прим. ред.*

[4] Часто его называют «класс универсальных задач».– *Прим. ред.*

1.6. О содержании книги

Хотя эта книга задумана прежде всего как руководство, помогающее выяснить, является ли *NP*-полной та или иная конкретная задача (либо посредством отыскания её в приложении, либо путём самостоятельного доказательства), в ней также обсуждаются различные возможности решения *NP*-полных задач. Ниже приводится краткий обзор содержания книги.

В гл. 2 изложены формальные основы теории *NP*-полных задач и доказывается теорема Кука. Здесь определяются центральные понятия теории *NP*-полных задач – «язык» и «машина Тьюринга» и указывается их связь с понятиями «задача» и «модель вычислительного устройства», которые обсуждались выше. Эта глава должна познакомить читателя с техникой рассуждений, применяемых в теории *NP*-полных задач.

Гл. 3 посвящена методам доказательства *NP*-полноты различных задач. Большое число примеров иллюстрирует структуру этих методов и поясняет, какой из них следует выбрать в каждом конкретном случае. По существу, доказательство *NP*-полноты задачи состоит в полиномиальном сведении к ней известной *NP*-полной задачи. В этой главе также даётся обзор наиболее подходящих для этой цели *NP*-полных задач и демонстрируется их применение.

В гл. 4 изучаются подходы, с помощью которых теорией *NP*-полных можно пользоваться для более тщательного анализа сложности задачи, для поиска «границы» между полиномиально разрешимыми случаями задачи и *NP*-полными.

В гл. 5 показано, как методы, применяемые для доказательства *NP*-полноты, можно перенести на задачи другого типа, с целью доказать, что они «столь же трудны», как *NP*-полные задачи. Чтобы помочь читателю разобраться в литературе по теории *NP*-полных задач, мы также приводим краткий исторический обзор развития основных идей и изменений в терминологии.

В гл. 6 обсуждаются различные подходы к работе с труднорешаемыми задачами, особенно метод отыскания почти оптимальных решений с помощью быстрых алгоритмов. Приводятся примеры, демонстрирующие достоинства и недостатки этих методов, а также иллюстрируется применение теории *NP*-полных задач в этой области.

Гл. 7 имеет целью познакомить читателя с некоторыми теоретическими результатами и идеями, возникшими параллельно с теорией *NP*-полных задач. Обсуждаются следующие вопросы: полиномиальная иерархия, теория NP-полных задач, полнота в классе задач, распознаваемых с полиномиальной памятью, и «релятивизация» проблемы труднорешаемости *NP*-полных задач.

Последнюю треть книги составляет приложение, в которое включено большое число *NP*-полных или более сложных задач, с соответствующими примечаниями. В списке задач выделены разделы, где собраны задачи из определённых областей, например теории графов, теории расписаний, алгебры и теории чисел, теории покрытий и разбиений, математического программирования, теории языков и автоматов, и самые разные задачи, не попавшие в эту классификацию. В нём указаны также ссылки на близкие задачи, вопрос о сложности которых остаётся открытым, т.е. относительно которых не известно ни полиномиального алгоритма решения, ни доказательства NP-полноты.

Глава 2.

Теория *NP*-полных задач

В настоящей главе изложен формальный аппарат теории *NP*-полных задач. Для того чтобы эта теория приобрела строгие математические черты, необходимо ввести формальные эквиваленты многих интуитивных понятий, например таких, как «задача» и «алгоритм». Одна из основных целей этой главы состоит в выявлении связи между формальными терминами и интуитивными понятиями, которыми чаще всего пользуются. Овладев этой связью, мы сможем в последующих главах вести изложение главным образом на неформальном уровне, обращаясь к формальному только тогда, когда это необходимо, в строгих формулировках и доказательствах.

Глава начинается с обсуждения задач распознавания свойств и представления этих задач в виде «языков», что даёт возможность отождествить «решение» такой задачи с «распознаванием» соответствующего языка. В качестве основной модели процесса вычисления принята одноленточная машина Тьюринга, которая используются также для определения класса P (*Polynomial* – полиномиальный) всех языков, детерминированно распознаваемых за полиномиальное время. Машина Тьюринга затем дополняется гипотетической способностью «угадывания», и эта дополненная модель используется для определения класса *NP* всех языков, «недетерминированно» распознаваемых за полиномиальное время (*Nondeterministically Polynomial*). После обсуждения взаимосвязи классов P и *NP* определяется понятие полиномиального преобразования одного языка в другой. Это понятие применяется для определения наиболее важного класса – класса *NP*-полных задач. Глава заканчивается формулировкой и доказательством фундаментальной теоремы Кука, которая даёт нам первую *NP*-полную задачу.

2.1. Задачи распознавания, языки и кодирование

Из соображений удобства теория *NP*-полных задач строится только для *задач распознавания свойств*. Такие задачи, как было упомянуто в гл. 1, имеют только два возможных решения – «да» или «нет». Выражаясь абстрактно, задача распознавания Π состоит просто из двух множеств: множества D_Π всех возможных *индивидуальных задач* и множества $Y_\Pi (Y_\Pi \subset D_\Pi)$ *индивидуальных задач с ответом «да»*. Однако содержательные задачи распознавания в большинстве случаев обладают различными дополнительными структурными свойствами, которые будут учитываться нами при описании задач.

Стандартная форма, которую мы будем применять для описания задач, состоит из двух частей. В первой части даётся описание условия задачи в терминах различных компонент – множеств, графов, функций, чисел и т.д. Во второй части в терминах условия формулируется вопрос, предполагающий один из двух ответов – «да» или «нет». Это описание определяет множества D_Π и Y_Π очевидным образом. Индивидуальная задача принадлежит D_Π в том и только том случае, когда она может быть получена из стандартной формы описания подстановкой конкретных значений во все компоненты условия. Индивидуальная задача принадлежит Y_Π в том и только в том случае, когда ответом на вопрос задачи будет «да».

В качестве примера ниже приведена хорошо известная задача распознавания из теории графов.

ИЗОМОРФИЗМ ПОДГРАФУ.

УСЛОВИЕ. Заданы два графа: $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$.

ВОПРОС. Верно ли, что G_1 содержит подграф, изоморфный G_2? Другими словами, существуют ли: (1) подмножества $V' \subseteq V_1$ и $E' \subset E_1$, такие, что $|V'| = |V_2|$, $|E'| = |E_2|$, (2) взаимно однозначная функция $f : V_2 \to V'$, такая, что $\{u, v\} \in E_2$ тогда и только тогда, когда $\{f(u), f(v)\} \in E'$?

Задача распознавания, соответствующая задаче о коммивояжере, может быть сформулирована следующим образом.

КОММИВОЯЖЕР

УСЛОВИЕ. Заданы конечное множество $C = \{c_1, c_2, ..., c_m\}$ «городов», «расстояние» $d(c_i, c_j) \in Z^+$ для каждой пары городов $c_i, c_j \in C$ и граница $B \in Z^+$ (здесь Z^+ обозначает положительные целые числа).

ВОПРОС. Существует ли «маршрут», проходящий через все города из C_1, длина которого не превосходит B? Другими словами, существует ли последовательность $< c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(m)} >$ элементов C, такая, что

$$\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}) \leqslant B?$$

В дальнейшем встретится много других примеров употребления такой формы записи задач, но и этих двух достаточно для уяснения её смысла. Второй пример служит также для иллюстрации важного приёма – получения из оптимизационной задачи соответствующей задачи распознавания. Если в оптимизационной задаче среди всех структур данного типа ищется структура, имеющая минимальную «стоимость» (например, среди всех маршрутов ищется маршрут минимальной длины), то этой задаче можно сопоставить задачу распознавания, в которой в качестве дополнительного параметра фигурирует числовая граница B, а вопрос ставится о существовании структуры данного типа, стоимость которой *не превосходит* B (например, маршрут длины не более B). Аналогичным образом задача распознавания может быть получена из задачи максимизации. Для этого достаточно условие «не превосходит» заменить условием «не меньше».

Относительно подобного соответствия между задачами распознавания и задачами оптимизации важно отметить, что поскольку значение функции легко оценить, то задача распознавания не может быть сложнее соответствующей задачи оптимизации. Если для задачи о коммивояжере можно за полиномиальное время найти маршрут минимальной длины, то совершенно ясно, как за полиномиальное время решить соответствующую задачу распознавания. Для этого нужно только найти маршрут минимальной длины, вычислить его длину и сравнить её с заданной границей B. Поэтому, если удастся показать, что КОММИВОЯЖЕР есть *NP*-полная задача (а это в самом деле так), то отсюда будет следовать, что и оптимизационная задача о коммивояжере будет по крайней мере столь же сложна. Таким образом, хотя в теории *NP*-полных задач рассматриваются только задачи распознавания, выводы этой теории вполне применимы к задачам оптимизации. (В гл. 5 будет установлена более тесная связь между задачами распознавания и задачами оптимизации. А именно, будет показано, что многие задачи распознавания, и в частности КОММИВОЯЖЕР, «не проще» соответствующих задач оптимизации.)

Как уже говорилось, теория *NP*-полных задач ограничивается только задачами распознавания. Это объясняется тем, что задачи распознавания имеют очень естественный формальный эквивалент, удобный для изучения методами теории вычислений. Этот эквивалент называется «языком» и определяется следующим образом.

Для любого конечного множества символов Σ (называемого алфавитом) обозначим через Σ^* множество всех конечных цепочек, составленных из символов алфавита Σ, такие цепочки мы будем называть словами. Например, если $\Sigma = \{0, 1\}$, то Σ^* состоит из пустого слова «ε», а также слов 0, 1, 00, 01, 10, 11, 000, 001 и всех других конечных слов, состоящих из нулей и единиц. Любое подмножество $L \subseteq \Sigma^*$ будем называть ***языком*** в алфавите Σ. Таким образом, множество $\{01, 001, 111, 1101010\}$ является языком в алфавите $\{0, 1\}$. Языками также являются множество двоичных записей квадратов целых чисел и само множество $\{0, 1\}^*$.

Соответствие между задачами распознавания и языками устанавливается с помощью схем кодирования, которые обычно применяются для представления индивидуальной задачи при её решении на ЭВМ. Напомним, что схема кодирования e задачи Π описывает каждую индивидуальную задачу из Π подходящим словом в некотором фиксированном алфавите Σ. Таким образом, задача Π и схема кодирования e задачи Π разбивают слова из Σ^* на три класса: слова, не являющиеся кодами индивидуальных задач из Π; слова, являющиеся кодами индивидуальных задач из Π с отрицательным ответом на вопрос, и слова, являющиеся кодами индивидуальных задач из Π с положительным ответом на вопрос. Третий класс слов как раз и есть тот язык, который ставится в соответствие задаче Π при кодировании e и обозначается через $L[\Pi, e]$:

$$L[\Pi, e] = \left\{ \begin{array}{c} \Sigma \text{ есть алфавит схемы кодирования } e, \\ x \in \Sigma^* : \text{а } \ x - \text{код индивидуальной задачи} \\ I \in Y_\Pi \text{ при схеме кодирования } e \end{array} \right\}.$$

При применении формальной теории *NP*-полноты к задачам распознавания будем говорить, что некоторый результат имеет место для задачи Π при схеме кодирования e, если этот результат имеет место для языка $L[\Pi, e]$.

В действительности, следуя общепринятой практике, мы не всегда будем соблюдать все эти формальности. Если ограничиться только «разумными схемами кодирования», то при введении любого нового понятия, или свойства, которое формулируется в терминах языка, оно фактически не будет зависеть от способа кодирования. Другими словами, если e и e' – любые две разумные схемы кодирования задачи Π, то рассматриваемое свойство языков $L[\Pi, e]$ и $L[\Pi, e']$ выполняется (или не выполняется) для них одновременно. Это позволяет, не указывая явно схемы кодирования, говорить неформально о том, что рассматриваемое свойство для задачи Π выполняется или не выполняется. Однако при использовании этого соглашения мы будем подразумевать, что в случае необходимости можно было бы указать такую схему кодирования e, что рассматриваемое свойство выполняется для языка $L[\Pi, e]$.

Заметим, что если вести изложение в стиле, не зависящем от кодирования, то теряется связь с формальным понятием «длина входа». Поэтому необходим некоторый параметр, через который можно выразить временну́ю сложность. Удобно считать, что с каждой задачей распознавания связана не зависящая от схем кодирования функция $Length : D_\Pi \to Z^+$, которая «полиномиально эквивалентна» длине кода индивидуальной задачи, получаемой при любой разумной схеме кодирования. *Полиномиальная эквивалентность* понимается в следующем смысле: для любой разумной схемы кодирования e задачи Π существуют два полинома p и p', такие, что если $I \in D$ и слово x есть код индивидуальной задачи I при кодировании e, то $Length[I] \leqslant p(|x|)$ и $|x| \leqslant p'(Lengrh[I])$, где $|x|$ – длина слова x. Например, в задаче ИЗОМОРФИЗМ ПОДГРАФУ можно положить

$$Length[I] = |V_1| + |V_2|;$$

$G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ – графы, образующие рассматриваемую индивидуальную задачу. В задаче КОММИВОЯЖЕР можно положить

$$Length[I] = m + \lceil \log_2 B \rceil + \max\{\lceil \log_2 d(c_i, c_j) \rceil : c_i, c_j \in C\}.$$

Так как любые две разумные схемы кодирования задачи Π дают полиномиально эквивалентные длины входов, то диапазон для выбора функции

Length очень широк, а получаемые результаты будут верны для любой функции *Length*, удовлетворяющей сформулированным выше условиям.

Плодотворность такого неформального, не зависящего от кодирования подхода связана, конечно, с соглашением о том, что именно понимается под «разумной схемой кодирования». Обычно схему кодирования считают «разумной», если она достаточно «сжатая» и «допускает декодирование». Содержание понятия «сжатое» связано с условиями, сформулированными в гл. 1. «Сжатость» нужна для того, чтобы при кодировании индивидуальной задачи сохранялась её естественная краткость, присущая записи задачи, когда её решают на ЭВМ, и не допускалось искусственное «раздувание» входа. Раздувание может, например, привести к такому увеличению длины входа, что экспоненциальный алгоритм превратиться неестественным образом в полиномиальный. Смысл понятия «декодируемость» заключается в том, чтобы по данной компоненте условия задачи можно было бы указать алгоритм с полиномиальным временем работы, который из любого заданного кода индивидуальной задачи позволял бы извлечь описание этой компоненты.

Приведённое выше уточнение понятия «разумное кодирование», конечно, не является формальным определением. Нам не известен даже удовлетворительный способ, как дать такое определение. Хотя мнения большинства людей относительно разумности или неразумности конкретного кодирования той или иной задачи совпадают, отсутствие формального определения вызывает некоторые неудобства. Для преодоления этих неудобств можно было бы потребовать, чтобы условие задачи всегда состояло из фиксированного набора основных теоретико-множественных объектов. Мы не будем здесь накладывать этого требования. Однако для уточнения того, что мы имеем в виду, говоря о «разумной схеме кодирования», дадим краткое описание (которое при первом чтении можно опустить) одного из способов определения стандартной схемы кодирования.

Эта стандартная схема кодирования отображает индивидуальные задачи в *правильно построенные слова* (ППС)[1] в алфавите $\psi = \{0, 1, -, [,], (,), \}$. ППС определяется рекурсивно:

(1) Двоичная запись целого числа k, рассматриваемая как слово, состоящее из нулей и единиц (со знаком «$-$» перед словом, если k отрицательно), есть ППС, представляющее целое число k.

(2) Если x есть ППС, представляющее целое число k, то $[x]$ есть ППС, которое может использоваться как «имя» (например, «имя» вершины графа, элемента множества или города в задаче о коммивояжере).

(3) Если $x_1, x_2, ..., x_m$ – правильно построенные слова, представляющие объекты $X_1, X_2, ..., X_m$, то $(x_1, x_2, ..., x_m)$ есть ППС, представляющее последовательность $<X_1, X_2, ..., X_m>$.

Для указания схемы кодирования конкретной задачи распознавания, представленной в нашей стандартной записи, заметим, что если каждая

[1] В оригинале *structured strings*, что в буквальном переводе означает *структурированные цепочки* (символов алфавита). – *Прим. ред.*

компонента (или объект) условия представлена некоторым ППС, то все условия можно закодировать, пользуясь правилом (3). Таким образом, достаточно указать способ кодирования объекта каждого типа. При этом мы ограничимся целыми числами, «элементами без структуры» (вершины графа, элементы множества, города и т.д.), последовательностями, множествами, графами, конечными функциями и рациональными числами.

Правила (1) и (3) указывают, как кодировать целые числа и последовательности. Для кодирования элементов без структуры, присвоим им, согласно правилу (2), различные «имена», соблюдая при этом следующее условие: если индивидуальная задача содержит не более N элементов без структуры, то используются имена, величина которых не превосходит N. Объекты остальных четырёх типов кодируются следующим образом.

Множество объектов представляется в виде произвольно упорядоченной последовательности $< X_1, X_2, ..., X_m >$ элементов этого множества и кодируется ППС, соответствующим этой последовательности.

Граф с множеством вершин V и множеством рёбер E кодируется ППС (x, y), где x и y есть ППС, представляющие множества V и E соответственно (элементами E являются двухэлементные подмножества V, образующие рёбра).

Конечная функция $f : \{u_1, u_2, ..., u_m\} \to W$ кодируется ППС $((x_1, y_1),$ $(x_2, y_2),$ ...,$(x_m, y_m))$, где x_i – ППС для объекта u_i, а y_i – ППС для объекта $f(u_i) \in W$, $l \leqslant i \leqslant m$.

Рациональное число q кодируется ППС (x, y), где x и y – ППС для целых чисел a и b, таких, что $a/b = q$ и наибольших общий делитель a и b равен 1.

Приведённый выше список кодов объектов довольно хорошо иллюстрирует понятие разумного кодирования. Его достаточно в большинстве встречающихся случаев, хотя иногда было бы удобнее иметь более широкий список типов объектов. Однако для описания условий задач можно, не уменьшая общности, ограничиться только перечисленными объектами, потому что другие типы объектов всегда могут быть выражены через объекты, перечисленные выше.

Отметим, что указанных рецептов не хватает для того, чтобы построить один–*единственный* код для каждой индивидуальной задачи, но вполне достаточно для того, чтобы любое ППС, которое является кодом задачи, обладало определёнными структурными особенностями. Изменение имён исходных элементов или порядка элементов при описании множества может привести к тому, что одна и та же индивидуальная задача будет кодироваться различными правильно построенными словами. В действительности не имеет значения, какими ППС закодирована данная индивидуальная задача, так как любое слово можно декодировать и получить существенные компоненты индивидуальной задачи. Эта возможность учтена в данных выше определениях, например, при кодировании e каждая индивидуальная задача из Π может быть представлена несколько раз в языке $L[\Pi, e]$ кодов индивидуальных задач с положительным ответом на вопрос.

Прежде чем продолжить обсуждение, напомним, что стандартная схема кодирования приведена с целью иллюстрации возможных определений стандартных схем кодирования; она также служит примером «разумной» схемы кодирования. Ничто не мешает воспользоваться какой-нибудь другой схемой кодирования или построить для каждой интересующей нас задачи свою схему кодирования.

Если выбранная схема кодирования «эквивалентна» указанной выше (в том смысле, что существуют алгоритмы с полиномиальным временем работы, переводящие коды индивидуальных задач при одной схеме кодирования в коды этих же задач при другой и обратно), то эта схема кодирования также будет «разумной». Если выбрать схему кодирования, которая в этом смысле *не* эквивалентна приведённой выше, то можно по-прежнему доказывать различные результаты относительно этой схемы кодирования, однако для формулировки получаемых результатов уже нельзя будет пользоваться независимой от кодирования терминологией. В настоящей книге мы ограничимся только разумными схемами кодирования.

2.2. Детерминированные машины Тьюринга и класс P

Для того чтобы формализовать понятие «алгоритм», необходимо зафиксировать определённую модель процесса вычисления. Далее такой моделью будет служить *детерминированная одноленточная машина Тьюринга* (сокращённо ДМТ), которая схематически изображена на рис. 2.1.

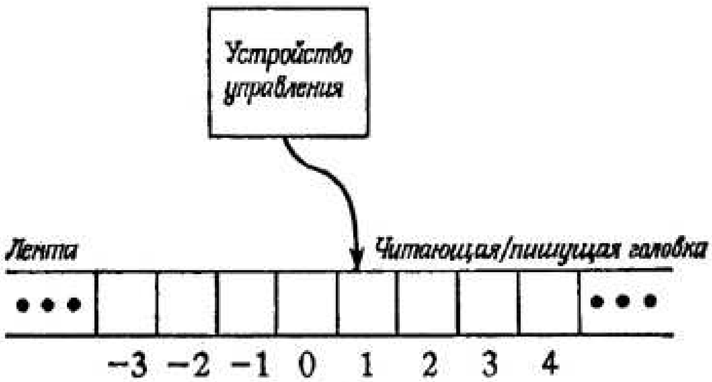


Рис. 2..1: *«Машина Тьюринга»*

Машина Тьюринга состоит из *управляющего устройства с конечным числом состояний, читающей/ пишущей головки*, которая может считывать и записывать символы и неограниченной в обе стороны *ленты*, разделённой на бесконечное число одинаковых ячеек, занумерованных целыми числами ..., -2, -1, 0, 1, 2, 3, ...

Программа для ДМТ, или ДМТ-программа, определяется следующими компонентами:

1. конечным множеством Γ *символов*, которые записываются на ленте, подмножеством $\Sigma \subset \Gamma$ *входных символов* и выделенным *пустым символом* $b \subset \Gamma \setminus \Sigma$;

2. конечным *множеством состояний* Q, в котором выделены начальное состояние q_0 и два *заключительных состояния* q_Y, q_N[2]);

3. *функцией перехода* $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma\{-1, 1\}$.

Работа программы определяется следующим образом. *Входом* для детерминированной программы является слово $x \in \Sigma^*$. Слово записывается на ленте в ячейках с номерами $1, 2, ..., |x|$ по одному символу в ячейке. Все другие ячейки в начальный момент времени содержат пустой символ и называются пустыми. Программа начинает работу, находясь в состоянии q_0, при этом читающая/пишущая головка находится над ячейкой с номером 1.

Далее процесс вычислений осуществляется последовательно, шаг за шагом. Если текущее состояние q есть q_Y или q_N, то процесс вычислений заканчивается, при этом результатом будет «да», если $q = q_Y$, и нет, если $q = q_N$. В противном случае текущее состояние принадлежит множеству $Q\ \{q_Y, q_N\}$, при этом головка читает на ленте некоторый символ $s \in \Gamma$ и определено значение $\delta(q, s)$. Предположим, что $\delta(q, s) = \delta(q', s', \Delta)$. В этом случае головка стирает s, пишет на этом месте s' и сдвигается на одну ячейку влево, если $\Delta = -1$, или на одну ячейку вправо, если $\Delta = +1$. Одновременно управляющее устройство переходит из состояния q в q'. На этом заканчивается один «шаг» процесса вычисления, и программа готова к выполнению следующего шага (если таковой имеется).

Пример простой ДМТ-программы M представлен на рис. 2.2. Функция перехода δ определена таблицей, где величина, записанная в γ-й строке и s-м столбце, есть значение $\delta(q, s)$. Рис. 2.3 иллюстрирует вычисление по программе M при входе $x = 10100$; здесь указаны состояние, расположение головки и содержимое непустых ячеек ленты до и после каждого шага.

Заметим, что это вычисление после восьми шагов оканчивается в состоянии q_Y, поэтому на входе 10100 ответом будет «да». В общем случае будем говорить, что программа M, имеющая входной алфавит Σ, принимает $x \in \Sigma^*$ в том и только в том случае, когда, будучи применённой ко входу x, она останавливается в состоянии q_Y. Язык L_M, *распознаваемый* программой M, задаётся следующим образом:

$$L_M = \{x \in \Sigma^* : M \text{ принимает } x\}.$$

Нетрудно видеть, что программа, представленная на рис. 2.2, распознаёт язык $\{x \in \{0, 1\}^*$: *два последних символа слова x являются нулями*$\}$.

$$\Gamma = \{0, 1, b\},\ \Sigma = \{0, 1\}$$
$$Q = \{q_0, q_1, q_2, q_3, q_Y, q_N\}$$

[2]От слов Yes – да, No – нет. – *Прим. перев.*

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

Рис. 2.2. Пример ДМТ-программы $M = (\Gamma, Q, \delta)$.

			∇						
q_0 :	...	b	1	0	1	0	0	b	...
				∇					
q_0 :	...	b	1	0	1	0	0	b	...
					∇				
q_0 :	...	b	1	0	1	0	0	b	...
						∇			
q_0 :	...	b	1	0	1	0	0	b	...
							∇		
q_0 :	...	b	1	0	1	0	0	b	...
								∇	
q_0 :	...	b	1	0	1	0	0	b	...
							∇		
q_1 :	...	b	1	0	1	0	0	b	...
						∇			
q_2 :	...	b	1	0	1	0	b	b	...
					∇				
q_y :	...	b	1	0	1	b	b	b	...

Рис. 2.3. Вычисление по программе M (см. рис. 2.2) при входе 10100.

Отметим, что это определение распознавания языка не требует, чтобы программа M останавливалась при всех входах из Σ^*; M обязана останавливаться лишь при входах из L_M. Если x принадлежит $\Sigma^* \setminus L_M$, то работа программы M на x может либо закончиться в состоянии q_N, либо может бесконечно продолжаться без остановки. Однако ДМТ-программа, соответствующая нашему пониманию алгоритма, должна останавливаться на всех словах входного алфавита. В этом смысле программа на рис. 2.2 является алгоритмической, так как, начиная работать на любом слове из символов 0 и 1, она будет останавливаться.

Соответствие между «распознаванием» языков и «решением» задач распознавания определяется следующим образом. Будем говорить, что ДМТ-программа M *решает* задачу распознавания Π при кодировании e, если M останавливается на всех словах, составленных из букв входного алфавита, и $L_M = L[\Pi, e]$. Программа на рис. 2.2 иллюстрирует это соответствие. Рассмотрим следующую задачу распознавания из теории чисел:

ДЕЛИМОСТЬ НА ЧЕТЫРЕ

УСЛОВИЕ. Дано положительное целое число N.

ВОПРОС. Существует ли положительное целое число m, такое, что $N = 4m$?

При стандартном кодировании целое число N представляется словом из 0 и 1, т.е. двоичной записью этого числа. Так как положительное целое число делится на 4 тогда и только тогда, когда последние две цифры двоичной записи этого числа являются нулями, то программа, изображённая на рис. 2.2., «решает» при нашем стандартном кодировании задачу ДЕЛИМОСТЬ НА ЧЕТЫРЕ.

Заметим для дальнейшего, что ДМТ-программой можно пользоваться также и для вычисления функций. Предположим, что программа M, имеющая входной алфавит Σ и ленточный алфавит Γ, останавливается при любом входе из Σ^*. Тогда M вычисляет *функцию* $f_M : \Sigma^* \to \Gamma^*$, которая для каждого $x \in \Sigma^*$ определяется следующим образом. Если программа M, начиная работать при входе x, останавливается, то в качестве $f_M(x)$ берётся слово, составленное из символов, записанных после остановки машины в ячейках с номерами 1, 2, 3, ..., включая последнюю непустую ячейку. Программа M, представленная на рис. 2.2, вычисляет функцию $f_M : \{0,1\}^* \to \{0,1,b\}^*$, которая отображает каждое слово $x \in \{0,1\}^*$ в слово $f_M(x)$, получаемое из x удалением двух крайних справа символов (если $|x| < 2$, то M выдаёт в качестве $f_M(x)$ пустое слово).

Хорошо известно, что ДМТ-программы способны решать задачи гораздо более сложные, чем в рассмотренном примере. Несмотря на то что ДМТ имеет только одну последовательную ленту и на каждом шаге может выполнять весьма ограниченную работу, ДМТ-программа может быть составлена так, что она выполнит (хотя и гораздо медленнее) любое вычисление, выполнимое обычной ЭВМ. Читателю, который интересуется тем, как это делается, можно посоветовать несколько хороших работ, например [388] или [215]. Читателя, который этим не интересуется, можно успокоить тем, что в данной книге не требуется умение программировать на ДМТ. Введение модели ДМТ объясняется только необходимостью дать формальный эквивалент понятию алгоритма и тем самым заложить основу для остальных определений.

Теперь можно определить понятие «временна́я сложность». *Время*, требуемое ДМТ-программой M для вычисления при входе x, есть число шагов, выполняемых до момента остановки. Если программа M останавливается на всех входах $x \in \Sigma^*$, то временну́ю сложность $T_M : Z^+ \to Z^+$ этой программы можно определить так:

$$T_M(n) = \max \left\{ m : \begin{array}{l} \text{существует такое слово } x \in \Sigma^*, |x| = n, \\ \text{что вычисление по программе } M \\ \text{на входе } x \text{ требует времени } m \end{array} \right\}.$$

Детерминированная программа M называется ***полиномиальной ДМТ-программой***, если существует такой полином p, что для всех $n \in Z^+$, $T_M(n) \leqslant$

$p(n)$.

Теперь можно формально определить первый важный класс языков – класс P. Он определяется следующим образом:

$P = \{L$: существует полиномиальная ДМТ-программа M, для которой $L = L_M\}$

Будем говорить, что задача распознавания Π принадлежит классу P при кодировании e, если $L[\Pi, e] \in P$, т.е. существует полиномиальная ДМТ-программа, которая «решает» задачу Π при кодировании e. Учитывая приведённые выше рассуждения об эквивалентности разумных кодирований, мы обычно не будем приводить конкретные схемы кодирования и будем просто говорить, что задача распознавания Π принадлежит классу P.

Термин «полиномиальный алгоритм» часто будет использоваться неформально. Формальным эквивалентом понятия полиномиального алгоритма является полиномиальная ДМТ-программа. Тем не менее если учесть указанную в гл. 1 эквивалентность «реалистичных» моделей ЭВМ относительно понятия «полиномиальное время работы», то формальное определение класса P можно сформулировать в терминах программ для любой из этих моделей, причём в результате получится один и тот же класс языков.

Таким образом, при доказательстве того, что определённые задачи могут быть решены полиномиальным алгоритмом, нет необходимости вдаваться в детали ДМТ. На самом деле, следуя общепринятой практике, мы будем обсуждать алгоритмы в машинно-независимом стиле, как будто они работают непосредственно с компонентами индивидуальной задачи (множествами, графами, числами и т.д.), а не с их кодами. Здесь подразумевается, что при наличии желания и терпения для любого полиномиального алгоритма можно построить соответствующую полиномиальную ДМТ-программу. Наши неформальные рассуждения следует рассматривать как указание пути такого построения, они должны убедить любого читателя, знакомого с основными типами задач, разрешимых за полиномиальное время на обычных ЭВМ.

2.3. Недетерминированное вычисление и класс *NP*

В настоящем разделе вводится второй важный класс языков (задач распознавания свойств) – класс *NP*. Прежде чем перейти к формальному определению в терминах языков и машин Тьюринга, полезно пояснить смысл понятия, лежащего в основе определения класса *NP*.

Рассмотрим задачу КОММИВОЯЖЕР, описание которой приведено в начале настоящей главы. В условии даны: множества городов, расстояния между ними и граница B; при этом спрашивается, существует ли проходящий через все города маршрут длины, не превосходящей B. Полиномиальный алгоритм решения этой задачи не известен. Предположим, однако, что относительно некоторой индивидуальной задачи кто-то получил ответ «да».

Если вы в этом сомневаетесь, то можно потребовать «доказательства» этого утверждения – предъявление маршрута, обладающего необходимыми свойствами. Имея предъявленное решение, нетрудно проверить, является ли оно на самом деле маршрутом, и если это так, то вычислить его длину, сравнить её с границей B и тем самым проверить соответствующее утверждение. Более того, эту «процедуру проверки» можно представить в виде алгоритма, временнáя сложность которого ограничена полиномом от $Length[I]$.

Другой пример задачи, которая обладает этим свойством,– задача ИЗОМОРФИЗМ ПОДГРАФУ из разд. 2.1. Пусть дана индивидуальная задача I, в которой указаны два графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$. Если ответом на вопрос задачи будет «да», то этот факт можно «доказать», предъявив подмножества $V' \subseteq V_1$ и $E' \subseteq E_1$ и взаимно-однозначную функцию $f : V_2 \to V'$, который обладают необходимыми свойствами. В этом случае верность утверждения опять-таки может быть легко установлена за полиномиальное от $Length[I]$ время путём простой проверки того, что V', E' и f удовлетворяют требованиям, сформулированным в задаче.

Именно понятие полиномиальной «проверяемости» позволяет выделить задачи класса *NP*. Отметим, что проверяемость за полиномиальное время не влечёт разрешимости за полиномиальное время. А именно, утверждая, что за полиномиальное время можно проверить ответ «да» для задачи КОММИВОЯЖЕР, мы не учитываем время, которое может понадобиться на поиск нужного маршрута среди экспоненциального числа всех возможных маршрутов. Мы лишь утверждаем, что по любому заданному маршруту для индивидуальной задачи I можно за полиномиальное время проверить, «доказывает» ли этот маршрут, что ответ на вопрос относительно индивидуальной задачи I есть «да».

Неформально класс *NP* можно определить с помощью понятия, которое мы будем называть *недетерминированным алгоритмом.* Такой алгоритм состоит из двух различных стадий – *стадии угадывания* и *стадии проверки.* По заданной индивидуальной задаче I на первой стадии происходит просто «угадывание» некоторой структуры S. Затем I и S вместе подаются в качестве входа на стадию проверки, которая выполняется обычным детерминированным образом и либо заканчивается ответом «да», либо заканчивается ответом «нет», либо продолжается бесконечно без остановки (как мы увидим позже, последние две возможности различать не обязательно). Недетерминированный алгоритм «решает» задачу распознавания Π, если для любой индивидуальной задачи $I \in D_\Pi$ выполнены следующие два свойства:

1. Если $I \in Y_\Pi$, то существует такая структура S, угадывание которой для входа I приведёт к тому, что стадия проверки, начиная работу на входе (I, S), закончится ответом «да».

2. Если $I \notin Y_\Pi$, то *не* существует такой структуры S, угадывание которой для входа I обеспечило бы окончание стадии проверки на входе (I, S) ответом «да».

Например, недетерминированный алгоритм решения задачи КОММИВОЯЖЕР можно было бы построить, используя в качестве стадии угадывания просто выбор произвольной последовательности городов, а в качестве стадии проверки – упомянутую выше полиномиальную процедуру «проверка доказательства» для задачи КОММИВОЯЖЕР. Очевидно, для любой индивидуальной задачи I найдётся такая догадка S, что результатом работы стадии проверки на входе (I, S) будет «да» в том и только в том случае, если для индивидуальной задачи I существует маршрут искомой длины.

Говорят, что недетерминированный алгоритм, решающий задачу распознавания П, работает в течение «полиномиального времени», если найдётся полином P, такой, что для любого $I \in Y_\Pi$ найдётся некоторая догадка S, приводящая на стадии детерминированной проверки на входе (I, S) к ответу «да» за время $p(Length[I])$. Отсюда следует, что «размер» угадываемой структуры S будет обязательно ограничен полиномом от $Length[I]$, так как на проверку догадки S может быть затрачено не более чем полиномиальное время.

Класс *NP*, определяемый неформально,– это класс всех задач распознавания П, которые при разумном кодировании могут быть решены недетерминированными ($N - nondeterministic$) алгоритмами за полиномиальное ($P - polynomial$) время. Пример, приведённый выше, показывает, что задача КОММИВОЯЖЕР принадлежит *NP*. У читателя не должно возникнуть трудностей при доказательстве аналогичного утверждения о задаче ИЗОМОРФИЗМ ПОДГРАФУ.

В подобных неформальных определениях термином «решает» следует, конечно, пользоваться осторожно. Должно быть ясно, что основное назначение «полиномиального недетерминированного алгоритма» состоит в объяснении понятия «проверяемости за полиномиальное время», а не в том, чтобы служить реалистическим методом решения задач распознавания свойств. При каждом входе такой алгоритм имеет не одно, а несколько возможных вычислений – по одному для каждой возможной догадки.

Имеется ещё одно важное отличие «решения» задачи распознавания недетерминированным алгоритмом от решения детерминированным алгоритмом, а именно в первом случае отсутствует симметрия между ответами «да» и «нет». Если задача: «Дано I; верно ли, что для I выполняется свойство X?» может быть решена полиномиальным (детерминированным) алгоритмом, то такое же утверждение справедливо и для дополнительной задачи: «Дано I, верно ли, что для I не выполняется свойство X?». Это следует из того, что детерминированный алгоритм останавливается при всех входах, поэтому достаточно поменять местами ответы «да» и «нет» (переставить состояния q_Y и q_N в ДМТ-программе).

Совершенно не очевидно, что то же самое верно для всех задач, разрешимых за полиномиальное время недетерминированными алгоритмами. Рассмотрим, например, дополнение задачи КОММИВОЯЖЕР: дано множество городов, расстояния между ними и граница B; верно ли, что нет маршрута, проходящего через все города и имеющего длину, не превосходящую B? Для выяснения, имеет ли поставленный вопрос ответ «да», не известен способ, который был бы короче, чем проверка всех (или почти всех) возможных маршрутов. Другими словами, не известен полиномиальный недетерминированный алгоритм решения этой дополнительной задачи. Та же самая ситуация имеет место для многих других задач из NP. Таким образом, принадлежность задачи Π классу P влечёт принадлежность дополнительной задачи классу P, но не известно, имеет ли место аналогичное утверждение для класса NP.

В заключение настоящего раздела мы формализуем приведённое определение в терминах языков и машин Тьюринга. Формальным эквивалентом недетерминированного алгоритма является программа для *недетерминированной одноленточной машины Тьюринга* (НДМТ). Для простоты будем пользоваться несколько нестандартной моделью НДМТ. (Версии более близкие к стандартным изложены в книгах [9, 215]. Полезным упражнением для читателя может оказаться проверка полиномиальной эквивалентности по отношению ко времени работы, между стандартными моделями и моделью изложенной ниже.)

Модель НДМТ, которой мы будем пользоваться, имеет в точности такую же структуру, как ДМТ; отличие состоит лишь в том, что НДМТ дополнена *угадывающим модулем со своей головкой*, которая может только записывать на ленту (см. рис. 2.4). Угадывающий модуль даёт информацию для записывания «догадки» и применяется исключительно с этой целью.

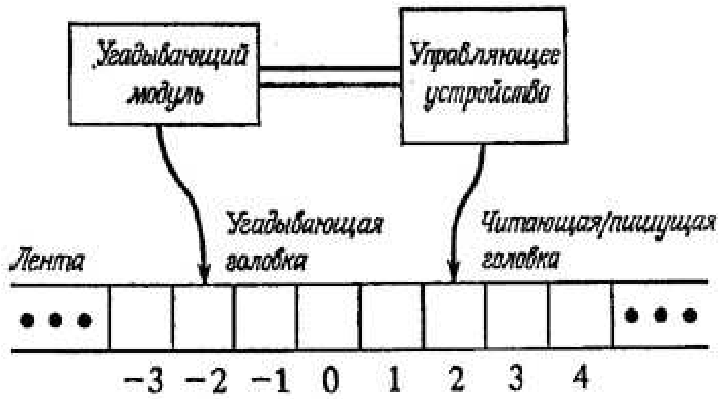


Рис. 2..2: *«Машина Тьюринга»*

Программа для НДМТ, или НДМТ-программа, определяется точно так же, как ДМТ-программа, при этом используются:

ленточный алфавит Γ, входной алфавит Σ, пустой символ b,

множество состояний Q, начальное состояние q_0, заключительные состояния q_Y и q_N,

функция перехода $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$.

Вычисление НДМТ-программы при входе $x \in \Sigma^*$ в отличие от вычисления ДМТ-программы имеет две различные стадии.

На первой стадии происходит «угадывание». В начальный момент времени входное слово x записывается в ячейках с номерами $1, 2, ..., |x|$ (остальные ячейки пусты), читающая/ пишущая головка «смотрит» на ячейку с номером 1, пишущая головка угадывающего модуля смотрит на ячейку с номером -1, а устройство управления «пассивно». Затем угадывающий модуль начинает управлять угадывающей головкой, которая делает один шаг в каждый момент времени и либо пишет в находящейся под ней ячейке одну из букв алфавита Γ и сдвигается при этом на одну ячейку влево, либо останавливается. В последнем случае угадывающий модуль переходит в пассивное состояние, а управляющее устройство начинает работу в состоянии q_0. Угадывающий модуль решает, продолжить ли работу (перейти ли в пассивное состояние, какую букву из Γ написать на ленте), причём делает это совершенно произвольно. Таким образом, угадывающий модуль до момента окончания своей работы может написать любое слово из Γ^* и в действительности может никогда не остановиться.

Стадия проверки начинается в тот момент, когда управляющее устройство переходит в состояние q_0. Начиная с этого момента, вычисление НДМТ-программы осуществляется в точности по тем же правилам, что и ДМТ-программы. Угадывающий модуль и его головка в вычислении больше не участвуют, выполнив свою роль, т.е. записав на ленте слово-догадку. Конечно, слово-догадка может (и обычно будет) просматриваться читающей/ пишущей головкой в процессе проверки. Вычисление заканчивается тогда, когда управляющее устройство перейдёт в одно из двух заключительных состояний (q_Y или q_N); оно называется *принимающим вычислением*, если остановка происходит в состоянии q_Y. Все остальные вычисления, заканчивающиеся или нет, называются *непринимающими*.

Отметим, что любая НДМТ-программа M может иметь бесконечное число возможных вычислений при данном входе x, по одному для каждого слова-догадки из Γ^*. Будем говорить, что НДМТ-программа M *принимает* x, если по крайней мере одно из её вычислений на входе x является принимающим. Язык, *распознаваемый* программой M,– это язык

$$L_M = \{x \in \Sigma^*; M \text{ принимает } x\}.$$

Время, требующееся недетерминированной программе M для того, чтобы принять слово $x \in L_M$, – это минимальное число шагов, выполняемых на стадиях угадывания и проверки до момента достижения заключительного состояния q_Y, где минимум берётся по всем *принимающим* вычислениям программы M на входе x. Временна́я сложность НДМТ-программы M – это функция $T_M : Z^+ \to Z^+$, определяемая следующим образом:

$$T_M(n) = \max\left(\{1\} \cup \left\{ m : \begin{array}{c} \text{существует } x \in L_M, |x| = n, \\ \text{такое, что время принятия } x \\ \text{программой } M \text{ равно } m \end{array} \right\}\right).$$

Заметим, что временна́я сложность программы M зависит только от числа шагов, выполняемых в принимающих вычислениях, а также, что мы полагаем $M(n)$ равным 1, если нет ни одного входа длины n, принимаемого программой M.

НДМТ-программа называется *НДМТ-программой с полиномиальным временем работы*, если найдётся полином p, такой, что $T_M(n) \leqslant p(n)$ для всех $n \geqslant 1$. Наконец, класс *NP* формально определяется так:

$$NP = \left\{ L : \begin{array}{l} \text{существует НДМТ-программа } M \\ \text{с полиномиальным временем} \\ \text{работы, такая, что } L_M = L \end{array} \right\}.$$

Нетрудно установить взаимосвязь между этими формальными определениями и предшествующими неформальными. Единственный момент, который следует специально оговорить, состоит в следующем. Если, как принято считать, недетерминированный алгоритм угадывает структуру S, зависящую некоторым образом от условия рассматриваемой задачи, то угадывающий модуль НДМТ, напротив, полностью игнорирует входное слово. Хотя любое слово из Γ^* может быть получено в результате работы угадывающего модуля, это не существенно, так как НДМТ-программу всегда можно сконструировать так, что стадия проверки будет начинаться с проверки того, соответствует ли догадка (при той неявной интерпретации, которую наша программа даёт словам) подходящей структуре для заданного входа. В противном случае программа может сразу же перейти в заключительное состояние q_N.

Будем говорить, что задача распознавания принадлежит классу *NP* при схеме кодирования e, если $L[\Pi, e] \in NP$. Как и в случае класса P, мы будем просто говорить, что Π лежит в *NP*, не упоминая конкретную схему кодирования, когда ясно, что некоторая разумная схема кодирования задачи Π даст язык, принадлежащий *NP*.

Более того, поскольку любая реалистичная модель вычислительного устройства может быть дополнена аналогом рассмотренного «угадывающего модуля с пишущей головкой», то наши формальные определения можно было бы перефразировать для любой другой стандартной модели вычислительного устройства. В силу того что все такие модели эквивалентны с точностью до детерминированных полиномиальных вычислений, все получаемые при этом версии класса *NP* будут идентичны. Таким образом, правомерно сделанное ранее предложение отождествить формально определённый класс *NP* с классом всех задач распознавания, «разрешимых» недетерминированными алгоритмами с полиномиальным временем работы.

В качестве подготовки к определению третьего, самого важного для настоящей книги класса – класса *NP*-полных задач в следующем разделе мы обсудим взаимоотношения между классами P и *NP*.

2.4. Взаимоотношения между классами P и NP

Вопрос о взаимоотношении классов P и NP имеет фундаментальное значение для теории NP-полных задач. Одно соотношение, которое неявно присутствовало в проводившихся ранее рассуждениях, но до настоящего момента явно не формулировалось, заключается в том, что $P \subseteq NP$. Всякая задача распознавания, разрешимая за полиномиальное время детерминированным алгоритмом, разрешима также за полиномиальное время недетерминированным алгоритмом. Чтобы убедиться в этом, достаточно заметить, что любой детерминированный алгоритм может быть использован в качестве стадии проверки недетерминированного алгоритма. Если $\Pi \in P$ и A – произвольный детерминированный полиномиальный алгоритм решения Π. то полиномиальный недетерминированный алгоритм для Π можно получить, воспользовавшись A в качестве стадии проверки и игнорируя стадию угадывания. Таким образом, из $\Pi \in P$ следует, что $\Pi \in NP$.

Из проводившихся ранее рассуждений можно понять, что есть много причин считать это включение строгим, т.е. считать, что P не совпадает с NP. Полиномиальные недетерминированные алгоритмы определённо оказываются более мощными, чем полиномиальные детерминированные алгоритмы, и не известны общие методы их превращения в детерминированные полиномиальные алгоритмы. В действительности самый сильный из известных в настоящее время результатов состоит в следующем:

Теорема 2.1. *Если $\Pi \in NP$, то существует такой полином p, что Π может быть решена детерминированным алгоритмом с временно́й сложностью $O(2^{p(n)})$.*

Доказательство. Пусть A – полиномиальный недетерминированный алгоритм решения задачи Π и $q(n)$ – полином, ограничивающий временну́ю сложность алгоритма A. (Не умаляя общности, можно считать, что q может быть вычислен за полиномиальное время; этого можно добиться, взяв, например, ($q(n) = c_1 n^{c_2}$ при достаточно больших константах c_1 и c_2).

По определению класса NP, для каждого принимаемого входа длины n найдётся некоторое слово-догадка (в алфавите Γ символов ленты) длины не более $q(n)$, такое, что в алгоритме A стадия проверки даёт при рассматриваемом входе ответ «да» не более чем за $q(n)$ шагов. Таким образом, общее число догадок, которые нужно рассмотреть, не превосходит $k^{q(n)}$, где $k = |\Gamma|$ (если слово-догадка короче $q(n)$, то его можно дополнить пустыми символами и рассматривать как слово длины $q(n)$).

Теперь можно детерминированным образом выяснить, имеет ли алгоритм A на заданном входе длины n принимающее вычисление. Для этого достаточно на каждой из $k^{q(n)}$ возможных догадок запустить детерминированную стадию проверки алгоритма A и позволить ей работать до того момента, пока она не остановится или не сделает $q(n)$ шагов. Этот моделирующий алгоритм даст ответ «да», если ему встретится слово-догадка, приводящее к принимающему вычислению длины не более $q(n)$, и ответ «нет» –

в противном случае. Такой алгоритм, очевидно, будет детерминированным алгоритмом решения задачи П. Более того, его временна́я сложность, по существу, равна $q(n)^* k^{q(n)}$; хотя это экспонента, но при подходящем выборе полинома p сложность не превосходит $O(2^{p(n)})$. ■

Безусловно, процесс моделирования, предложенный в доказательстве теоремы 2.1, можно в некоторой степени ускорить с помощью метода ветвей и границ или метода обратного поиска или с помощью более тщательного перебора, когда избегаются, очевидно, ненужные слова-догадки. Тем не менее, несмотря на значительную экономию, которая может быть при этом достигнута, не известен метод, осуществляющий такое моделирование быстрее, чем за экспоненциальное время.

Таким образом, способность недетерминированного алгоритма проверить за полиномиальное время экспоненциальное число возможностей может навести на мысль, что полиномиальные недетерминированные алгоритмы являются более мощным средством, чем полиномиальные детерминированные алгоритмы. В самом деле, для многих частных задач класса *NP*, таких, как КОММИВОЯЖЕР, ИЗОМОРФИЗМ ПОДГРАФУ, и большого числа других задач не найдено полиномиального детерминированного алгоритма, несмотря на упорные усилия многих известных исследователей.

Не удивляет поэтому широко распространённое мнение, что $P \neq NP$, хотя пока доказательство этой гипотезы отсутствует.

Конечно, скептик может сказать, что неудача в доказательстве гипотезы $P \neq NP$ является столь же сильным аргументом в пользу соотношения $P = NP$, как и неудача в поиске соответствующих полиномиальных алгоритмов – в пользу противоположного утверждения. Задачи всегда кажутся труднорешаемыми, пока не найдены эффективные алгоритмы их решения. Однако даже скептик, вероятно, согласится, что при существующем в настоящее время уровне знаний, по-видимому, более разумно работать при соглашении, что $P \neq NP$, чем пытаться доказать противоположное. Во всяком случае, на основе накопленного опыта мы будем представлять себе класс *NP* так, как он изображён на рис. 2.5, ожидая (хотя и не с полной уверенностью), что затенённая область, обозначающая $NP \setminus P$, не пуста.

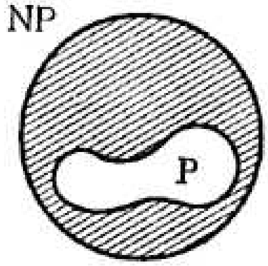


Рис. 2..3: *«Гипотетическая картина класса NP»*

2.5. Полиномиальная сводимость и NP-полные задачи

Если P не совпадает с NP, то различие между P и $NP \setminus P$ очень существенно. Все задачи из P могут быть решены полиномиальными алгоритмами, а все задачи из $NP \setminus P$ труднорешаемы. Поэтому, если $P \neq NP$, то для каждой конкретной задачи $\Pi \in NP$ важно знать, какая из этих двух возможностей реализуется.

Конечно, до тех пор пока не доказано, что $P \neq NP$, нет никакой надежды показать, что некоторая конкретная задача принадлежит классу $NP \setminus P$. По этой причине цель теории NP-полных задач заключается в доказательстве более слабых результатов вида: «если $P \neq NP$, то $\Pi \in NP \setminus P$». Хотя доказательство таких условных результатов может показаться столь же трудным, как и безусловных, однако, как мы увидим в дальнейшем, имеются несложные методы доказательства. Такие условные результаты можно рассматривать как подтверждение труднорешаемости с той же степенью уверенности, с какой мы считаем, что класс P отличается от NP.

Основная идея подобного условного подхода основана на понятии полиномиальной сводимости. Будем говорить, что имеет место *полиномиальная сводимость* языка $L_1 \subseteq \Sigma_1^*$ к языку $L_2 \subseteq \Sigma_2^*$, если существует функция $f : \Sigma_1^* \to \Sigma_2^*$, удовлетворяющая двум условиям[3]:

1. Существует ДМТ-программа, вычисляющая f с временнóй сложностью, ограниченной полиномом.

2. Для любого $x \in \Sigma_1^*$, $x \in L_1$ в том и только в том случае, если $f(x) \in L_2$.

Если L_1 полиномиально сводится к L_2, то будем писать $L_1 \propto L_2$ и говорить «L_1 сводится к L_2» (опуская слово «полиномиально»).

Важность понятия полиномиальная сводимость вытекает из следующей леммы:

`Лемма 2.1.`
Если $L_1 \propto L_2$, то из $L_2 \in P$ следует, что $L_1 \in P$. (Эквивалентное утверждение: из $L_1 \notin P$ следует, что $L_2 \notin P$.)

Доказательство. Пусть Σ_1 и Σ_2 – алфавиты языков L_1 и L_2 соответственно, функция $f : \Sigma_1^* \to \Sigma_2^*$ осуществляет полиномиальную сводимость L_1 к L_2; M_f – полиномиальная ДМТ- программа, вычисляющая f, и M_2 – полиномиальная ДМТ-программа, распознающая L_2. Полиномиальная ДМТ-программа, распознающая L_1, может быть получена композицией программы M_f и M_2. Ко входу $x \in \Sigma_1^*$ вначале применяется M_f, чтобы построить $f(x) \in \Sigma_2^*$. Затем к $f(x)$ применяется программа M_2, выясняющая, верно ли, что $f(x) \in L_2$. Поскольку $x \in L_1$ тогда и только тогда, когда $f(x) \in L_2$, то это описание даёт ДМТ-программу, распознающую L_1. То,

---

[3] Часто говорят, что язык L_1 полиномиально сводится к языку L_2, а функция f реализует эту сводимость. – *Прим. ред.*

что время работы этой программы ограничено полиномом, немедленно следует из полиномиальности программ M_f и M. Точнее, если p_f и p_2 – полиномы, ограничивающие время работы программ M_f и M_2 соответственно, то $|f(x)| \leqslant p_f(|x|)$, а время работы только что построенной программы, как не трудно видеть, ограничено функцией $O(p_f(|x|) + p_2(p_f(|x|))$, которая является полиномом от $|x|$. ■

Если Π_1 и Π_2 – задачи распознавания, а e_1 и e_2 – их схемы кодирования, то будем писать $\Pi_1 \propto \Pi_2$ (относительно заданных схем кодирования), если существует полиномиальная сводимость языка $L[\Pi_1, e_1]$ к $L[\Pi_2, e_2]$. Когда будет действовать стандартное предположение о «разумности» используемых схем кодирования, упоминание о конкретных схемах кодирования, как обычно, будет опускаться. Таким образом, на уровне задач полиномиальная сводимость задачи распознавания Π_1 к задаче распознавания Π_2 означает наличие функции $f : D_{\Pi_1} \to D_{\Pi_2}$, удовлетворяющей двум условиям:

1. f вычисляется полиномиальным алгоритмом и
2. для всех $I \in D_\Pi, I \in Y_{\Pi_1}$ тогда и только тогда, когда $f(I) \in Y_{\Pi_2}$.

Чтобы получить конкретное представление о содержательном смысле этого определения, рассмотрим пример. Пусть $G = (V, E)$ – граф с множеством вершин V и множеством рёбер E. *Простым циклом* в G называется такая последовательность $< v_1, v_2, ..., v_k >$ различных вершин из V, что $\{v_i, v_{i+1}\} \in E$, $1 \leqslant i < k$, и $\{v_k, v_1\} \in E$.

Гамильтоновым циклом в G называется простой цикл, содержащий все вершины графа G. Задача ГАМИЛЬТОНОВ ЦИКЛ определяется следующим образом:

ГАМИЛЬТОНОВ ЦИКЛ

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Верно ли, что G содержит гамильтонов цикл?

Читатель, несомненно, заметил определённую связь между этой задачей и задачей распознавания КОММИВОЯЖЕР. Покажем, что ГАМИЛЬТОНОВ ЦИКЛ (ГЦ) сводится к КОММИВОЯЖЕРУ (КМ). Для этого требуется указать функцию f, которая отображает каждую индивидуальную задачу из ГЦ в соответствующую индивидуальную задачу из КМ и удовлетворяет двум условиям, которые требуются от полиномиальной сводимости.

Функция f определяется очень просто. Пусть $G = (V, E)$, $|V| = m$ означает фиксированную индивидуальную задачу из ГЦ. Соответствующая задача из КМ строится так: множество городов C совпадает с V; для любых двух городов $v_i, v_j \in C$ расстояние $d(v_i, v_j)$ между ними полагаем равным 1, если $\{v_i, v_j\} \in E$, и 2 в противном случае. Граница B для длины искомого маршрута берётся равной m.

Легко показать (на неформальном уровне), что эта функция f осуществляет сводимость и может быть вычислена за полиномиальное время. Для вычисления $m(m-1)/2$ расстояний $d(v_i, v_j)$ необходимо лишь выяснить, принадлежит ли $\{v_i, v_j\}$ множеству E или нет. Поэтому первое требование полиномиальности сводимости выполнено. Для проверки второго требования необходимо показать, что G содержит гамильтонов цикл тогда и только тогда, когда в $f(G)$ имеется проходящий через все города маршрут длины, не превосходящей B. Вначале допустим, что $<v_1, v_2, ..., v_m>$ – гамильтонов цикл в G. Тогда $<v_1, v_2, ..., v_m>$ – маршрут в $f(G)$, а его длина равна m ($m = B$), так как расстояние между любыми соседними городами маршрута равно 1, поскольку оно соответствует ребру в G. Наоборот, предположим, что $<v_1, v_2, ..., v_m>$ – маршрут в $f(G)$, длина которого не превосходит B. Поскольку расстояние между любыми двумя городами в $f(G)$ равно либо 1, либо 2 и при вычислении длины маршрута суммируется ровно m таких расстояний, то из равенства $B = m$ следует, что расстояние между каждой парой соседних городов в маршруте равно 1. По определению $f(G)$, отсюда следует, что $\{v_i, v_{i+1}\}$, $1 \leqslant i < m$, и $\{v_m, v_i\}$ является рёбрами графа G и, следовательно, $<v_1, v_2, ..., v_m>$ – гамильтонов цикл в G.

Таким образом, мы доказали, что ГЦ $\propto$ КМ. Хотя это доказательство гораздо проще многих доказательств, которые будут рассмотрены в дальнейшем, оно содержит все существенные элементы доказательства полиномиальной сводимости и на неформальном уровне может служить моделью для построения таких доказательств.

Значение леммы 2.1 для задач распознавания теперь может быть проиллюстрировано на примере сводимости ГЦ $\propto$ КМ. По существу, мы пришли к следующему заключению: если задача КОММИВОЯЖЕР может быть решена полиномиальным алгоритмом, то этот факт верен и для задачи ГАМИЛЬТОНОВ ЦИКЛ, а если ГЦ – труднорешаемая задача, то и КМ также труднорешаемая задача. Таким образом, лемма 2.1 позволяет интерпретировать сводимость ГЦ $\propto$ КМ как утверждение, что задача Π_2 «не проще» задачи Π_1.

Отношение «полиномиальной сводимости» особенно удобно, поскольку оно является транзитивным. Это устанавливает следующая лемма.

Лемма 2.2. *Если* $L_1 \propto L_2$ *и* $L_2 \propto L_3$, *то* $L_1 \propto L_3$.

Доказательство.

Пусть Σ_1, Σ_2 и Σ_3 – алфавиты языков L_1, L_2 и L_3 соответственно, функция $f_1 : \Sigma_1^* \to \Sigma_2^*$ реализует полиномиальную сводимость L_1 к L_2, а $f_2 : \Sigma_2^* \to \Sigma_3^*$ – полиномиальную сводимость L_2 к L_3. Тогда функция $f : \Sigma_1^* \to \Sigma_3^*$, которая для всех $x \in \Sigma^*$ определяется соотношением $f(x) = f_2(f_1(x))$, реализует искомую сводимость языка L_1 к L_3. В самом деле, $f(x) \in L_3$ тогда и только тогда, когда $x \in L_1$, а вычислимость f за полиномиальное время получается с помощью рассуждений, аналогичных рассуждениям, использованным при доказательстве леммы 2.1. ■

Теперь можно сказать, что языки L_1 и L_2 (соответственно задачи распознавания Π_1 и Π_2) ***полиномиально эквивалентны***, если они сводятся друг к другу, т.е. $L_1 \propto L_2$ и $L_2 \propto L_1$ (имеет место сводимость $\Pi_1 \propto \Pi_2$ и $\Pi_2 \propto \Pi_1$). Лемма 2.2 утверждает, что это отношение является отношением эквивалентности, а также, что отношение «$\propto$» определяет частичное упорядочение возникающих классов эквивалентности языков (задач распознавания). На самом деле класс P – это «наименьший» относительно этого частичного порядка класс эквивалентности и с вычислительной точки зрения его можно рассматривать как класс «самых лёгких» языков (задач распознавания). Класс NP-полных языков (задач) даёт нам другой класс эквивалентности, который характеризуется тем, что он содержит «самые трудные» языки (задачи распознавания) из NP.

Язык L называется NP-полным, если $L \in NP$ и любой другой язык $L' \in NP$ сводится к L. Говоря неформально, задача распознавания Π называется NP-полной, если $\Pi \in NP$ и любая другая задача распознавания $\Pi' \in NP$ сводится к Π. Таким образом лемма 2.1 позволяет отождествить NP-полные задачи с «самыми трудными задачами из NP». Если хотя бы одна NP-полная задача может быть решена за полиномиальное время, то и все задачи из NP также могут быть решены за полиномиальное время. Если хотя бы одна задача из NP труднорешаема, то и все NP-полные задачи труднорешаемы. Следовательно, любая NP-полная задача Π обладает свойством, которое сформулировано в начале настоящего раздела: если $P \neq NP$, то $\Pi \in NP \setminus P$. Точнее, $\Pi \in P$ тогда и только тогда, когда $P = NP$.

В предположении $P \neq NP$ можно дать более точную картину класса NP, см. рис. 2.6.

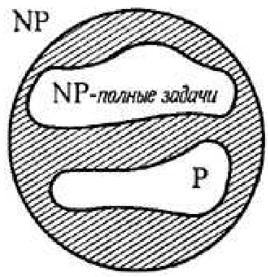


Рис. 2..4: *«Уточнённая картина класса NP)»*

Заметим, что NP не просто разделяется на две области: «класс P» и «класс NP-полных задач». Как мы увидим в гл. 7, если P отличен от NP, то должны существовать задачи из NP, неразрешимые за полиномиальное время не являющиеся NP-полными.

Мы будем интересоваться в основном NP-полными задачами. Хотя в начале настоящего раздела говорилось, что имеются простые методы доказательства NP-полноты задач, требования, только что описанные нами, как будто свидетельствуют об обратном. Нужно доказать, что *любая* задача из NP сводится к некоторому кандидату на NP-полную задачу. Совершенно неясно, как это можно было бы доказать. Априори, не очевидно даже, что существует хотя бы одна NP-полная задача.

Следующая лемма, которая немедленно следует из определений и транзитивности отношения $\propto$, показывает, что вопрос сильно упростился бы, если бы была известна по крайней мере одна *NP*-полная задача.

Лемма 2.3. *Если L_1 и L_2 принадлежат классу NP, L_1 – это NP- полный язык и $L_1 \propto L_2$, то L_2 также NP-полный язык.*

Доказательство. Так как $L_2 \in NP$, то достаточно показать, что для всякого $L' \in NP$ L' сводится к L_2. Рассмотрим любой язык $L' \in NP$. Так как L_1 – это *NP*-полный язык, то L' сводится к L_1. В силу транзитивности отношения $\propto$, из сводимости $L_1 \propto L_2$ следует $L' \propto L_2$. ∎

На уровне задач распознавания эта лемма указывает простой путь доказательства *NP*-полноты новой задачи Π, если известна хотя бы одна *NP*-полная задача. Для того чтобы доказать *NP*-полноту задачи Π, достаточно показать, что

1. $\Pi \in NP$.
2. Какая-то одна известная *NP*-полная задача Π' сводится к Π.

Однако, прежде чем можно будет воспользоваться этим методом доказательства, необходимо найти некоторую исходную *NP*-полную задачу. Такую задачу даёт нам фундаментальная теорема Кука, которую мы сформулируем и докажем в следующем разделе.

2.6. Теорема Кука

Честь быть «первой» *NP*-полной задачей выпала на долю задачи распознавания из булевской логики, задачи, которую обычно называют ВЫПОЛНИМОСТЬ (сокращённо ВЫП). Термины, использующиеся для её описания, определяются следующим образом.

Пусть $U = \{u_1, u_2, ..., u_m\}$ – множество булевских *переменных.* Под *набором значений истинности* на множестве U будем понимать функцию $t : U \to \{T, F\}$[4]. Если $t(u) = T$, то будем говорить, что u принимает значение «истина» относительно t; если $t(u) = F$, то будем говорить, что u принимает значение «ложь». Если u – переменная из U, то u и $\bar{u}$ назовём литералами из U. Литерал u принимает значение «истина» относительно t в том и только в том случае, если переменная u принимает значение «истина» относительно t; литерат $\bar{u}$ принимает значение истина в том и только в том случае, если переменная u принимает значение «ложь».

Дизъюнкцией над U назовём множество литералов над U, например $\{u_1, \bar{u}_3, u_8\}$. Она представляет дизъюнкцию этих литералов и называется *выполненной* при некотором наборе значений истинности тогда и только тогда, когда при рассматриваемом наборе значений истинности хотя бы один из её членов принимает значение «истина». В нашем примере дизъюнкция будет выполнена относительно t, если одновременно *не окажется,*

[4]От слов *Truth* – истина, *False* – ложь (англ.). – *Прим. перев.*

что $t(u_1) = F$, $t(u_3) = T$, $t(u_8) = F$. Набор C дизъюнкций над U называется ***выполнимым*** в том и только в том случае, если найдётся некоторый набор значений истинности на множестве U, такой, что одновременно выполнены все дизъюнкции из C. Такой набор значений истинности называется ***выполняющим набором значений истинности*** для C. Задача ВЫПОЛНИМОСТЬ формулируется следующим образом:

ВЫПОЛНИМОСТЬ

УСЛОВИЕ. Заданы множество переменных U и набор C дизъюнкций над U.

ВОПРОС. Существует ли выполняющий набор значений истинности для C?

Например, пусть $U = \{u_1, u_2\}$ и $C = \{\{u_1, \bar{u}_2\}, \{\bar{u}_1, u_2\}\}$. Это индивидуальная задача из ВЫП, ответ на которую есть «да». Выполняющее задание значений истинности определяется так: $t(u_1) = t(u_2) = T$. С другой стороны, заменив C на $C' = \{\{u_1, u_2\}, \{u_1, u_2\}, \{\bar{u}_1\}\}$, получим пример индивидуальной задачи, ответ на которую есть «нет», поскольку C' невыполнима.

Теперь можно сформулировать фундаментальную теорему Кука [91].

`Теорема 2.2.` *(теорема Кука). Задача ВЫПОЛНИМОСТЬ есть NP-полная задача.*

Доказательство 1.

Легко видеть, что ВЫП лежит в классе *NP*. Недетерминированному алгоритму для её решения достаточно указать набор значений истинности на исходном множестве переменных и осуществить проверку того, что этот набор значений выполняет все дизъюнкции из исходного набора C. Всё это легко сделать (недетерминированным образом) за полиномиальное время. Таким образом, первое требование, которому должны удовлетворять *NP*-полные задачи, выполнено.

Для того чтобы проверить выполнение второго требования, вернёмся к уровню языков, т.е. к представлению ВЫП языком $L_{\text{ВЫП}} = L[\text{ВЫП}, \text{e}]$ для некоторой разумной схемы кодирования e. Необходимо показать, что для всех языков $L \in NP$ имеет место соотношение $L \propto L_{\text{ВЫП}}$. Разные языки из *NP* могут сильно отличаться; число этих языков бесконечно, поэтому невозможно указать отдельное сведение для каждого из них. Однако каждый язык из *NP* может быть представлен в стандартном виде, а именно НДМТ-программой, работающей полиномиальное время и распознающей этот язык.

Такое представление позволяет иметь дело с общей НДМТ-программой, время работы которой полиномиально, и получить общую сводимость языка, распознаваемого этой программой, к $L_{\text{ВЫП}}$. Если эту общую сводимость специализировать для конкретной НДМТ-программы M, распознающей L_M, то получится искомое полиномиальное сведение L_M к $L_{\text{ВЫП}}$. Таким образом, по существу, для всех $L \in NP$ будет представлено общее доказательство того, что $L \propto L_{\text{ВЫП}}$.

Вначале рассмотрим произвольную НДМТ-программу M с полиномиальным временем работы, которая имеет компоненты Γ, Σ, b, Q, q_0, q_Y, q_N, δ и распознаёт язык $L = L_M$. Кроме того, пусть $p(n)$ – полином с целыми коэффициентами, ограничивающий сверху временнýю сложность $T_M(n)$. (Не умаляя общности, можно считать, что $p(n) \geqslant n$ для всех $n \in Z^+$.) Функция f_L, реализующая общую сводимость, будет описана в терминах M, Γ, Σ, b, Q, q_0, q_Y, q_N, δ и p.

Удобно рассматривать f_L как отображение из множества слов в алфавите Σ в индивидуальные задачи из ВЫП, а не в слова в алфавите Σ, которые кодируют задачи из ВЫП, так как детали, связанные с кодированием, могут быть легко восполнены. Таким образом, функция f_L будет обладать тем свойством, что для всех $x \in \Sigma^*$ принадлежность $x \in L$ имеет место в том и только в том случае, если для набора дизъюнкций $f_L(x)$ имеется выполняющий набор значений истинности. Ключом к построению функции f_L является использование некоторого набора дизъюнкций для записи утверждения, что x принимается НДМТ-программой M, т.е. утверждение $x \in L$.

Если входное слово $x \in \Sigma^*$ принимается программой M, то для x существует принимающее вычисление программы M, такое, что число шагов на стадии проверки и число символов в слове-догадке ограничены величиной $p(n)$, где $n = |x|$. В таком вычислении будут участвовать лишь ячейки с номерами от $-p(n)$ до $p(n)+1$. Это следует из того, что читающая/ пишущая головка начинает работу в ячейке с номером 1 и на каждом шаге сдвигается не более чем на 1 ячейку. Проверяющее вычисление полностью определяется заданием в каждый момент времени содержания ячеек с указанными номерами, внутреннего состояния и положения читающей/ пишущей головки. Далее, поскольку в проверяющем вычислении имеется не более $p(n)$ шагов, то необходимо учесть не более $p(n) + 1$ моментов времени. Это позволяет полностью описать такое вычисление, используя полиномиально ограниченное число булевских переменных и некоторый набор значений истинности на их множестве.

Множество переменных U, участвующих в описании функции f_L, предназначено именно для указанных целей. Перенумеруем элементы множеств Q и Γ следующим образом: Q : q_0, $q_1 = q_Y$, $q_2 = q_N$, $q_3, ..., q_r$, где $r = |Q|$; Γ: $s_0 = b$, s_1, s_2, ..., s_v, где $v = |\Gamma| - 1$. В дальнейших рассуждениях будут участвовать три типа переменных, каждому из которых придаётся определённый смысл согласно рис. 2.7. При этом фраза «в момент времени i» служит сокращением точного выражения «после выполнения i-го шага проверяющего вычисления».

Переменная	Пределы изменения индексов	Придаваемый смысл
$Q[i,k]$	$0 \leqslant i \leqslant p(n)$ $0 \leqslant k \leqslant r$	В момент времени i программа M находится в состоянии q_k
$H[i,j]$	$0 \leqslant i \leqslant p(n)$ $-p(n) \leqslant j \leqslant p(n)+1$	В момент времени i читающая/ пишущая головка просматривает ячейку с номером j
$S[i,j,k]$	$0 \leqslant i \leqslant p(n)$ $-p(n) \leqslant j \leqslant p(n)+1$ $0 \leqslant k \leqslant v$	В момент времени i в j-й ячейке записан символ s_k

Рис. 2.7. Переменные набора дизъюнкций $f_L(x)$ и придаваемый им смысл.

Вычисление программы M очевидным образом индуцирует на множестве этих переменных набор значений истинности, если принять соглашение, что при окончании вычисления раньше момента времени $p(n)$ конфигурация остаётся неизменной во все моменты времени после остановки, т.е. сохраняются заключительное состояние, положение головки и запись на ленте. В нулевой момент времени запись на ленте состоит из входного слова x, расположенного в ячейках с номерами от 1 до n, и слова-догадки w в ячейках с номерами от -1 до $|w|$, при этом все остальные ячейки пусты.

С другой стороны, произвольный набор значений истинности этих переменных не обязательно соответствует какому-то вычислению, а тем более принимающему. При произвольном наборе значений истинности переменных в некоторой ячейке могли бы быть одновременно записаны несколько различных символов, машина могла бы находиться одновременно в нескольких различных состояниях, а читающая/ пишущая головка могла бы одновременно просматривать любое подмножество ячеек с номерами от $-p(n)$ до $p(n)+1$. Описание функции f_L осуществляется посредством построения такого набора дизъюнкций, содержащего перечисленные переменные, что набор значений истинности будет *выполняющим* тогда и только тогда, когда этот набор значений истинности индуцируется некоторым принимающим вычислением на входе x, причём стадия проверки этого вычисления выполняется не более чем за $p(n)$ шагов и слово-догадка имеет длину не более $p(n)$. Таким образом мы получим импликации:

$x \in L$	$\Leftrightarrow$	на входе x существует принимающее вычисление программы M
	$\Leftrightarrow$	на входе x существует принимающее вычисление программы M, число шагов которого не превосходит $p(n)$, а слово-догадка w имеет длину, равную $p(n)$
	$\Leftrightarrow$	существует выполняющее задание значений истинности для набора дизъюнкций задачи $f_L(x)$.

Это будет означать, что для f_L выполнено одно из двух условий, требующихся в определении полиномиальной сводимости. Другое условие, заключающееся в том, что f_L должна быть вычислима за полиномиальное время, можно будет легко проверить после того, как описание f_L будет закончено.

Дизъюнкции индивидуальной задачи $f_L(x)$ можно подразделить на 6 групп, каждая из которых будет налагать ограничение определённого типа на любой выполняющий набор значений истинности. Эти группы показаны на рис. 2.8.

Легко видеть, что если все 6 групп дизъюнкций действительно осуществляют поставленные перед ними цели, то выполняющий набор значений истинности обязан соответствовать нужному принимающему вычислению на входе x. Единственное, что остаётся,– это указать способ построения групп дизъюнкций, осуществляющих эти цели.

Группа дизъюнкций	*Налагаемое ограничение*
G_1	В любой момент времени i программа M находится ровно в одном состоянии.
G_2	В любой момент времени i читающая/ пишущая головка просматривает ровно одну ячейку.
G_3	В любой момент времени i каждая ячейка содержит ровно один символ из Г.
G_4	В момент времени 0 вычисление находится в исходной конфигурации стадии проверки при входе x.
G_5	Не позднее чем через $p(n)$ шагов M переходит в состояние q_Y и, следовательно, принимает x.
G_6	Для любого момента времени i, $0 \leqslant i \leqslant p(n)$, конфигурация программы M в момент времени $i+1$ получается из конфигурации в момент времени i одноразовым применением функции перехода δ.

Рис. 2.8. Группы дизъюнкций для $f_L(x)$ и ограничения, накладываемые ими на выполняющий набор истинностных значений.

Группа G_1 состоит из следующих дизъюнкций:

$$\{Q[i,0], Q[i,1], ..., Q[i,r]\}, 0 \leqslant i \leqslant p(n);$$

$$\{\overline{Q[i,j]}, \overline{Q[i,j']}, 0 \leqslant i \leqslant p(n), 0 \leqslant j < j' < r.$$

Первые $p(n)+1$ из этих дизъюнкций могут быть выполнены одновременно в том и только в том случае, если в каждый момент времени i программа M находится по крайней мере в одном состоянии. Остальные $(p(n)+1)(r+$

$1)(r/2)$ дизъюнкций могут быть выполнены одновременно в том и только в том случае, если ни в один момент времени i программа M не находится *более чем в одном* состоянии. Таким образом, G_1 выполняет своё назначение.

Группы G_2 и G_3 строятся аналогично, а группы G_4 и G_5 очень просты, каждая из них включает дизъюнкции, состоящие только из одного литерала. На рис. 2.9 дано полное описание первых пяти групп дизъюнкций. Отметим, что как число дизъюнкций в этих группах, так и максимальное число литералов, входящих в каждую дизъюнкцию, ограничены некоторым полиномом от n (так как r и v – константы, которые определяются по M и, значит, по языку L).

Несколько сложнее выглядит описание последней группы дизъюнкций – G_6, которая гарантирует, что каждая следующая конфигурация машины получается из предыдущей в результате применения одной команды программы M. Эта группа состоит из двух подгрупп дизъюнкций.

Группа дизъюнкций	*Дизъюнкции, входящие в группу*
G_1	$\{Q[i,0], Q[i,1], ..., Q[i,r]\}$, $0 \leqslant i \leqslant p(n)$; $\{\overline{Q[i,j]}, \ \overline{Q[i,j']}\}$, $0 \leqslant i \leqslant p(n)$, $0 \leqslant j < j' < r$
G_2	$\{H[i,-p(n)], H[i,-p(n)+1], ..., H[i,p(n)+1]\}$, $0 \leqslant i \leqslant p(n)$; $\{H[i,j], H[i,j']\}$, $0 \leqslant i \leqslant p(n)$, $-p(n) \leqslant j < j' \leqslant p(n)+1$
G_3	$\{S[i,j,0], S[i,j,1], ..., S[i,j,v]\}$, $0 \leqslant i \leqslant p(n)$, $-p(n) \leqslant j \leqslant p(n)+1$, $\{\overline{S[i,j,k]}, \overline{S[i,j,k']}$, $0 \leqslant i \leqslant p(n)$, $-p(n) \leqslant j \leqslant p(n)+1$, $0 \leqslant k < k' \leqslant v$,
G_4	$\{Q[0,0]\}$, $\{H[0,1]\}$, $\{S[0,0,0]\}$, $\{S[0,1,k_1]\}$, $\{S[0,2,k_2]\}$, ..., $\{S[0,n,k_n]\}$, $\{S[0,n+1,0]\}$, $\{S[0,n+2,0]\}$, ..., $\{S[0,p(n)+1,0]\}$, где $x = s_{k_1} s_{k_1} \dots s_{k_n}$,
G_5	$\{Q[p(n),1]\}$

Рис. 2.9. Первые 5 групп дизъюнкций для $f_L(x)$.

Первая подгруппа гарантирует, что если читающая/ пишущая головка в момент времени i *не* просматривает ячейку с номером j, то символ в ячейке с номером j от момента времени i к моменту времени $i+1$ не изменится. Дизъюнкции этой группы имеют вид

$$\{\overline{S[i,j,l]}, H[i,j], S[i+1,j,l]\}, \quad 0 \leqslant i \leqslant p(n),$$
$$-p(n) \leqslant j \leqslant p(n)+1, \quad 0 \leqslant 1 \leqslant v.$$

Зафиксируем произвольно момент времени i, ячейку с номером j и символ s_l. Если в момент времени i читающая/ пишущая головка не просматривает ячейку с номером j и в этой ячейке в момент времени i записан символ j, а в момент времени $i+1$ его уже там нет, то указанная выше дизъюнкция не будет выполнена (в противном случае она *будет* выполнена). Таким образом, $2(p(n)+1)^2(v+1)$ дизъюнкций этой группы выполняют своё назначение.

Вторая подгруппа дизъюнкций из группы G_6 гарантирует, что *перестройка* одной машинной конфигурации в следующую происходит согласно функции перехода δ программы M. Для каждой четвёрки

(i, j, k, l), $0 \leqslant i < p(n)$, $-p(n) \leqslant j \leqslant p(n)+1$, $0 \leqslant k \leqslant r$ и $0 \leqslant l \leqslant v$,

эта подгруппа содержит следующие три дизъюнкции:

$\{\overline{H[i,j]},\ \overline{Q[i,k]},\ \overline{S[i,j,l]},\ H[i+1, j+\Delta]\}$,

$\{\overline{H[i,j]},\ \overline{Q[i,k]},\ \overline{S[i,j,l]},\ Q[i+1, k']\}$,

$\{\overline{H[i,j]},\ \overline{Q[i,k]},\ \overline{S[i,j,l]},\ S[i+1, j, l']\}$,

где значения Δ, k' и l' определены так:

если $q_k \in Q \setminus \{q_Y, q_N\}$, то $\delta(q_k, s_l) = (q'_k, s'_l, \Delta)$,

а если $q_k \in \{q_Y, q_N\}$, то $\Delta = 0$, $k' = k$ и $l' = l$.

Нетрудно понять (хотя для этого может потребоваться несколько минут раздумий), что эти $6(p(n))(p(n)+1)(r+1)(v+1)$ дизъюнкций дают необходимое ограничение на выполняющий набор значений истинности.

Таким образом, мы показали, как построить группы дизъюнкций G_1 – G_6, которые выполняют своё назначение. Если $x \in L$, то у программы M на входе x есть принимающее вычисление длины не более $p(n)$, и это вычисление даёт при заданной интерпретации переменных набор значений истинности, который выполнит все дизъюнкции из $C = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_6$. Наоборот, набор дизъюнкций C построен так, что любой выполняющий набор значений истинности для C обязан соответствовать некоторому принимающему вычислению программы M на входе x. Отсюда следует, что для $f_L(x)$ имеется выполняющий набор значений истинности тогда и только тогда, когда $x \in L$.

Теперь осталось показать, что для любого фиксированного языка L индивидуальная задача $f_L(x)$ может быть построена за время, ограниченное полиномом от $n = |x|$. Если язык L задан, то можно выбрать некоторую НДМТ-программу M, которая распознает L за время, ограниченное полиномом p (нет необходимости за полиномиальное время находить саму недетерминированную программу, так как необходимо лишь показать, что отображение f_L *существует*. Если имеются определённая НДМТ-программа M и многочлен p, то построение множества переменных U и набора дизъюнкций C требует чуть больше работы, чем заполнение всех позиций в стандартной (хотя и довольно сложной) формуле. Полиномиальная ограниченность этого вычисления будет ясна, если мы покажем, что $Length[f_L(x)]$ ограничена сверху полиномом от n, где $Length[I]$, как указано в разд. 2.1, характеризует длину слова, кодирующего индивидуальную задачу I, при некоторой разумной схеме кодирования.

В качестве «разумной» функции длины для задачи ВЫП можно, например, взять $|U| \cdot |C|$. Ни одна из дизъюнкций не может содержать более $2 \cdot |U|$ литералов (т.к. общее число литералов равно $2 \cdot |U|$), а число символов, необходимое для описания каждого индивидуального литерала, не более $\log |U|$, но этой величиной можно пренебречь, так как она даёт полиномиальный вклад в общую длину задачи. Поскольку r и v фиксированы заранее, то они могут увеличить $|U|$ и $|C|$ в постоянное число раз, поэтому имеем $|U| = O(p(n)^2)$ и $|C| = O(P(n))^2$. Следовательно, $Length[f_L(x)] = |U|\cdot|C| = O(p(n)^4)$ и, значит, ограничена полиномом от n, что и требовалось.

Таким образом, преобразование f_L может быть вычислено алгоритмом, имеющим полиномиальную временну́ю сложность (однако конкретная полиномиальная граница для сложности будет зависеть от L, а также от выбора M и p), из чего можно заключить, что для всех $L \in NP$ функция f_L вычислима за полиномиальное время и отображает L в ВЫП (точнее, отображает L в $L_{\text{ВЫП}}$). Отсюда вытекает утверждение теоремы, что ВЫП есть *NP*-полная задача.

■

Глава 3.

Доказательство результатов об *NP*-полноте

Если бы все доказательства *NP*-полноты были так же сложны, как доказательство *NP*-полноты задачи ВЫПОЛНИМОСТЬ, то очень сомнительно, что список *NP*-полных задач смог бы стать столь обширным, как в настоящее время. Однако, как указывалось в разд. 2.4, если уже известна одна *NP*-полная задача, то процедура доказательства *NP*-полноты других задач значительно упрощается. Для доказательства *NP*-полноты задачи $\Pi \in NP$ достаточно показать, что какая-нибудь из известных *NP*-полных задач Π' может быть сведена к Π. Таким образом, в дальнейшем процесс доказательства *NP*-полноты задачи распознавания Π будет состоять из следующих четырёх шагов:

1. доказательства того, что Π лежит в *NP*;
2. выбора известной *NP*-полной задачи Π';
3. построения функции f, сводящей задачу Π' к задаче Π, и
4. доказательства того, что функция f осуществляет полиномиальное сведение.

Цель настоящей главы состоит не только в том, чтобы ознакомить читателей с конечными результатами этого процесса (с окончательными доказательствами *NP*-полноты), но также и в том, чтобы подготовить их к самостоятельному поиску таких доказательств.

В разд. 3.1 приводится 6 задач, которыми чаще других пользуются при доказательстве результатов об *NP*-полноте в качестве «известных *NP*-полных задач», и доказывается *NP*-полнота этих шести задач. В разд. 3.2 приводится описание трёх общих подходов к установлению сводимости задач и даётся их иллюстрация посредством доказательства *NP*-полноты большого числа различных задач. Заключительный раздел содержит несколько результатов об *NP*-полноте, которые предлагается доказать читателю.

3.1. Шесть основных NP-полных задач

Когда опытным практикам встречается задача П, NP-полноту которой требуется доказать, их преимущество заключается в том, что они имеют богатый выбор путей такого доказательства. Вполне может случиться, что в прошлом они уже доказывали или видели доказательство NP-полноты похожей задачи Π'. Это может привести к мысли попытаться построить доказательство NP-полноты задачи П, копируя доказательство NP-полноты задачи Π', или просто сводя задачу Π' к задаче П. Во многих случаях это может привести к достаточно простому доказательству NP-полноты задачи П.

Однако очень часто не удаётся найти NP-полную задачу, похожую на П (даже располагая обширным списком задач, подобным тому, который приведён в конце книги). В таких случаях практику может быть не ясно, какая из сотен известных NP-полных задач лучше всего подходит в качестве основы искомого доказательства. Однако предыдущий опыт может оказаться полезным для ограничения области выбора некоторым «ядром» из базовых задач, использовавшихся ранее. Хотя теоретически *любую* из известных NP-полных можно наравне с другими выбрать для доказательства NP-полноты новой задачи, на практике оказывается, что некоторые задачи подходят для этой цели гораздо лучше других. Следующие шесть задач входят в число тех, которые используются наиболее часто и для начинающего они могут служить «основным ядром» списка известных NP-полных задач.

3-ВЫПОЛНИМОСТЬ (3-ВЫП)

УСЛОВИЕ. Дан набор $C = \{c_1, c_2, ..., c_m\}$ дизъюнкций на конечном множестве переменных U, таких, что $|c_i| = 3$, $1 \leqslant i \leqslant m$.

ВОПРОС. Существует ли на U набор значений истинности, при котором выполняются все дизъюнкции из C?

ТРЁХМЕРНОЕ СОЧЕТАНИЕ (3-С)

УСЛОВИЕ. Дано множество $M \subseteq W \times X \times Y$, где W, X и Y – непересекающиеся множества, содержащие одинаковое число элементов q.

ВОПРОС. Верно ли, что M содержит трёхмерное *сочетание*, т.е. подмножество $M' \subseteq M$, такое, что $|M'| = q$ и никакие два разных элемента M' не имеют ни одной равной координаты?

ВЕРШИННОЕ ПОКРЫТИЕ (ВП)

УСЛОВИЕ. Дан граф $G = (V, E)$ и положительное целое число K, $K \leqslant |V|$.

ВОПРОС. Имеется ли в графе G *вершинное покрытие* не более чем из K элементов, т.е. такое подмножество $V' \subseteq V$, что $|V'| \leqslant K$ и для каждого ребра $\{u, v\} \in E$ хотя бы одна из вершин u или v принадлежит V'?

КЛИКА

УСЛОВИЕ. Дан граф $G = (V, E)$ и положительное целое число J, $J \leqslant |V|$.

ВОПРОС. Верно ли, что G содержит некоторую *клику* мощности не менее J, т.е. такое подмножество $V' \subseteq V$, что $|V'| \geqslant J$ и любые две вершины из V' соединены ребром из E?

ГАМИЛЬТОНОВ ЦИКЛ (ГЦ)

УСЛОВИЕ. Дан граф $G = (V, E)$.

ВОПРОС. Верно ли, что G содержит гамильтонов цикл, т.е. такую последовательность $(v_1, v_2, ..., v_n)$ вершин графа G, , что $n = |V|$, $\{v_n, v_1\} \in E$ и $\{v_i, v_{i+1}\} \in E$ для всех i, $1 \leqslant i \leqslant n$.

РАЗБИЕНИЕ

УСЛОВИЕ. Заданы конечное множество A и «вес» $s(a) \in Z^+$ для каждого $a \in A$.

ВОПРОС. Существует ли подмножество $A' \subseteq A$, такое, что

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

Одна из причин популярности этих шести задач заключается в том, что все они содержались в исходном списке из 21 *NP*-полной задачи, приведённом в работе *Карпа* [280]. Мы начнём демонстрацию методов доказательства *NP*-полноты с доказательства *NP*-полноты этих шести задач, указывая в подходящих случаях такие их варианты, *NP*-полнота которых более или менее непосредственно следует из *NP*-полноты этих шести задач.

Первая сводимость будет получена с помощью задачи ВЫПОЛНИМОСТЬ, поскольку пока это единственная задача, *NP*-полноту которой мы установили. Однако по мере доказательства *NP*-полноты этих шести задач список *NP*-полных задач будет расширяться и доказательство *NP*-полноты некоторой задачи может быть получено с использованием любой из задач, *NP*-полнота которой установлена раньше, чем для задачи П. На рис. 3.1 указана последовательность сведения задач, которая применяется в доказательстве *NP*-полноты шести основных задач, причём если стрелка ведёт от одной задачи к другой, то первая сводится ко второй.

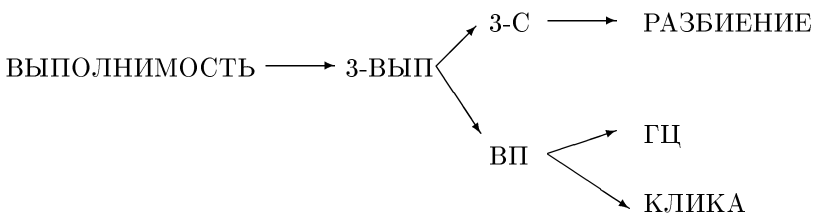


Рис. 3..1: *Диаграмма последовательности сведения задач, используемых для доказательства NP-полноты шести основных задач.*

Эта последовательность не совпадает с последовательностью, выбранной Карпом, но даже в тех случаях, когда они совпадают, мы иногда изменяем или модифицируем доказательство Карпа, чтобы лучше проиллюстрировать общие приёмы сводимости.

3.1.1. 3-выполнимость

Задача 3-ВЫПОЛНИМОСТЬ есть просто ограниченный вариант задачи ВЫПОЛНИМОСТЬ, в которой каждая индивидуальная задача имеет ровно три литерала в каждой дизъюнкции. Из-за своей простой структуры эта задача наиболее часто применяется для доказательства результатов об *NP*-полноте.

Теорема 3.1. *Задача 3-ВЫПОЛНИМОСТЬ является NP-полной.*

Доказательство 1.

Нетрудно видеть, что 3-ВЫП $\in NP$. Это следует из того, что недетерминированному алгоритму необходимо угадать лишь набор значений истинности переменных задачи и проверить за полиномиальное время, будут ли при таком наборе значений истинности выполняться все заданные трёхлитеральные дизъюнкции.

Сведём задачу ВЫП к задаче 3-ВЫП. Пусть $U = \{u_1, u_2, ..., u_n\}$ – множество переменных и $C = \{c_1, c_2, ..., c_m\}$ – произвольный набор дизъюнкций, определяющий произвольную индивидуальную задачу из ВЫП. Построим набор C' трёхлитеральных дизъюнкций на некотором множестве переменных U', такой, что C' выполним тогда и только тогда, когда выполним C.

Набор C' будет строиться путём замены каждой отдельной дизъюнкции $c_j \in C$ «эквивалентным» набором C'_j трёхлитеральных дизъюнкций на множестве U исходных переменных и множестве U'_j некоторых дополнительных переменных, причём переменные из U'_j будут использоваться только в дизъюнкциях из C'_j. Другими словами,

$$U = U \cup (\bigcup_{j=1}^{m} U'_j)$$

и

$$C' = \bigcup_{j=1}^{m} C'_j.$$

Таким образом, нужно только показать, как исходя из c_j можно построить C'_j и U'_j.

Пусть c_j задаётся множеством $\{z_1, z_2, ..., z_k\}$, где z_i – литералы на множестве U. Способ образования C'_j и U'_j зависит от значения k.

Случай 1. $k = 1$. Тогда $U'_j = \{y_j^1, y_j^2\}$,

$C'_j = \{\{z_1, y_j^1, y_j^2\}, \{z_1, y_j^1, y_j^{-2}\}, \{z_1, y_j^{-1}, y_j^2\}, \{z_1, y_j^{-1}, y_j^{-2}\}\}$.

Случай 2. $k = 2$. Тогда $U'_j = \{y_j^1\}$,

$C'_j = \{\{z_1, z_2, y_j^1\}, \{z_1, z_2, y_j^{-1}\}\}$,

Случай 3. $k = 3$. Тогда $U'_j = \varnothing$, $C'_j = \{\{c_j\}\}$.

Случай 4. $k > 3$. Тогда $U'_j = \{y_j^i : 1 \leqslant i \leqslant k-3\}$,

$C'_j = \{\{z_1, z_2, y_j^1\}\} \cup \{\{y_j^{-i}, z_{i+2}, y_j^{i+1}\} : 1 \leqslant i \leqslant k-4\} \cup \{y_j^{-k-3}, z_{k-1}, z_k\}\}$.

Для доказательства того, что здесь в самом деле имеет место сводимость, необходимо показать, что набор дизъюнкций C' выполним тогда и только тогда, когда выполним набор дизъюнкций C. Предположим вначале, что $t : U \to \{T, F\}$ есть набор значений истинности, выполняющий C. Покажем, что t может быть продолжен до набора значений истинности $t' : U' \to \{T, F\}$, который выполняет C'. Поскольку переменные в множестве $U' \setminus U$ делятся на группы U'_j и так как переменные в каждой группе U'_j входят только в дизъюнкции, принадлежащие C'_j, то достаточно показать, как t может быть продолжен на каждое множество U'_j отдельно, и в каждом случае нужно проверить, что выполняются все дизъюнкции в соответствующем множестве C'_j.

Это можно сделать следующим образом. Если U'_j построены, как в случае 1 или 2, то дизъюнкции из C'_j уже выполнены с помощью t, поэтому t может быть продолжен на U'_j произвольно, например, если положить $t'(y) = T$ для всех $y \in U'_j$. Если U'_j построено, как в случае 3, то U'_j – пусто и единственная дизъюнкция в C'_j уже выполнена с помощью t. Остаётся только случай 4, который соответствует дизъюнкции $\{z_1, z_2, ..., z_k\}$ из C, причём $k > 3$. Поскольку t – выполняющий набор значений истинности для C, то найдётся такое целое l, что z_l при t принимает значение *истина*. Если $l = 1$ или 2, то полагаем $t'(y_j^i) = F$, $1 \leqslant i \leqslant k-3$. Если $l = k-1$ или k, то полагаем $t'(y_j^i) = F$, $1 \leqslant i \leqslant k-3$. В противном случае полагаем $t'(y_j^i) = T$, $1 \leqslant i \leqslant l-2$ и $t'(y_j^i) = F$ при $l-1 \leqslant i \leqslant k-3$. Легко проверить, что такой набор значений истинности обеспечит выполнение всех дизъюнкций в C'_j, поэтому t' выполняет все дизъюнкции из C'. Обратно, если t' – некоторый выполняющий набор значений истинности для C', то легко проверить, что ограничение t' на переменные из U должно быть выполняющим набором значений истинности для C. Таким образом, C' выполним тогда и только тогда, когда выполним C.

Для того чтобы убедиться, что эта сводимость полиномиальная, достаточно заметить, что число трёхлитеральных дизъюнкций в C' ограничено полиномом от mn. Следовательно, размер индивидуальной задачи 3-ВЫП ограничен сверху некоторым полиномом от размера соответствующей задачи ВЫП, а так как все детали построения сводимости очевидны, то читателю будет нетрудно проверить, что эта сводимость является полиномиальной. ■

Ограниченная структура задачи 3-ВЫП делает её гораздо более полезной при доказательстве результатов об *NP*-полноте по сравнению с задачей ВЫП. Любое доказательство, основанное на задаче ВЫП (кроме только что приведённого), может быть легко перестроено в доказательство, основанное на 3-ВЫП, даже без изменения функции, осуществляющей сводимость. На самом деле дополнительное условие о том, что все дизъюнкции имеют одинаковую длину, часто может упростить сводимость, которую нужно построить, и тем самым облегчить её отыскание. Более того, очень малая длина дизъюнкций позволяет использовать сводимости, которые вообще не могли бы быть построены для индивидуальных задач с дизъюнкциями большей длины. Это наводит на мысль, что было бы ещё лучше, если бы удалось показать, что аналогичная задач 2-ВЫПОЛНИМОСТЬ (каждая дизъюнкция содержит ровно два литерала) является *NP*-полной. Однако 2-ВЫП может быть решена методом «резолюций» за время, ограниченное полиномом от произведения числа дизъюнкций и числа переменных заданной индивидуальной задачи [91] (см. также [114]), и, следовательно, принадлежит классу P.

3.1.2. Трёхмерное сочетание

Задача ТРЁХМЕРНОЕ СОЧЕТАНИЕ (3-С) обобщает следующую известную задачу о «бракосочетании»: имеется n холостых мужчин и n незамужних женщин, а также список всех пар мужчин и женщин, согласных вступить в брак друг с другом; можно ли осуществить n бракосочетаний так, чтобы каждый получил бы приемлемого супруга (или супругу), причём никто не должен вступать в брак дважды? Аналогичным образом в задаче ТРЁХМЕРНОЕ СОЧЕТАНИЕ множества W, X и Y соответствуют *трём* различным «полам», а каждая тройка из M отвечает «трёхмерному сочетанию», приемлемому для всех трёх участников. Приверженцы традиций будут довольны, узнав, что, в то время как задача 3-С является *NP*-полной, обычная задача о бракосочетании может быть решена за полиномиальное время (см., например, [212]).

Теорема 3.2. *Задача ТРЁХМЕРНОЕ СОЧЕТАНИЕ является NP-полной.*

Доказательство 2. Легко видеть, что 3-С $\in$ *NP*. Это следует из того, что недетерминированному алгоритму нужно только угадать подмножество в M, состоящее из $q = |W| = |X| = |Y|$ троек, и за полиномиальное время проверить, что любые две из угаданных троек отличаются по всем координатам.

Сведём задачу 3-ВЫП к 3-С. Пусть $U = \{u_1, u_2, ..., u_n\}$ – множество переменных и $C = \{c_1, c_2, ..., c_m\}$ – множество дизъюнкций произвольной индивидуальной задачи из 3-ВЫП. Нужно построить непересекающиеся множества W, X, Y и $M \subseteq W \times X \times Y$, такие, что $|W| = |X| = |Y|$ и M содержит паросочетание в том и только в том случае, когда C выполнима.

Искомое множество упорядоченных троек M будет разбито на три различных класса в соответствии с их функциональным назначением: «набор значений истинности» и «развёртка», «проверка выполнения» и «достройка».

Тройки, входящие в класс «набор значений истинности и развёртка», разбиваются на группы (компоненты), каждая из которых соответствует одной переменной $u \in U$, а её строение зависит от общего числа m дизъюнкций в C. Строение этой компоненты для случая $m = 4$ показано на рис. 3.2.

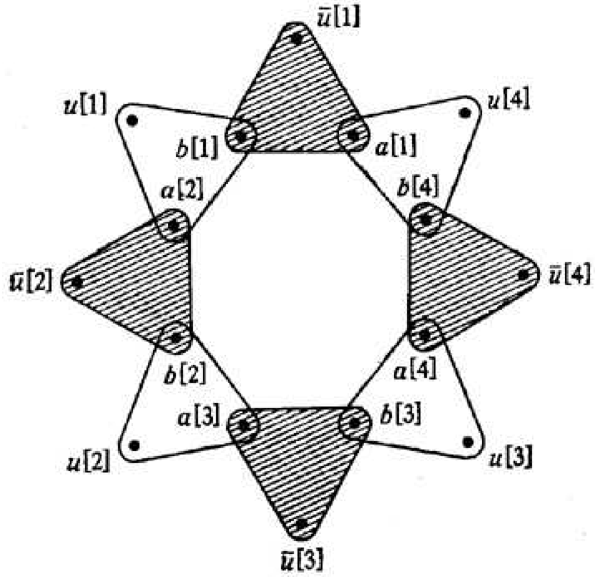


Рис. 3..2: *Компонента T_i, задающая значения истинности при $m = 4$ (нижние индексы у переменных для простоты опущены). Должно быть выбрано либо множество T_i^t (заштрихованное множество), либо множество T_i^f (незаштрихованное множество), при этом остаются НЕПОКРЫТЫМИ (т.е. не объединёнными в тройки) либо все $u_i[j]$, либо все $\bar{u}_i[j]$ соответственно.*

В общем случае каждая такая компонента, соответствующая переменной u_i, включает «внутренние элементы» $a_i[j] \in X$ и $b_i[j] \in Y$, $1 \leqslant j \leqslant m$, которые не будут входить ни в какую тройку вне этой компоненты, и «внешние» элементы $u_i[j], \bar{u}_i[j] \in W$, $1 \leqslant j \leqslant m$ которые *будут* входить в другие тройки. Образующие эту компоненту тройки могут быть подразделены на два множества:

$T_i^t = \{(\bar{u}_i[j], a_i[j], b_i[j]) : 1 \leqslant j \leqslant m\};$

$T_i^f = \{(u_i[j], a_i[j+1], b_i[j]) : 1 \leqslant j \leqslant m\} \cup \{(u_i[m], a_i[l], b_i[m])\}$.

Так как внутренние элементы $\{a_i[j], b_i[j] : 1 \leqslant j \leqslant m\}$ могут входить только в тройки, принадлежащие множеству $T_i = T_i^t \cup T_i^f$, то легко видеть, что любое трёхмерное сочетание M' должно иметь ровно m троек из T_i, а именно либо все тройки из T_i^t, либо все тройки из T_i^f. Следовательно, можно считать, что компонента из T_i приводит к тому, что трёхмерное сочетание индуцирует присвоение переменной u_i значения «истина» или «ложь». Таким образом, трёхмерное сочетание $M' \subseteq M$ определяет набор значений истинности на множестве U, при этом переменная u_i принимает значение «истина» тогда и только тогда, когда $M' \cap T_i = T_i^t$.

Каждая компонента множества M, предназначенная для проверки выполнимости, соответствует одной дизъюнкции $c_j \in C$. Такая компонента содержит только два «внутренних» элемента $s_1[j] \in X$ и $s_2[j] \in Y$, а также «внешние» элементы из множества $\{u_i[j], \bar{u}_i[j] : 1 \leqslant i \leqslant n\}$, выбираемые в зависимости от того, какие литералы содержатся в дизъюнкции c_j. Множество троек, образующих эту компоненту, определяются следующим образом:

$$C_j = \{(u_i[j], s_1[j], s_2[j]) : \bar{u}_i \in c_j\} \cup \{(\bar{u}_i[j], s_1[j], s_2[j]) : \bar{u}_i \in c_j\}.$$

Итак, любое трёхмерное сочетание $M' \subseteq M$ должно содержать ровно одну тройку из C_j. Но это условие может быть выполнено только в том случае, если для некоторого литерала $u_i \in c_j$ (или $\bar{u}_i \in c_j$) элемент $u_i[j]$ (соответственно $\bar{u}_i[j]$) не войдёт ни в одну из троек множества $T_i \cap M'$, а это будет иметь место тогда и только тогда, когда набор значений истинности, определяемый множеством M', выполняет дизъюнкцию c_j.

Построение нужной индивидуальной задачи из 3-С будет закончено после описания одной большой компоненты «достройки» G. . Эта компонента включает внешние элементы $g_1[k] \in X$, $g_2[k] \in Y$, $1 \leqslant k \leqslant m(n-1)$, а также внешние элементы вида $u_i[j]$ (или $\bar{u}_i[j]$ из множества W). Компонента G состоит из следующего множества троек:

$G = \{(u_i[j], g_1[k], g_2[k]), (\bar{u}_i[j], g_1[k], g_2[k]) :$

$1 \leqslant k \leqslant m(n-1), 1 \leqslant j \leqslant n, 1 \leqslant j \leqslant m$

Таким образом, каждая пара $g_1[k]$, $g_2[k]$ должна войти в трёхмерное сочетание только с одним из элементов $u_i[j]$ или $\bar{u}_i[j]$, не попавших в тройки из $M' \setminus G$. Имеется ровно $m(n-1)$ таких «непокрытых» внешних элементов, и структура компоненты G обеспечивает, что они всегда могут быть покрыты с помощью выбора соответствующего подмножества $M' \cap G$. Следовательно, компонента G гарантирует, что, если для некоторого подмножества из $M \setminus G$ выполняются все ограничения, налагаемые компонентами набора значений истинности, тогда это подмножество может быть дополнено до трёхмерного сочетания в M.

Резюмируя сказанное, положим:

$W = \{u_i[j], \bar{u}_i[j] : 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$,

$X = A \cup S_1 \cup G_1$

где
$A = \{a_i[j] : 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$,
$S_1 = \{s_1[j] : 1 \leqslant j \leqslant m\}$,
$G_1 = \{g_1[j] : 1 \leqslant j \leqslant m(n-1)\}$;
$Y = B \cup S_2 \cup G_2$,
где
$B = \{b_i[j] : 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$,
$S_2 = \{s_2[j] : 1 \leqslant j \leqslant m\}$,
$G_2 = \{g_2[j] : 1 \leqslant j \leqslant m(n-1)\}$;
и
$M = \left(\bigcup_{i=1}^{n} T_i\right) \cup \left(\bigcup_{j=1}^{m} C_j\right) \cup G.$

Заметим, что каждая тройка из M есть элемент множества $W \times X \times Y$, что и требовалось. Кроме того, поскольку множество M содержит только

$$2mn + 3m + 2m^2n(n-1)$$

троек и множество M в терминах индивидуальной задачи из 3-ВЫП определяется явным образом, легко видеть, что M может быть построено за полиномиальное время. Из комментариев, которыми сопровождалось описание множества M, немедленно следует, что оно не может содержать трёхмерного сочетания, если набор C невыполним. Теперь необходимо доказать, что существование выполняющего набора значений истинности для набора дизъюнкций C влечёт за собой существование в M трёхмерного сочетания.

Пусть $t : U \to \{T, F\}$ – произвольный выполняющий набор значений истинности для C. Построим трёхмерное сочетание $M' \subseteq M$ следующим образом. Для каждой дизъюнкции $c_j \in C$ рассмотрим некоторый литерал $z_j \in \{u_i, \bar{u}_i | 1 \leqslant i \leqslant n\} \cap c_j$, принимающий значение «истина» при отображении t (такое z_j должно существовать, поскольку t выполняет c_j). Затем положим

$$M' = \left(\bigcup_{t(u_i)=T} T_i^t\right) \cup \left(\bigcup_{t(u_i)=F} T_i^f\right) \cup \left(\bigcup_{j=1}^{m} \{(z_j[j], s_1[j], s_2[j])\}\right) \cup G',$$

где G' – соответствующее подмножество множества G, выбранное так, что оно содержит все $g_1[k]$, $g_2[k]$ и оставшиеся $u_i[j]$ и $\bar{u}_i[j]$. Нетрудно проверить, что такое множество G' всегда можно выбрать и что полученное в результате множество M' будет трёхмерным сочетанием. ■

Вместо задачи 3-С для доказательства результатов об NP-полноте часто пользуются следующей, более общей и более наглядной версией этой задачи.

ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ (ТП-3)

УСЛОВИЕ. Задано конечное множество X, такое, что $|X| = 3q$, и семейство C трёхэлементных подмножеств множества X.

ВОПРОС. Верно ли, что семейство C содержит *точное покрытие* множества X, т.е. такое подсемейство $C' \subseteq C$, что каждый элемент из X содержится ровно в одном элементе из C?

Заметим, что любую индивидуальную задачу из 3-С можно рассматривать как индивидуальную задачу из ТП-3, считая просто, что M состоит из неупорядоченных троек элементов множества $W \cup X \cup Y$. При этом трёхмерные сочетания для индивидуальной задачи из 3-С взаимно однозначно соответствуют точным покрытиям соответствующей индивидуальной задачи из ТП-3. Таким образом, задача 3-С есть ограниченная версия задачи ТП-3, и *NP*-полнота ТП-3 следует из естественного сведения к ней задачи 3-С.

3.1.3. Вершинное покрытие и Клика

Несмотря на то что задачи ВЕРШИННОЕ ПОКРЫТИЕ и КЛИКА при доказательстве результатов об *NP*-полноте используются независимо, в действительности они представляют собой всего лишь разные способы рассмотрения одной и той же задачи. Для того чтобы это понять, удобно рассмотреть их вместе с третьей задачей, которая называется НЕЗАВИСИМОЕ МНОЖЕСТВО.

Независимым множеством в графе $G = (V, E)$ называется такое подмножество V' из V, что для любых $u, v \in V'$, ребро $\{u, v\}$ не принадлежит E. В задаче НЕЗАВИСИМОЕ МНОЖЕСТВО для заданного графа $G = (V, E)$ и положительного целого числа $J \leqslant |V|$ спрашивается, верно ли, что G содержит независимое множество V', такое, что $|V'| \geqslant J$. Легко убедиться, что независимые множества, клики и вершинные покрытия связаны между собой следующим образом.

Лемма 3.1. *Для любого графа $G = (V, E)$ и подмножества $V' \subseteq V$ следующие утверждения эквивалентны:*
(а) V' есть вершинное покрытие G;
(б) $V \setminus V'$ есть независимое множество в G;
(в) $V \setminus V'$ есть клика в дополнении $\bar{G}$ графа G, где $\bar{G} = (V, \bar{E})$, $\bar{E} = \{(u, v) : u, v \in V$ и $\{u, v\} \notin E\}$.

Таким образом, эти три задачи можно рассматривать в некотором смысле как «различные версии» одной и той же задачи. Более того, взаимосвязи между задачами, указанные в лемме 3.1, делают тривиальным вопрос о сводимости между любыми двумя из них.

Сведём, например, задачу ВЕРШИННОЕ ПОКРЫТИЕ к задаче КЛИКА. Пусть $G = (V, E)$ и $K \leqslant |V|$ – произвольная индивидуальная задача из ВП. Соответствующая индивидуальная задача из КЛИКИ получается, если взять граф $\bar{G}$ и целое число $J = |V| - K$.

Отсюда следует, что *NP*-полнота все трёх задач следует из *NP*-полноты любой из них. Докажем, что задача ВЕРШИННОЕ ПОКРЫТИЕ *NP*-полна.

Теорема 3.3. *Задача ВЕРШИННОЕ ПОКРЫТИЕ NP-полна.*

Доказательство 3. Легко видеть, что задача ВП $\in$ *NP*. Это следует из того, что недетерминированному алгоритму достаточно угадать подмножество вершин и за полиномиальное время проверить, что это подмножество содержит хотя бы один конец любого ребра и имеет соответствующее число элементов.

Сведём задачу 3-ВЫП к задаче ВП. Пусть $U = \{u_1, u_2, ..., u_n\}$и $C = \{c_1, c_2, ..., c_m\}$ определяют произвольную индивидуальную задачу из 3-ВЫП. Укажем граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$, такие, что G имеет вершинное покрытие с числом элементов не более K тогда и только тогда, когда выполн$\grave{\text{и}}$м набор дизъюнкций C.

Как и в предыдущем доказательстве, наша конструкция будет составлена из нескольких компонент. В данном случае, однако, будут присутствовать только компоненты набора значений истинности и проверки выполнимости, дополненные некоторыми вспомогательными рёбрами, связывающими различные компоненты.

Для каждой переменной $u_i \in U$ имеется компонента $T_i = (V_i, E_i)$ набора значений истинности (где $V_i = \{u_i, \bar{u}_i\}$, $E_i = \{\{u_i, \bar{u}_i\}\}$), т.е. T_i – это пара вершин, соединённых одним ребром. Заметим, что любое вершинное покрытие должно покрыть ребро из E_i, поэтому оно должно содержать по крайней мере одну из вершин u_i или $\bar{u}_i$.

Для каждой дизъюнкции $c_j \in C$ имеется компонента проверки выполнимости $S_j = (V'_j, E'_j)$, состоящая из трёх вершин и трёх соединяющих их рёбер, образующих треугольник:

$V'_j = \{a_1[j], a_2[j], a_3[j]\}$,

$E'_j = \{\{a_1[j], a_2[j], a_1[j], a_3[j], a_2[j], a_3[j]\}\}$.

Отметим, что любое вершинное покрытие должно содержать хотя бы две вершины из V'_j, чтобы покрыть все три ребра из E'_j.

Связывающие рёбра – это единственная часть конструкции, зависящая от того, какие литералы входят в дизъюнкции. Их лучше всего рассматривать с точки зрения компонент проверки выполнимости. Для каждой дизъюнкции $c_j \in C$ обозначим через x_j, y_j, z_j три входящих в неё литерала. Тогда связывающие рёбра, исходящие из S_j, задаются следующим образом:

$$E''_j = \{\{a_1[j], x_j\}, \{a_2[j], y_j\}, \{a_3[j], z_j\}\}.$$

Построение индивидуальное задачи из ВП будет закончено, если положить $k = n + 2m$ и $G = (V, E)$, где

$V = \left(\bigcup_{i=1}^{n} V_i\right) \cup \left(\bigcup_{j=1}^{m} V'_i\right)$;

$E = \left(\bigcup_{i=1}^{n} E_i\right) \cup \left(\bigcup_{j=1}^{m} E'_i\right) \cup \left(\bigcup_{j=1}^{m} E''_i\right)$.

На рис. 3.3 приведён граф, соответствующий индивидуальной задаче из 3-ВЫП, когда $U = \{u_1, u_2, u_3, u_4\}$ и $C = \{\{u_1, \bar{u}_3, \bar{u}_4\}, \{\bar{u}_1, u_2, \bar{u}_4\}\}$.

Нетрудно видеть, что эту конструкцию можно осуществить за полиномиальное время. Поэтому достаточно показать, что C выполнима тогда и только тогда, когда у G есть вершинное покрытие с числом элементов не более K .

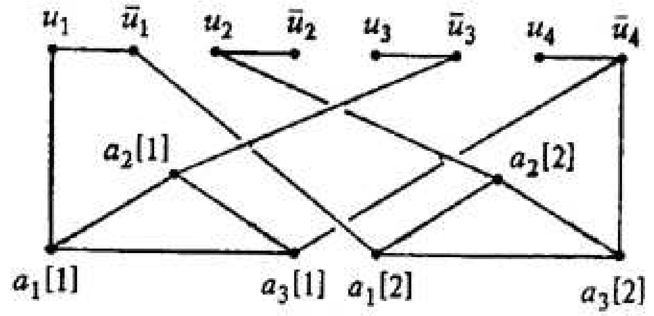


Рис. 3..3: *Индивидуальная задача из ВЕРШИННОЕ ПОКРЫТИЕ, к которой приводит индивидуальная задача из 3-ВЫП при* $U = \{u_1, u_2, u_3, u_4\}$, $C = \{\{u_1, \bar{u}_3, \bar{u}_4\}, \{\bar{u}_1, u_2, \bar{u}_4\}\}$.

Предположим вначале, что $V' \subseteq V$ – вершинное покрытие G и $|V'| \leqslant K$. В силу сделанных выше замечаний, V' должно содержать по крайней мере одну вершину каждого T_i и хотя бы две вершины каждого S_j. В совокупности это даёт по крайней мере $n + 2m = K$ вершин, поэтому V' должно содержать *ровно* одну вершину каждого T_i и *ровно* две вершины каждого S_j. Таким образом, можно использовать пересечение V' с компонентами наборов значений истинности, чтобы получить отображение $t : U \to \{T, F\}$. Для этого положим $t(u_i) = T$, если $u_i \in V'$, и $t(u_i) = F$, если $\bar{u}_i \in V'$. Для доказательства того, что этот набор значений истинности выполняет каждую дизъюнкцию $c_j \in C$, рассмотрим три ребра подграфа E''_j. Только два из них могут быть инцидентны вершинам из $V'_j \cap V'$, поэтому одно из них должно быть инцидентно вершине некоторого подграфа V_i, принадлежащей также nV'. Отсюда следует, что соответствующий литерал u_i (или $\bar{u}_i$) дизъюнкции c_j принимает значение «истина» при рассматриваемом наборе значений истинности t, и, следовательно, t выполняет дизъюнкцию c_j. Поскольку это утверждение имеет место для каждой дизъюнкции $c_j \in C$, то отсюда следует, что t – выполняющий набор значений истинности для C.

Обратно, предположим, что $t : U \to \{T, F\}$- выполняющий набор значений истинности для C. Тогда соответствующее вершинное покрытие V' содержит одну вершину из каждого T_i и две вершины из каждого S_j. При этом вершиной из T_i, принадлежащей V', будет u_i (если $t(u_i) = T$) или $\bar{u}_i$ (если $t(u_i) = F$). Поскольку t выполняет каждую дизъюнкцию c_j, то это обеспечивает покрытие по крайней мере одного из трёх рёбер, принадлежащих множеству E''_j. Следовательно, осталось включить в V' концы оставшихся двух рёбер из E''_j, принадлежащие подграфу S_j (которые могут быть либо инцидентны, либо не инцидентны вершинам из компонент наборов значений истинности), при этом получится искомое вершинное покрытие. ■

3.1.4. Гамильтонов цикл

В гл. 2 мы видели, что к задаче ГАМИЛЬТОНОВ ЦИКЛ (ГЦ) может быть сведена задача распознавания КОММИВОЯЖЕР, поэтому *NP*-полнота последней задачи следует из *NP*-полноты задачи ГЦ. После завершения доказательства *NP*-полноты задачи ГЦ будет указано несколько вариантов задачи ГЦ, *NP*-полнота которых более или менее непосредственно следует из аналогичного факта для задачи ГЦ.

В дальнейшем для удобства будем пользоваться следующим соглашением: если $(v_1, v_2, ..., v_n)$ – некоторый гамильтонов цикл, то рёбра $\{v_i, v_{i+1}\}$, $1 \leqslant i \leqslant n$, и $\{v_n, v_1\}$ будут называться рёбрами в этом цикле. Приводимая ниже сводимость получена комбинацией двух сводимостей из работы Карпа [280], она указана в работе Лиу и Гелдмахера [358].

Теорема 3.4. *Задача ГАМИЛЬТОНОВ ЦИКЛ является NP-полной.*

Доказательство 4. Легко видеть, что ГЦ $\in$ *NP*. Это вытекает из того, что недетерминированному алгоритму нужно угадать только последовательность вершин и за полиномиальное время проверить, что все вершины этой последовательности соединены рёбрами, принадлежащими данному графу.

Сведём задачу ВЕРШИННОЕ ПОКРЫТИЕ к задаче ГЦ. Пусть индивидуальная задача из ВП задана графом $G = (V, E)$ и положительным целым числом $K \leqslant |V|$. Построим граф $G' = (V', E')$, такой, что в G' есть гамильтонов цикл тогда и только тогда, когда в G' есть вершинное покрытие, состоящее из не более чем K элементов.

В данном случае наша конструкция опять будет состоять из нескольких компонент, соединённых линиями связи. Во-первых, граф G' содержит компоненту, «проверяющую покрытие», которая будет использована для того, чтобы обеспечить присутствие одного из концов каждого ребра среди K выбираемых вершин. Компонента, соответствующая ребру $e = \{u, v\} \in E$, показана на рис. 3.4.

Она имеет 12 вершин

$$V'_e = \{(u, e, i), (v, e, i)\} : 1 \leqslant i \leqslant 6$$

и 14 рёбер

$$\begin{aligned} E'_e = \{&\{(u, e, i), (u, e, i+1), (v, e, i), (v, e, i+1)\} : 1 \leqslant i \leqslant 5\} \\ &\cup \{\{(u, e, 3), (v, e, 1), (v, e, 3), (u, e, 1)\}\} \\ &\cup \{\{(u, e, 6), (v, e, 4), (v, e, 6), (u, e, 4)\}\}. \end{aligned}$$

Когда конструкция будет закончена, то дополнительные рёбра, ведущие к этой проверяющей покрытие компоненте, будут инцидентны только вершинам $(u, e, 1)$, $(v, e, 1)$, $(u, e, 6)$ и $(v, e, 6)$. Отсюда следует (в этом читатель может легко убедиться), что любой гамильтонов цикл в графе G' должен пересекать рёбра из E'_e по одной из трёх конфигураций, изображённых на рис. 3.5. Например, если цикл «подходит» к рассматриваемой компоненте в

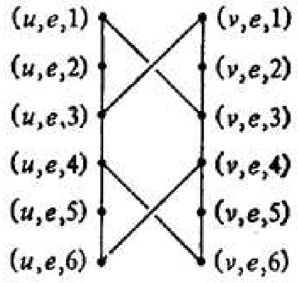


Рис. 3..4: *Компонента для проверки покрытия, соответствующая ребру* $e = \{u, v\}$ *в сведении задачи ВЕРШИННОЕ ПОКРЫТИЕ к задаче ГАМИЛЬТОНОВ ЦИКЛ.*

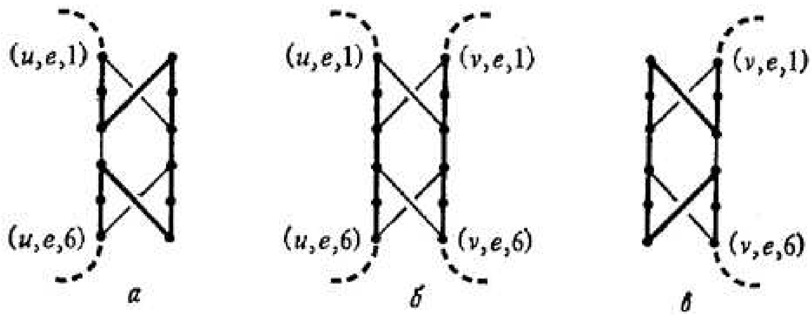


Рис. 3..5: *Три возможные конфигурации пересечения гамильтонова цикла с компонентой для проверки покрытия ребра* $e = \{u, v\}$. *Конфигурации соответствуют следующим случаям: (а)* u *принадлежит, а* v *не принадлежит покрытию: (б)* u *и* v *принадлежат покрытию; (в)* v *принадлежит, а* u *не принадлежит покрытию.*

вершине $(u, e, 1)$, то он «выходит» обязательно в вершине $(u, e, 6)$ и проходит либо через все 12 вершин этой компоненты, либо только через 6 вершин (u, e, i), $1 \leqslant i \leqslant 6$.

Дополнительные рёбра в окончательной конструкции служат либо для соединения пар компонент проверки покрытия, либо для соединения компоненты проверки покрытия с выбирающей вершиной. Для каждой вершины $v \in V$ упорядочим произвольным образом рёбра, инцидентные v : $e_{v[1]}, e_{v[2]}, ..., e_{v[deg(v)]}$, где $deg(v)$ – *степень* вершины v в графе G (т.е. число рёбер, инцидентных вершине v). Все компоненты проверки покрытия, соответствующие рёбрам, инцидентным v, соединяются вместе следующими связующими рёбрами:

$$E'_v = \{\{(v, e_{v[i]}, 6), (v, e_{v[i+1]}, 1)\} : 1 \leqslant i \leqslant deg(v)\}.$$

Как показано на рис. 3.6, такое соединение образует единственный путь в G', включающий в точности те вершины (x, y, z), для которых $x = v$.

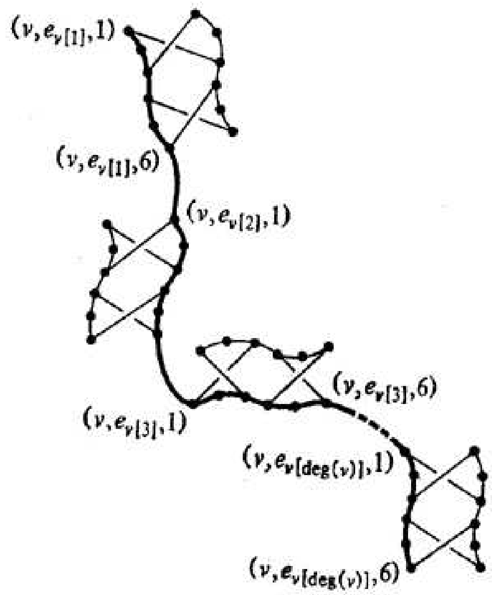


Рис. 3..6: *Путь, соединяющий все компоненты проверки покрытия, которые соответствуют рёбрам из E, инцидентным вершине v.*

Последняя группа связующих рёбер в G' соединяет первую и последнюю вершины этого пути с каждой из выбирающих вершин $a_1, a_2, ..., a_k$. Эти рёбра имеют следующий вид:

$$E'' = \{\{a_i, (v, e_{v[1]}, 1)\}, \{a_i, (v, e_{v[deg(v)]}, 6)\} 1 \leqslant i \leqslant K, v \in V\}.$$

Окончательно граф $G' = (V', E')$ имеет вид

$$V' = \{a_i : 1 \leqslant i \leqslant K\} \cup \left(\bigcup_{e \in E} V'_e\right)$$

и

$$E' = \left(\bigcup_{e \in E} E'_e\right) \cup \left(\bigcup_{v \in V} E'_v\right) \cup E''.$$

Нетрудно видеть, что граф G' можно построить за полиномиальное время исходя из G и K.

Мы утверждаем, что G' имеет гамильтонов цикл тогда и только тогда, когда G имеет вершинное покрытие, состоящее не более чем из K вершин. Предположим, что $(v_1, v_2, ..., v_n)$ – гамильтонов цикл в графе G' (здесь $n = |V'|$). Рассмотрим любую часть этого цикла, которая начинается и кончается в вершинах множества $\{a_1, a_2, ..., a_K\}$ и не содержит ни одной вершины этого множества в качестве промежуточной. В силу ранее упомянутых ограничений на конфигурацию, по которой гамильтонов цикл может проходить через компоненту проверки покрытия, рассматриваемая часть цикла должна проходить только через компоненты проверки покрытия, соответствующие рёбрам из E, инцидентным некоторой общей вершине $v \in V$. Каждая компонента проверки покрытия пересекается одним из способов (а), (б) или (в), указанных на рис. 3.5, причём она не проходит через вершины ни из какой другой компоненты проверки покрытия.

Таким образом, K вершин множества $\{a_1, a_2, ..., a_K\}$ делят рассматриваемый гамильтонов цикл на K путей, соответствующих некоторым вершинам $v \in V$. Поскольку гамильтонов цикл должен проходить через все вершины каждой компоненты проверки покрытия, причём вершины из компоненты проверки покрытия ребра $e \in E$ могут принадлежать только пути, соответствующему одной из вершин, инцидентных e, то любое ребро в E должно быть инцидентно по крайней мере одной из K выбранных вершин. Следовательно, это множество, состоящее из K вершин, образует искомое вершинное покрытие графа G.

Обратно, предположим, что $V^* \subseteq V$ – вершинное покрытие графа G и $|V^*| \leqslant K$. Можно считать, что $|V^*| = K$, поскольку если к V^* добавить несколько вершин из V, то V^* останется вершинным покрытием. Обозначим элементы V^* через $v_1, v_2, ..., v_K$. Рассмотрим, какие рёбра нужно включить в гамильтонов цикл графа G. Из компоненты проверки покрытия, представляющей ребро $e = \{u, v\} \in E$, выберем рёбра, изображённые на рис. 3.5 (а), (б) или (в), в зависимости от того, равно ли $\{u, v\} \cap V^*$ соответственно $\{u\}$, $\{u, v\}$ или $\{v\}$. Поскольку V^* – вершинное покрытие графа G, то одна из этих возможностей обязательно должна быть реализована. Затем возьмём все рёбра в E'_{v_i}, $1 \leqslant i \leqslant K$. Наконец, выберем рёбра

$$\{a_i, (v_i, e_{v_i}[1], 1)\}, 1 \leqslant i \leqslant K,$$

$$\{a_{i+1}, (v_i, e_{v_{i[deg(v_i)]}}, 6)\}, 1 \leqslant i \leqslant K,$$

и

$$\{a_i, (v_K, e_{v_{K[deg(v_K)]}}, 6)\}.$$

Проверить то, что выбранное множество рёбер в действительности соответствует гамильтонову циклу в G', мы предоставляем читателю. ■

Рассмотрим несколько вариантов задачи ГАМИЛЬТОНОВ ЦИКЛ. Задача ГАМИЛЬТОНОВ ПУТЬ формулируется так же, как задача ГЦ, но без условия, что первая и последняя вершины в последовательности должны быть соединены ребром. Задача ГАМИЛЬТОНОВ ПУТЬ МЕЖДУ ДВУМЯ ВЕРШИНАМИ отличается от задача ГАМИЛЬТОНОВ ПУТЬ только тем, что в условии этой задачи фиксируются две вершины u и v и спрашивается, верно ли, что G содержит гамильтонов путь из u в v. *NP*-полнота обеих этих задач может быть доказана с помощью простой модификации сводимости, использованной для доказательства *NP*-полноты задачи ГЦ. Модифицируем граф G', который получается в конце рассмотренной выше конструкции следующим образом: добавим три новые вершины a_0, a_{K+1} и a_{K+2}, два новых ребра $\{a_0, a_1\}$ и $\{a_{K+1}, a_{K+2}\}$, а также заменим каждое ребро вида $\{a_1, (v, e_{v[deg(v)]}, 6)\}$ на ребро $\{a_{K+1}, (v, e_{v[deg(v)]}, 6)\}$. В последней модификации задачи ГЦ выделенными вершинами являются a_0 и a_{K+1}.

Все три перечисленные выше задачи остаются *NP*-полными и в том случае, если неориентированный граф G заменить ориентированным, а неориентированный гамильтонов цикл или путь заменить на ориентированный. Вспомним, что ориентированный граф $G = (V, A)$ состоит из множества вершин V и множества A *упорядоченных* пар вершин, называемых *дугами*. Гамильтоновым путём в ориентированном графе $G = (V, A)$ называется такая последовательность $< v_1, v_2, ..., v_n >$ $(n = |V|)$ вершин графа G, что $< v_i, v_{i+1} >\in A$, $1 \leqslant i < n$. Гамильтонов путь называется гамильтоновым циклом, если $< v_n, v_1 >\in A$. Каждая из упомянутых выше задач на неориентированных графах может быть сведена к задаче на ориентированном графе заменой ребра $\{u, v\}$ неориентированного графа парой дуг (u, v) и (v, u). По существу, задачи на неориентированных графах являются частными случаями соответствующих задач на ориентированных графах.

3.1.5. Разбиение

В настоящем разделе будет рассмотрена последняя из шести основных *NP*-полных задач – задача РАЗБИЕНИЕ. Это задача особенно полезна при доказательстве *NP*-полноты задач, условие которых содержит такие числовые параметры, как длины, веса, стоимости, пропускные способности и т.д.

Теорема 3.5. *Задача РАЗБИЕНИЕ является NP-полной.*

Доказательство 5. Легко видеть, что РАЗБИЕНИЕ $\in$ *NP*. В самом деле, недетерминированный алгоритм должен угадать только подмножество A' множества A и за полиномиальное время проверить, что сумма размеров элементов из A' такая же, как сумма размеров элементов из $A \setminus A'$.

Сведём задачу 3-С к задаче РАЗБИЕНИЕ. Пусть множества W, X, Y при $|W| = |X| = |Y| = q$ и подмножество $M \subseteq W \times X \times Y$ образуют произвольную индивидуальную задачу из 3-С. Обозначим элементы этих множеств следующим образом:

$W = \{w_1, w_2, ..., w_q\}$,
$X = \{x_1, x_2, ..., x_q\}$,
$Y = \{y_1, y_2, ..., y_q\}$,
и

$M = \{m_1, m_2, ..., m_q\}$, где $k = |M|$.

Требуется построить множество A и размеры $s(a) \in Z^+$ всех элементов $a \in A$ так, чтобы A содержало подмножество A', для которого справедливо

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$$

в том и только в том случае, если M содержит трёхмерное сочетание.

Множество A будет содержать $k + 2$ элемента и строиться в два шага. Первыми k элементами множества A будут $\{a_i : 1 \leqslant i \leqslant k\}$, где a_i ассоциируется с тройкой $m_i \in M$. Размер $s(a_i)$ элемента a_i определяется с помощью двоичного представления числа $s(a_i)$. Слово из нулей и единиц, соответствующее $s(a_i)$, разбивается на $3q$ «зон» по $p = \lceil \log_2(k+1) \rceil$ бит в каждой. Каждая из этих зон помечается некоторым элементом из $W \cup X \cup Y$ так, как указано на рис. 3.7.

Двоичное представление числа $s(a_i)$ зависит от соответствующей тройки $m_i = (w_{f(i)}, x_{g(i)}, y_{h(i)}) \in M$ (где f, g и h – функции, задающие индексы первой, второй и третьей компонент каждой тройки m_i). В числе $s(a_i)$ правые концы зон, соответствующие числа $w_{f(i)}, x_{g(i)}, y_{h(i)}$, помечены 1, остальные знаки числа $s(a_i)$ – нули. Другими словами, $s(a_i) = 2^{p(3q-f(i))} + 2^{2q-g(i))} + 2^{p(q-h(i))}$.

Так как двоичная запись каждого из чисел $s(a_i)$ имеет длину не более $3pq$, то ясно, что $s(a_i)$ можно построить по заданной индивидуальной задаче из 3-С за полиномиальное время.

На данной стадии конструкции важно отметить, что если просуммировать содержимое одной зоны всех элементов множества $\{a_i : 1 \leqslant i \leqslant k\}$, то результат всегда не будет превосходить $2^p - 1$. Следовательно, при суммировании $\sum_{a \in A'} s(a)$ по некоторому подмножеству $A' \subseteq \{a_i : 1 \leqslant i \leqslant k\}$ никогда не придётся переносить единицы из одной зоны в соседнюю.

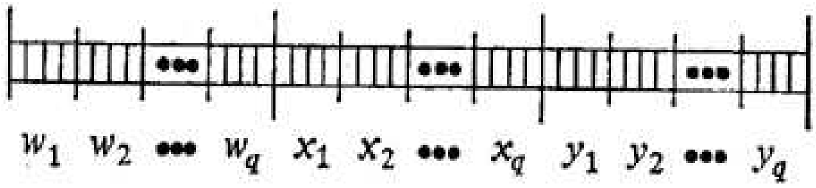


Рис. 3..7: *Двоичная запись числа $s(a)$ с помеченными $3q$ зонами с $p = \lceil \log_2(k+1) \rceil$ двоичными знаками в каждой, которая используется в сведении 3-С к РАЗБИЕНИЕ.*

Отсюда следует, что если положить

$$B = \sum_{j=0}^{3q-1} 2^{pj}$$

(B – число, в двоичной записи которого в правом конце каждой зоны стоит 1), то для любого подмножества $A' \subseteq \{a_i : 1 \leqslant i \leqslant k\}$ соотношение

$$\sum_{a \in A'} s(a) = B$$

выполняется тогда и только тогда, когда $M' = \{m_i : a_i \in A'\}$ – трёхмерное сочетание для M.

На последней стадии конструкции строятся оставшиеся два элемента из A. Они определяются элементами b_1 и b_2, имеющими следующие размеры:

$$s(b_1) = 2\left(\sum_{i=1}^{k} s(a_i)\right) - B,$$

$$s(b_2) = 2\left(\sum_{i=1}^{k} s(a_i)\right) + B^1$$

Двоичные записи этих величин имеют длину не более $3pq + 1$ и могут быть построены за время, зависящее полиномиальным образом от размера индивидуальной задачи из 3-С.

Теперь предположим, что имеется подмножество $A' \subseteq A$, такое, что

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a).$$

Тогда каждая из этих сумм должна быть равна $2\left(\sum_{i=1}^{k} s(a_i)\right)$, причём одно из множеств A' или $A \setminus A'$ содержит b_1 и не содержит b_2. Отсюда следует, что остальные элементы этого множества принадлежат множеству $\{a_i : 1 \leqslant i \leqslant k\}$, сумма их размеров равна B и, в силу сделанного замечания, они образуют подмножество, соответствующее некоторому трёхмерному сочетанию M' в M. Обратно, если задано трёхмерное сочетание $M' \subseteq M$, то множество $\{b_1\} \cup \{a_i : m_i \in M'\}$ образует искомое множество A' для заданной индивидуальной задачи из ПАРОСОЧЕТАНИЕ. Таким образом, задача 3-С $\propto$ РАЗБИЕНИЕ, и теорема доказана.

■

3.2. Некоторые методы доказательства *NP*-полноты

Методы, используемые при доказательстве результатов об *NP*-полноте, меняются почти в столь же широких пределах, как и сами *NP*-полные задачи,

[1] Если в множестве M нет трёхмерного сочетания, то, вообще говоря, $s(b_1)$ может быть отрицательно, что противоречит определению $s(a_i) \in Z^+$. В этом случае, для справедливости текста доказательства следует положить $s(b_1) = 0$, $s(b_2) = 3\left(\sum_{i=1}^{k} s(a_i)\right)$.– *Прим. ред.*

поэтому здесь у нас нет возможности проиллюстрировать их все. Однако имеется несколько общих приёмов доказательства, которые часто встречаются и могут подсказать путь к доказательству *NP*-полноты новой задачи. Они называются: (а) сужение задачи, (б) локальная замена и (в) построение компоненты.

В настоящем разделе, в основном на примерах, мы объясним, что понимается под каждым из этих приёмов доказательства. Было бы ошибкой пытаться дать их явное определение. Многие доказательства допускают интерпретации, позволяющие причислить их к любой из этих трёх категорий. Некоторые доказательства так сильно зависят от специфики описания задачи, что их не удаётся классифицировать естественным образом в столь ограниченном множестве категорий. Так что читателю не следует рассматривать нижеизложенное как попытку классификации всех доказательств *NP*-полноты. Наша единственная цель – проиллюстрировать некоторые подходы к доказательству *NP*-полноты, которые мы (а также другие авторы) с интуитивной точки зрения нашли конструктивными и привлекательными.

Для краткости мы будем в дальнейшем опускать во всех доказательствах проверку того, что рассматриваемая задача принадлежит *NP*. Относительно каждой из рассматриваемых ниже задач легко установить разрешимость за полиномиальное время недетерминированным алгоритмом, и у читателя не должно возникнуть затруднений, если потребуется построить такой алгоритм.

3.2.1. Сужение задачи

Из трёх рассматриваемых приёмов доказательства метод сужения – самый простой и, вероятно, наиболее часто встречается.

Доказательство методом сужения *NP*-полноты фиксированной задачи $\Pi \in NP$ заключается просто-напросто в установлении того, что задача Π включает в качестве частного случая известную *NP*-полную задачу Π'. Суть состоит в том, чтобы указать дополнительные ограничения, которые требуется наложить на индивидуальные задачи из Π, чтобы получившаяся в результате сужения задача была бы эквивалентна Π'. При этом не требуется, чтобы возникающая в результате задача была *точной* копией известной *NP*-полной задачи, необходимо только, чтобы между задачами имелось «очевидное» взаимно-однозначное соответствие, сохраняющее ответы «да» или «нет». Взаимно-однозначное соответствие, которое даёт введение Π' к Π, обычно настолько очевидно, что его даже не требуется указывать явно.

Несколько примеров доказательств этого типа уже встречалось в разд. 3.1.2. *NP*-полнота задачи ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ была доказана путём рассмотрения среди её индивидуальных задач только тех, в которых трёхэлементные множества содержат по одному элементу из множеств W, X и Y (где W, X и Y – равномощные непересекающиеся множества). Такие задачи идентичны задаче 3-С. В разд. 3.1.4 *NP*-полнота задачи ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ была доказана благодаря тому, что среди её индивидуальных задач мы рассматривали только те, в которых ориентированных граф вместе с каждой дугой (u, v) содержит противоположно ориентированную дугу (v, u), т.е. индивидуальные задачи, в совокупности образующие задачу, идентичную задаче ГАМИЛЬТОНОВ ЦИКЛ.

Таким образом, метод сужения можно рассматривать скорее как иной взгляд на задачу, а не как стандартный способ доказательства NP-полноты. Вместо того чтобы пытаться свести одну из известных NP-полных задач к заданной, мы концентрируем внимание на последней и пытаемся отбросить «несущественные» её детали с тем, чтобы получилась известная NP-полная задача.

Ниже приведено ещё несколько примеров задач, NP-полнота которых доказывается методом сужения; при этом доказательства формулируются настолько кратко, насколько это возможно.

1. МИНИМАЛЬНОЕ ПОКРЫТИЕ

УСЛОВИЕ. Задано семейство C подмножеств множества S положительное целое число K.

ВОПРОС. Содержит ли C *покрытие* множества S размера не более K, т.е. найдётся ли подмножество $C' \subseteq C$, такое, что $|C'| \leqslant K$ и $\bigcup_{c \in C} c = S$?

Доказательство. Задача превращается в ТП-3, если ограничиться только теми индивидуальными задачами, у которых $|C| = 3$ для всех множеств $C \in C$ и $K = |S|/3$.

2. МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ

УСЛОВИЕ. Задано семейство C подмножеств множества S и положительное целое число K.

ВОПРОС. Содержит ли S множество *представителей* для C мощности, не превосходящей K; т.е. существует ли подмножество $S' \subseteq S$, такое, что $|S'| \leqslant K$ и S' содержит по крайней мере один элемент из каждого множества семейства C?

Доказательство. Задача превращается в задачу ВП, если рассмотреть только те индивидуальные задачи, у которых $|c| = 2$ для всех $c \in C$.

3. ИЗОМОРФИЗМ ПОДГРАФУ

УСЛОВИЕ. Заданы два графа $G = (V_1, E_1)$ и $H = (V_2, E_2)$.

ВОПРОС. Содержит ли граф G подграф, *изоморфный* H? Другими словами, существуют ли такие подмножества $V \subseteq V_1$, $E \subseteq E_1$ и взаимнооднозначная функция $f : V_2 \to V$, что $|V| = |V_2|$, $|E| = |E_2|$ тогда и только тогда, когда $\{f(u), f(v)\} \in E$?

Доказательство. Задача превращается в задачу КЛИКА, если рассматривать только те индивидуальные задачи, для которых $H-$ полный граф, т.е. индивидуальные задачи, в которых E_2 содержит все возможные рёбра, соединяющие вершины из V_2.

4. Остовное дерево ограниченной степени

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число K, $K \leqslant |V| - 1$.

ВОПРОС. Существует ли в G *остовное дерево*, в котором все вершины имеют степень не более K, т.е. такое подмножество $E' \subseteq E$, что $|E'| = |V|-1$, граф $G' = (V, E')$ связен и ни одна вершина в V не принадлежит более чем K рёбрам из E'?

Доказательство. Если рассмотреть только те индивидуальные задачи, в которых $K = 2$, то задача превращается в задачу ГАМИЛЬТОНОВ ПУТЬ.

5. МИНИМАЛЬНЫЙ ЭКВИВАЛЕНТНЫЙ ОРИЕНТИРОВАННЫЙ ГРАФ

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число K, $K \leqslant |A|$.

ВОПРОС. Существует ли ориентированный граф $G' = (V, A')$, такой что $A' \subseteq A$, $|A'| \leqslant K$ и для каждой пары вершин u и v из V в G' имеется ориентированный путь от u к v тогда и только тогда, когда в G имеется ориентированный путь из u к v.

Доказательство. Если рассматривать только индивидуальные задачи, для которых граф G сильно связен, т.е. содержит ориентированный путь от любой вершины к любой другой вершине v и $|K| = |V|$, то задача превращается в задачу ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ В СИЛЬНО СВЯЗНОМ ОРИЕНТИРОВАННОМ ГРАФЕ, однако NP-полнота этой задачи немедленно следует из приведённых выше конструкций для задач ГЦ и ОРИЕНТИРОВАННЫЙ ГЦ.

6. РЮКЗАК

УСЛОВИЕ. Задано конечное множество U, «размеры» $s(u) \in Z^+$, «стоимости» $v(u) \in Z^+$ для всех $u \in U$, общее ограничение B на размеры, $B \in Z^+$, и значение стоимости $K \in Z^+$.

ВОПРОС. Существует ли подмножество $U' \subseteq U$, такое, что

$$\sum_{u \in U'} s(u) \leqslant B \quad \text{и} \quad \sum_{u \in U'} v(u) \geqslant K?$$

Доказательство. Если ограничиться рассмотрением только таких индивидуальных задач, в которых для всех $u \in U$, $s(u) = v(u)$ и $B = K = \frac{1}{2}\sum_{u \in U'} s(u)$, то задача превращается в РАЗБИЕНИЕ.

7. РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ

УСЛОВИЕ. Задано конечное множество A «заданий», «длительности» $l(a) \in Z^+$ для всех $a \in A$, число $m \in Z^+$ «процессоров» и «директивный срок» $D \in Z^+$.

ВОПРОС. Существует ли разбиение $A = A_1 \cup A_2 \cup \ldots A_m$ множества A на m непересекающихся множеств, такое, что

$$\max\left\{\sum_{a \in a_i} l(a) : 1 \leqslant i \leqslant m\right\} \leqslant D?$$

Доказательство. Если рассматривать только те индивидуальные задачи, для которых $m = 2$ и $D = \frac{1}{2}\sum_{a\in A} l(a)$, то задача превращается в РАЗБИЕНИЕ.

В заключение заметим, что из трёх обсуждаемых подходов к установлению *NP*-полноты метод сужения в наибольшей степени выигрывает от наличия обширного списка *NP*-полных задач, сверх основных шести задач и их вариантов. Многие возникающие на практике задачи представляют собой просто более сложные версии задач из списка *NP*-полных задач в приложении, и способность усмотреть этот факт во многих случаях позволяет быстро доказать *NP*-полноту методом сужения.

3.2.2. Локальная замена

Сводимости, возникающие при доказательстве методом локальной замены, достаточно нетривиальны, чтобы их всегда можно было с гарантией представить в стандартном виде, однако они остаются относительно несложными. Этот метод состоит в том, что выбирается некоторое характерное свойство известной *NP*-полной задачи, с помощью него образуется семейство *основных модулей*, а соответствующие индивидуальные задачи заданной задачи получаются путём единообразной замены каждого основного модуля некоторой другой структурой. Сводимость задачи ВЫП к задаче 3-ВЫП из разд. 3.1.1 относится к этому типу. В ней основными модулями задачи ВЫП были дизъюнкции, и каждая дизъюнкция заменялась набором дизъюнкций согласно некоторому общему правилу. Важным моментом здесь является то, что каждая замена приводила только к локальному изменению структуры. Эти замены, по существу, не зависели одна от другой, за исключением тех случаев, когда они отражали не подвергавшиеся изменению части исходной индивидуальной задачи.

Конкретизируем эти общие рассуждения с помощью примеров. Рассмотрим сначала задачу распознавания, соответствующую задаче минимизации числа умножений, необходимых для вычисления заданного набора произведений элементарных термов; при этом предполагается, что операций умножения ассоциативна и коммутативна.

ВЫЧИСЛЕНИЕ СЕМЕЙСТВА

УСЛОВИЕ. Задано семейство C подмножеств конечного множества A и положительное целое число J.

ВОПРОС. Существует ли такая последовательность

$$< z_1 = x_1 \cup y_1, z_2 = x_2 \cup y_2, \ldots, z_j = x_j \cup y_j >,$$

содержащая не более чем J операций типа объединение, что для каждого подмножества $c \in C$ найдётся номер i, $1 \leqslant i \leqslant j$, такой, что z_i совпадает с c? (Здесь x_i и y_i – это множества вида $\{a\}$ для некоторого $a \in A$, либо это множество вида z_k для некоторого $k < i$, причём x_i и y_i, $1 \leqslant i \leqslant j$, не пересекаются).

Теорема 3.6. *Задача ВЫЧИСЛЕНИЕ СЕМЕЙСТВА NP-полна.*

Доказательство 1.

Сведём задачу ВЕРШИННОЕ ПОКРЫТИЕ к задаче ВЫЧИСЛЕНИЕ СЕМЕЙСТВА. Пусть граф $G = (V, E)$ и положительное целое число K, $K \leqslant |V|$, определяют условие произвольной индивидуальной задачи из ВП.

В качестве основных модулей ВП возьмём рёбра графа G. Пусть a_0 – некоторый новый элемент, не принадлежащий V. Операция локальной замены замещает каждое ребро $\{u, v\} \in E$ подмножеством $\{a_0, u, v\} \in C$. В результате возникает индивидуальная задача из задачи ВЫЧИСЛЕНИЕ СЕМЕЙСТВА, условие которой полностью определяется следующими данными:

$A = V \cup \{a_0\}$,

$C = \{\{a_0, u, v\} : \{u, v\} \in E\}$,

$J = K + |E|$,

Легко видеть, что эту индивидуальную задачу можно построить за полиномиальное время. Утверждается, что в G есть вершинное покрытие не более чем из K элементов тогда и только тогда, когда для семейства C существует последовательность длины не более J, обладающая необходимыми свойствами.

Вначале предположим, что V' – вершинное покрытие в G, содержащее не более K вершин. Если к покрытию V' добавить новые вершины, то оно по-прежнему будет оставаться покрытием, поэтому без ограничения общности можно полагать, что $|V| = K$. Обозначим вершины покрытия V' через $e_1, e_2, \ldots, e_m$, где $m = |E|$. Поскольку V' – вершинное покрытие, то каждое ребро e_j инцидентно по крайней мере одной вершине из V'. Поэтому каждое ребро e_j может быть представлено в виде $e_j = \{u_j, v_{r[j]}\}$, где $r[j]$– целое число, заключённое между 1 и K. Нетрудно видеть, что приведённая ниже последовательность, состоящая из $K + |E| = J$ операций объединения, обладает всеми необходимыми свойствами:

$$< z_1 = \{a_0\} \cup \{v_1\}, z_2 = \{a_0\} \cup \{v_2\}, \ldots, z_K = \{a_0\} \cup \{v_K\},$$

$$z_{K+1} = \{u_1\} \cup \{z_{r[1]}\}, z_{K+2} = \{u_2\} \cup \{z_{r[2]}\}, \ldots, z_j = \{u_m\} \cup \{z_{r[m]}\} > .$$

Обратно, предположим, что $S =< z_1 = x_1 \cup y_1, \ldots, z_j = x_j \cup y_j >$ – искомая последовательность для индивидуальной задачи из ВЫЧИСЛЕНИЕ СЕМЕЙСТВА, содержащая не более J операций объединения. Предположим далее, что последовательность S–кратчайшая из всех последовательностей для рассматриваемой индивидуальной задачи и что среди всех таких кратчайших последовательностей S содержит минимальное число операций вида $z_i = \{u\} \cup \{v\}, u, v \in V$. Первое утверждение состоит в том, что в S *вообще нет* операций этого вида. Действительно, предположим, что операция $z_i = \{u\} \cup \{v\}, u, v \in V$, содержится в S. Поскольку $\{u, v\}$ не принадлежит C, а S имеет минимальную возможную длину, то с необходимостью имеем $\{u, v\} \in E$ и операция $\{a_0, u, v\} = \{a_0\} \cup z_i$ (или $z_i \cup \{a_0\}$) должна содержаться в S. Но поскольку $\{u, v\}$–подмножество только одного элемента из

C, то z_i не может участвовать ни в одной другой операции рассматриваемой последовательности S. Отсюда следует, что пара операций

$$z_i = \{u\} \cup \{v\} \text{и} \{a_0, u, v\} = \{a_0\} \cup z_i$$

может быть заменена другой парой

$$z_i = \{a_0\} \cup \{u\} \text{и} \{a_0, u, v\} = \{v\} \cup z_i,$$

при этом длина всей последовательности S не увеличится, а общее число операций вида $\{u, v\} u, v \in V$ уменьшится, что противоречит выбору S. Следовательно, S состоит только из операций, имеющих вид $z_i = \{a_0\} \cup \{u\}, u \in V$ или $\{a_0, u, v\} = \{a_0\} \cup z_i, \{u, v\} \in E$ (мы пренебрегаем порядком этих операндов). Так как $|C| = E$ и любой элемент из C содержит три элемента, то S должно содержать в точности $|E|$ операций второго вида и в точности $j - |E| \leqslant J - |E| = K$ операций первого вида. Поэтому множество

$$V' = \{u \in V : z_i = \{a_0\} \cup \{u\} \text{ есть операция из } S\}$$

содержит самое большое K вершин из V. Исходя из способа построения C, нетрудно проверить, что V' должно быть вершинным покрытием в G.

■

Ниже приведён другой пример полиномиальной сводимости, которая получается из задачи ТОЧНОЕ ПОКРЫТИЕ ТРЁХЭЛЕМЕНТНЫМИ МНОЖЕСТВАМИ методом локальной замены.

РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ

УСЛОВИЕ. Задан граф $G = (V, E)$, число вершин которого $|V|$ делится на три, т.е. $|V| = 3q$ для некоторого целого числа q.

ВОПРОС. Существует ли такое разбиение V на q непересекающихся подмножеств $V_1, V_2, \dots, V_q$ (содержащих по 3 элемента), что для каждого $V_i = \{v_i[1], v_i[2], v_i[3]\}$ три ребра $\{v_i[1], v_i[2]\}$, $\{v_i[1], v_i[3]\}$ и $\{v_i[2], v_i[3]\}$ принадлежат E?

Теорема 3.7. *Задача РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ NP-полна.*

Доказательство 2. Сведём задачу ТОЧНОЕ ПОКРЫТИЕ ТРЁХЭЛЕМЕНТНЫМИ множествами к задаче РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ. Пусть множество $X (|X| = 3q)$ и семейство C его трёхэлементных подмножеств образуют условие произвольной индивидуальной задачи из ТП-3. Построим граф $G = (V, E)$, такой, что $|V| = 3q'$ и для него существует искомое разбиение тогда и только тогда, когда C содержит точное покрытие.

Основными модулями рассматриваемой индивидуальной задачи из ТП-3 будут трёхэлементные подмножества из C. Операция локальной замены «превращает» каждое такое подмножество $C_i = \{x_i, y_i, z_i\} \in C$ в набор

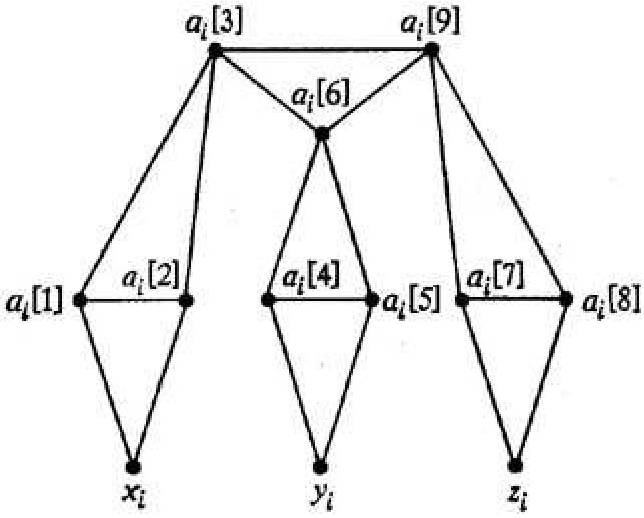


Рис. 3..8: *Элемент, используемый для местной замены подмножества $c_i = (x_i, y_i, z_i) \in C$ при построении сведения задачи ТП-3 к задаче РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ.*

E_i, состоящий из 18 рёбер, который изображён на рис. 3.8. Таким образом, $G = (V, E)$ определяется так:

$$V = X \bigcup_{i=1}^{|C|} \{a_i[j] : 1 \leqslant j \leqslant 9\},$$

$$E = \bigcup_{i=1}^{|C|} E_i.$$

Заметим, что только вершины из множества X могут быть инцидентны более чем одному множеству E_i. Отметим также, что

$$|V| = |X| + 9|C| = 3q + 9|C|,$$

так что $q' = q + 3|C|$. Нетрудно видеть, что эту индивидуальную задачу РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ можно построить, отправляясь от рассматриваемой индивидуальной задачи из ТП-3, за полиномиальное время.

Если $c_1, c_2, \ldots, c_q$ – трёхэлементные подмножества из C, которые образуют точное покрытие X, то соответствующее разбиение $V = V_1 \cup V_2 \cup \ldots \cup V_{q'}$ множества V определяется следующим образом. Если $c_i = \{x_i, y_i, z_i\}$ есть элемент точного покрытия, то из множества E_i выбираются следующие подмножества:

$$\{a_i[1], a_i[2], x_i\},\ \{a_i[4], a_i[5], y_i\},\ \{a_i[6], a_i[6], z_i\},\ \{a_i[3], a_i[6], a_i[9]\},$$

а если c_i ***не является*** элементом точного покрытия, то из E_i выбираются подмножества

$$\{a_i[1], a_i[2], a_i[3]\},\ \{a_i[4], a_i[5], a_i[6]\},\ \{a_i[7], a_i[8], a_i[9]\}.$$

Такой выбор подмножеств обеспечивает, что каждый элемент из X содержится ровно в одном трёхвершинном подмножестве рассматриваемого разбиения.

Обратно, если $V = V_1 \cup V_2 \cup ... \cup V_{q'}$ – произвольное разбиение графа G на треугольники, то соответствующее точное покрытие получается, если выбирать те $c_i \in C$, для которых $\{a_i[3], a_i[6], a_i[9]\} = V_j$ при некотором $j, 1 \leqslant j \leqslant q'$. Непосредственную проверку того, что два построенных разбиения обладают всеми нужными свойствами, мы оставляем читателю. ■

Оба только что приведённых примера представляют собой доказательство методом локальной замены «в чистом виде». Структура рассматриваемой индивидуальной задачи однозначно определялась структурой исходной задачи (*NP*-полнота которой установлена) и операцией локальной замены. Часто оказывается полезным дополнить рассматриваемую индивидуальную задачу вспомогательными элементам, которые играют роль «ограничителя»[2], налагающего дополнительные ограничения на способы получения ответа «да».

Если рассматриваемая задача имеет вид «Задана индивидуальная задача I; верно ли, что существует некоторый объект X_I с требуемым свойством?», то роль ограничителя в задаче I – сузить множество допустимых объектов X_I так, чтобы они отражали все варианты, возможные в исходной индивидуальной задаче. В то же время часть индивидуальной задачи I, получающаяся операцией локальной замены из исходной задачи, даёт различные варианты этих ответов и обеспечивает выполнение условий, требующихся от этих ответов. Элементы b_1 и b_2 из доказательства *NP*-полноты задачи РАЗБИЕНИЕ как раз и играют роль подобного ограничителя. Приведём ещё два примера на доказательство методом локальной замены с использованием ограничителей. Сначала рассмотрим следующую задачу о составлении расписания.

УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ

УСЛОВИЕ. Задано конечное множество «заданий» T и для каждого $t \in T$ целое число $r(t) \geqslant 0$ – время готовности, «директивный срок» $d(t) \in Z^+$ и «длительность» $l(t) \in X^+$.

ВОПРОС. Существует ли для T *допустимое расписание*, т.е. такая функция $\sigma : T \to Z^+$, что для всех $t \in T$ имеет место: (1) $\sigma(t) \geqslant r(t)$, (2) $\sigma(t) + l(t) \leqslant d(t)$, и (3) если $t' \in T \setminus \{t\}$, то либо $\sigma(t') + l(t') \leqslant \sigma(t)$, либо $\sigma(t') \geqslant \sigma(t) + l(t)$? (Другими словами, задание t «выполняется» с момента времени $\sigma(t)$ до момента $\sigma(t) + l(t)$, оно не может быть начато ранее момента $r(t)$, должно быть закончено не позднее $d(t)$, и временной интервал

[2]образный термин (enforcer), предложенный Зыманским [511].

выполнения задания t не может перекрываться с интервалом выполнения другого задания t'.)

Теорема 3.8. *Задача УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ NP-полна.*

Доказательство 3.

Сведём к этой задаче задачу РАЗБИЕНИЕ. Пусть конечное множество A и размеры $s(a)$ всех элементов из A составляют условие индивидуальной задачи РАЗБИЕНИЕ; пусть также $B = \sum_{a \in A} s(a)$.

Возьмём в качестве базисных модулей исходной индивидуальной задачи РАЗБИЕНИЕ отдельные элементы $a \in A$. Операция локальной замены превращает a в задание t_a, такое, что $r(t_a) = 0$, $d(t_a) = B + 1$ и $l(t_a) = s(a)$. В качестве «ограничителя» возьмём ещё одно задание $\bar{t}$, такое, что $r(\bar{t}) = \lceil B/2 \rceil, d(\bar{t}) = \lceil (B + 1)/2 \rceil$ и $l(\bar{t}) = 1$. Ясно, что эта индивидуальная задача может быть построена по исходной индивидуальной задаче РАЗБИЕНИЕ за полиномиальное время.

Ограничения, которые налагает ограничитель на допустимые расписания, имеют двоякий характер. Во-первых, ограничитель не позволяет построить допустимое расписание, если B нечётно (в этом случае для рассматриваемой задачи РАЗБИЕНИЕ искомое разбиение на два подмножества не существует), по той причине, что если B нечётно, то $r(\bar{t}) = d(\bar{t})$ и $\bar{t}$ вообще не может быть включено в расписание. Поэтому с настоящего момента будем предполагать, что B чётно. В этом случае на передний план выступает второе ограничение. Так как B чётно, то $r(\bar{t}) = B/2$ и $d(\bar{t}) = r(\bar{t}) + 1$ поэтому в любом допустимом расписании $\sigma(\bar{t})$ должно быть равно $B/2$. Отсюда время, которое доступно для выполнения остальных заданий, делится на два отдельных блока, каждый из которых, как показано на рис. 3.9, имеет

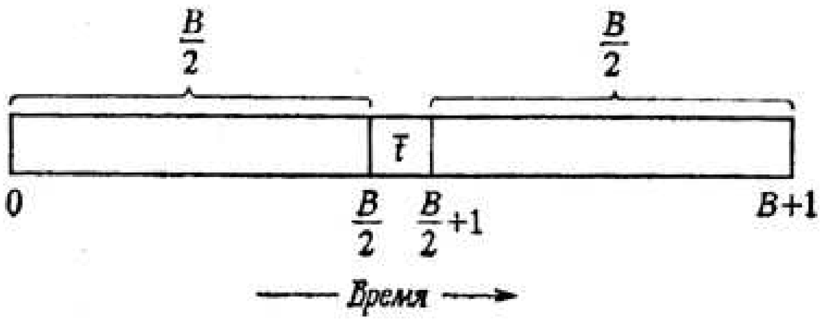


Рис. 3..9: *Расписание, получаемое с помощью «ограничителя» при сведении задачи РАЗБИЕНИЕ к задаче УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ.*

длину $B/2$. Таким образом, задача составления расписания превращается в задачу выбора подмножества заданий, которые будут выполняться раньше задания $\bar{t}$, и подмножества заданий, которые будут выполняться после

$\bar{t}$. Поскольку общее время, имеющееся в обоих блоках, равно общей длительности B оставшихся заданий, то отсюда следует, что каждый из этих блоков должен быть полностью заполнен. Это возможно тогда и только тогда, когда существует подмножество $A' \subseteq A$, такое, что

$$\sum_{a \in A'} s(a) = B/2 = \sum_{a \in A \setminus A'} s(a).$$

Таким образом, для рассматриваемой индивидуальной задачи РАЗБИЕНИЕ искомое подмножество A' существует тогда и только тогда, когда для соответствующей индивидуальной задачи УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ существует допустимое расписание.

■

В качестве заключительного примера использования ограничителя в методе локальной замены рассмотрим следующую задачу, возникающую в диагностическом тестировании.

МИНИМАЛЬНЫЙ НАБОР ТЕСТОВ

УСЛОВИЕ. Задано конечное множество A «возможных диагнозов», набор C подмножеств множества A представляющий бинарные «тесты», и положительное целое число $J \leqslant |C|$.

ВОПРОС. Существует ли поднабор $C' \subseteq C, |C'| \leqslant J$, такой, что для любой пары a_i, a_j возможных диагнозов из A имеется некоторых текст $c \in C'$ для которого $|\{a_i, a_j\} \cap c| = 1$ (т.е. тест c, «различающий» a_i и a_j)?

`Теорема 3.9.` *Задача МИНИМАЛЬНЫЙ НАБОР ТЕСТОВ NP-полна.*

Доказательство 4. Сведём к этой задаче задачу 3-С. Пусть W, X, Y такие, что $|W| = |X| = |Y| = q$, и подмножество $M \subseteq W \times X \times Y$ составляет условие индивидуальной задачи из 3-С.

В качестве основного модуля задачи 3-С возьмём упорядоченные тройки из множества M. Операция локальной замены вместо каждой тройки $m = (w, x, y) \in M$ образует подмножество $m = (w, x, y) \in C$. Роль ограничителей играют три дополнительных элемента w_0, x_0 и y_0, не принадлежащие $W \cup X \cup Y$, и два дополнительных теста $W \cup \{w_0\}$ и $X \cup \{x_0\}$. Окончательно индивидуальная задача МНИМАЛЬНЫЙ НАБОР ТЕСТОВ определяется следующим образом:

$$A = W \cup X \cup Y \cup \{w_0, x_0, y_0\},$$

$$C = \{\{w, x, y\} : (w, x, y) \in M\} \cup \{W \cup \{w_0\}, X \cup \{x_0\}\},$$

$$J = q + 2.$$

Легко видеть, что эта индивидуальная задача может быть построена по исходной задаче 3-С за полиномиальное время.

Опять-таки ограничитель налагает определённые ограничения на вид возможных компонент (в данном случае на набор тестов C'). Во-первых, набор C' должен содержать как $W \cup \{w_0\}$, так и $X \cup \{x_0\}$ по той причине, что только эти тесты отличают y_0 от w_0 и x_0. Далее, поскольку w_0, x_0, y_0 не принадлежат больше ни одному тесту из C то каждый элемент множества $W \cup X \cup Y$ должен отличаться от одного элемента w_0, x_0 или y_0 ввиду того, что он входит в некоторый дополнительный тест $c \in C' \setminus \{W \cup \{w_0\}, X \cup \{x_0\}\}$. При этом может быть использовано самое большее $J - 2 = q$ дополнительных тестов. Поскольку каждый из оставшихся тестов содержит ровно по одному элементу из множества W, X и Y, и, кроме того, множества W, X и Y попарно не пересекаются, и каждое из них содержит ровно q элементов, то отсюда следует, что любые q дополнительных тестов из C' должны соответствовать q тройкам из M, образующим трёхмерное сочетание. Обратно, если задано любое трёхмерное сочетание из M, то соответствующие q тестов из C можно использовать для завершения построения оставшихся тестов в нужном количестве. Таким образом, M содержит трёхмерное сочетание тогда и только тогда, когда в множестве C существует искомый набор тестов.

■

Хотя в обоих рассмотренных примерах ограничители очень просты, читатель должен отдавать себе отчёт в том, что это не всегда так. Ограничитель особо сложной конструкции используется в доказательстве *NP*-полноты задачи ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ПУТЬ В ПЛОСКОМ ГРАФЕ [157]. Другие довольно сложные ограничители можно найти в работах [143, 155, 357].

3.2.3. Построение компонент

Последний и наиболее сложный из рассматриваемых нами методов доказательств *NP*-полноты – это построение компонент. Доказательства *NP*-полноты задач ТРЁХМЕРНОЕ СОЧЕТАНИЕ, ВЕРШИННОЕ ПОКРЫТИЕ и ГАМИЛЬТОНОВ ЦИКЛ, приведённые в разд. 3.1., представляют собой типичные примеры доказательств этого типа.

Основная идея таких доказательств заключается в том, чтобы с помощью составных частей рассматриваемой задачи сконструировать некоторые «компоненты», соединяя которые можно «реализовать» индивидуальные задачи известной *NP*-полной задачи. В трёх перечисленных примерах имеются компоненты двух основных типов. Одну из них можно рассматривать как компоненту, «делающую выбор» (например, выбирающую вершины, выбирающую значение истинности переменных), а другую – как компоненту, «проверяющую свойства» (например, проверяющую, что каждое ребро покрыто или что каждая дизъюнкция выполнена).

В рассматриваемой индивидуальной задаче эти компоненты связаны так, что выбранные значения передаются компонентам, проверяющим условия, и последние проверяют, удовлетворяют ли сделанные выборы значений необходимым условиям. Взаимодействие компонент осуществляется с помощью прямых связей (таких, например, как рёбра, соединяющие компоненты набора значений истинности с компонентами проверки выполнения условий в сводимости задачи 3-ВЫП к задаче ВП), а также с помощью глобальных ограничений (таких, например, как общая граница K в сводимости задачи 3-ВЫП к задаче ВП, которая наряду со структурой компонент обеспечивает, чтобы каждая компонента наборов значений истинности содержала в точности одну вершину из покрытия и каждая компонента проверки свойств содержала ровно две вершины этого покрытия).

Вообще говоря, любое доказательство можно считать основанным на методе построения компонент, если конструируемая в нём индивидуальная задача представляет собой набор компонент, каждая из которых выполняет определённые функции, формулируемые в терминах исходной задачи. Общая сводимость, применённая в гл. 2 для доказательства теоремы Кука, является хорошим примером доказательства этого типа; при этом каждая из шести групп дизъюнкций представляет собой один из типов компонент.

Поскольку доказательства методом построения компонент часто оказываются довольно длинными и несколько примеров таких доказательств уже было рассмотрено, то в настоящем разделе мы приведём ещё только один пример. (Другие примеры можно найти в работах [114, 158, 477, 496].) Этот заключительный пример, сильно отличающийся от стандартных, иллюстрирует подход, который оказался полезным для сведения задачи КЛИКА к другим задачам. Здесь доказывается *NP*-полнота задачи теории расписаний, близкой по формулировке к задаче УПОРЯДОЧИВАНИЕ ВНУТРИ ИНТЕРВАЛОВ, рассмотренной в предыдущем подразделе.

УПОРЯДОЧЕНИЕ С МИНИМАЛЬНЫМ ЗАПАЗДЫВАНИЕМ

УСЛОВИЕ. Заданы множество T «заданий», каждое из которых имеет длительность 1 и «директивный» срок $d(t) \in Z^+$; частичное упорядочение $\lessdot$ на T и неотрицательное целое число $K \leqslant |T|$.

ВОПРОС. Существует ли «расписание» $\sigma : T \to \{0, 1, \dots, |T|-1\}$, такое, что (1) $\sigma(t) \neq \sigma(\bar{t})$ при $t \neq t'$, (2) $\sigma(t) < \sigma(t')$ при $t \lessdot t$ и (3) $|\{t \in T : \sigma(t)+1 > d(t)\}| \leqslant K$?

Теорема 3.10. *Задача УПОРЯДОЧЕНИЕ С МИНИМАЛЬНЫМ ЗАПАЗДЫВАНИЕМ NP-полна.*

Доказательство 5. Пусть граф $G = (V, E)$ и положительное целое число $J, J \leqslant |V|$, образуют произвольную индивидуальную задачу КЛИКА. Соответствующая индивидуальная задача из задачи УПОРЯДОЧЕНИЕ С МИНИМАЛЬНЫМ ЗАПАЗДЫВАНИЕМ имеет множество заданий $T = V \cup E$, $K = |E| - J(J-1)/2$; частичное упорядочение заданий определяется следующим образом:

$t \lessdot t' \Leftrightarrow t \in V, t' \in E$ и вершина t инцидентна ребру t';

$$d(t) = \begin{cases} J(J+1)/2, & \text{если } t \in E; \\ |V| + |E|, & \text{если } t \in V. \end{cases}$$

Таким образом, «компонента», соответствующая каждой вершине, состоит из единственного задания с директивным сроком $|V| + |E|$, а «компонента», соответствующая каждому ребру, состоит из единственного задания с директивным сроком $J(J + 1)/2$. Благодаря наличию частичного порядка на множестве T в искомом расписании задание, соответствующее ребру, следует за заданиями, соответствующими его концам, и поэтому задания, для которых имеется опасность запаздывания (т.е. завершения уже после директивного срока), обязательно соответствуют рёбрам.

Искомое расписание удобно представлять себе схематически, как показано на рис. 3.10.

Вершины клики	Рёбра клики		
$\overbrace{J}$ Задания, соответствующие вершинам	$\overbrace{J(J-1)/2}$ Задания, соответствующие рёбрам	$\vert V\vert - J$ Задания, соответствующие вершинам	$\vert E\vert - J(J-1)/2$ Задания, соответствующие рёбрам

0 $\frac{J(J+1)}{2}$ $|V| + |E|$

Время $\longrightarrow$

Рис. 3.10. Диаграмма искомого расписания для индивидуальной задачи УПОРЯДОЧЕНИЕ С МИНИМАЛЬНЫМ ЗАПАЗДЫВАНИЕМ, соответствующей КЛИКЕ размера J.

Часть расписания до директивного срока заданий, соответствующих рёбрам, можно рассматривать как «компоненту выбора клики». До этого срока может быть выполнено не более $J(J+1)/2$ заданий. Для того, чтобы число запоздавших заданий не превосходило заданной величины K, по крайней мере $J(J-1)/2$ из «ранних» заданий должны соответствовать рёбрам. Однако если задание, соответствующее ребру, выполняется ранее своего директивного срока, то и задания, соответствующие его концевым вершинам, также должны быть завершены раньше этого срока. Минимально возможное число вершин, которые могут быть инцидентны $J(J-1)/2$ различным рёбрам, равно J (причём это может быть тогда и только тогда, когда эти рёбра образуют полный граф на этих J вершинах). Отсюда следует, что среди «ранних» заданий должно иметься по *крайней мере* J заданий, соответствующих вершинам. Однако до директивного срока заданий, соответствующих рёбрам, может быть выполнено *не более*

$$J(J + 1)/2 - J(J - 1)/2 = J$$

заданий, отвечающих вершинам. Следовательно, в любом таком расписании до директивного срока заданий, соответствующих рёбрам, должно закончиться *ровно* J заданий, отвечающих вершинам и *ровно* $J(J-1)/2$ заданий, соответствующих рёбрам, которые в совокупности соответствуют клике на

J вершинах в графе G. Обратно, если граф G содержит полный подграф на J вершинах, то искомое расписание можно построить способом, указанным на рис. 3.10.

■

3.3. Упражнения

В настоящем разделе сформулированы двенадцать *NP*-полных задач, доказательства *NP*-полноты которых предоставляются читателю. Ни одна из этих задач не требует сложного доказательства, поэтому мы советуем попытаться найти их все. В этих доказательствах в качестве «известных» *NP*-полных задач достаточно использовать только задачи, упоминавшиеся в разд. 3.1. Для облегчения решения задачи сгруппированы по методам доказательства, которые представляются авторам наилучшими. Однако читатель может не обращать внимания на эти указания, если другой путь доказательства покажется ему более перспективным. Желающие выбрать дополнительные (или более трудные задачи) могут обратиться к списку из приложения, при этом, однако, следует иметь в виду, что в списке есть задачи, для которых известные доказательства *NP*-полноты весьма сложны.

Метод сужения

1. Самый длинный путь

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Имеется ли в G простой путь (т.е. путь, не проходящий дважды ни через одну вершину), состоящий не менее чем из K рёбер?

2. Упаковка множеств

УСЛОВИЕ. Заданы семейство C конечных множеств и положительное целое число $K, K \leqslant |C|$.

ВОПРОС. Верно ли, что в C имеется K непересекающихся множеств?

3. Разбиение на гамильтоновы подграфы

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Можно ли множество вершин графа G разбить на $k \leqslant K$ непересекающихся подмножеств $V_1, V_2, \dots, V_K$ так, чтобы для всех $i(1 \leqslant i \leqslant k)$ каждый подграф, индуцированный под множеством V_i, содержал гамильтонов цикл?

4. Разбиение на гамильтоновы подграфы

УСЛОВИЕ. Заданы графы $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ и положительное целое число K.

ВОПРОС. Существуют ли такие подмножества $E_1' \subseteq E_1$ и $F_2' \subseteq E_2$, что $|E_1'| = |E_2'| \geqslant K$, а подграфы $G_1' = (V_1, E_1')$, $G_2' = (V_2, E_2')$ изоморфны?

5. Минимум суммы квадратов

УСЛОВИЕ. Заданы конечное множество A, «размер» $s(a) \in Z^+$ для каждого $a \in A$ и положительные целые числа K и J.

ВОПРОС. Могут ли элементы из A быть разбиты на K непересекающихся множеств $A_1, A_2, \ldots, A_K$ так, что $\sum_{i=1}^{K} \left(\sum_{a \in A_i} s(a)\right)^2 \leqslant J$?

Метод локальной замены

6. Множество вершин, разрезающих контуры

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K, K \leqslant |V|$.

ВОПРОС. Существует ли такое подмножество $V' \subseteq V$, что $V' \leqslant K$ и любой ориентированный цикл в G содержит по крайней мере одно ребро из V'?

7. Точное покрытие четырёхэлементными множествами

УСЛОВИЕ. Задано конечное множество $X, |X| = 4q$, целое число q и семейство C четырёхэлементных подмножеств множества X.

ВОПРОС. Существует ли подсемейство $C' \subseteq C$ такое, что каждый элемент из X принадлежит в точности одному элементу из C'?

8. Доминирующее множество

УСЛОВИЕ. Заданы граф $G = (V, E)$ и целое число $K, K \leqslant |V|$.

ВОПРОС. Существует ли такое подмножество $V' \subseteq V$, что $V' \leqslant K$ и каждая вершина $v \in V \setminus V'$ соединена ребром по крайней мере с одной вершиной из V'?

9. Дерево Штейнера

УСЛОВИЕ. Заданы граф $G = (V, E)$, подмножество $R \subseteq V$ и положительное целое число $K \subseteq |V| - 1$.

ВОПРОС. Существует ли поддерево в G, содержащее все вершины из R и имеющее не более K рёбер?

10. Неэквивалентность регулярных выражений, не содержащих звёздочек

УСЛОВИЕ. Заданы два не содержащие звёздочек регулярных выражения E_1 и E_2 в конечном алфавите Σ. Такое выражение определяется следующим образом: (1) любой символ σ алфавита Σ есть не содержащее звёздочек регулярное выражение, (2) если e_1 и e_2 – два не содержащие звёздочек регулярных выражения, то и слова e_1e_2 и $(e_1 \vee e_2)$ также не содержащие звёздочек регулярные выражения.

ВОПРОС. Верно ли, что E_1 и E_2 представляют различные языки в алфавите Σ? (Язык, представляемый символом $\sigma \in \Sigma$, есть $\{\sigma\}$, а если e_1 и e_2 представляют соответственно языки L_1 и L_2, то e_1e_2 представляет язык $\{xy : x \in L_1, y \in L_2\}$, а $(e_1 \vee e_2)$ представляет язык $L_1 \cup L_2$.)

Метод построения компонент

11. Расщепление множества

УСЛОВИЕ. Заданы семейство C подмножеств конечного множества S.

ВОПРОС. Существует ли разбиение S на две части S_1 и S_2, такое, что ни одно подмножество из C не содержится ни в S_1 ни в S_2?

Указание. Использовать 3-ВЫП.

12. Разбиение на пути длины 2

УСЛОВИЕ. Заданы граф $G = (V, E)$, где $|V| = 3q$ для некоторого целого положительного числа q.

ВОПРОС. Существует ли разбиение V на q непересекающихся подмножеств $V_1, V_2, \ldots, V_q$, по три вершины в каждом, такое, что для всех $V_i = \{v_{i[1]}, v_{i[2]}, v_{i[3]}\}$ по крайней мере два из трёх рёбер $\{v_{i[1]}, v_{i[2]}\}$, $\{v_{i[1]}, v_{i[3]}\}$, $\{v_{i[2]}, v_{i[3]}\}$ принадлежат E?

Указание. Использовать 3-С.

13. Нумерация графа по Грунди

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$.

ВОПРОС. Существует ли такая нумерация $L : V \to Z^+$ (где каждой вершине может быть присвоен только один номер), что для всех $v \in V$ $L(v)$ есть наименьшее неотрицательное целое число, не принадлежащее множеству $\{L(u) : u \in V, (v, u) \in A\}$?

14. Раскрашиваемость графа в 3 цвета

УСЛОВИЕ. Задан ориентированный граф $G = (V, E)$.

ВОПРОС. Можно ли G раскрасить в 3 цвета?

Другими словами, существует ли функция $f; V \to \{1, 2, 3\}$, такая, что $f(u) \neq f(v)$, если $\{u, v\} \in E$?

Указание. Использовать 3-ВЫП.

Глава 4.

Применение теории *NP*-полноты для анализа задач

Теперь, когда мы овладели основными элементами теории *NP*-полноты, можно перейти к использованию этой теории для анализа задач.

Из рассуждений, приведённых в гл. 1, следует, что, встретившись с новой задачей, естественно сначала задать вопрос: «Можно ли рассматриваемую задачу решить полиномиальным алгоритмом?» Если ответ на этот вопрос оказывается положительным, то с точки зрения теории *NP*-полноты ничего большего о задаче сказать нельзя. Дальнейшие усилия мы можем сконцентрировать на поиске как можно более эффективных полиномиальных алгоритмов. Однако, если для решения задачи не известно полиномиального алгоритма, естественно возникает второй вопрос: «Верно ли, что рассматриваемая задача *NP*-полна?»

Для того чтобы поставленный вопрос имел смысл, предположим, что рассматриваемая задача сформулирована как задача распознавания и что она принадлежит классу *NP*. Заметим, что в некоторых случаях легко устанавливается полиномиальная разрешимость интересующей нас задачи, в других же оказывается очевидной её *NP*-полнота. Если задача оказалась *NP*-полной, то это является сильным аргументом в пользу того, что её *нельзя* решить за полиномиальное время.

В большинстве случаев, однако, найти ответ на каждый из поставленных вопросов довольно трудно. Обычно не очевидно, что задача разрешима за полиномиальное время или что он *NP*-полна, а требуется некоторая работа для выяснения, какая из этих двух возможностей в действительности реализуется (в том случае, если реализуется хотя бы одна из них; напомним, что в разд. 2.5 говорилось о том, что если $P \neq NP$, то в классе *NP* существуют задачи которые неразрешимы за полиномиальное время и не *NP*-полны). Как в этом случае выяснить сложность рассматриваемой задачи?

P	NP-полные
КРАТЧАЙШИЙ ПУТЬ между ДВУМЯ ВЕРШИНАМИ УСЛОВИЕ. Заданы: граф $G = (V, E)$, длины $l(e) \in Z^+$ для всех $e \in E$, выделенные вершины $a, b \in V$ и положительное целое число B. ВОПРОС. Существует ли в G простой путь из a в b, имеющий длину не более B?	**САМЫЙ ДЛИННЫЙ ПУТЬ между ДВУМЯ ВЕРШИНАМИ** УСЛОВИЕ. Заданы: граф $G = (V, E)$, длины $l(e) \in Z^+$ для всех $e \in E$, выделенные вершины $a, b \in V$ и положительное целое число B. ВОПРОС. Существует ли в G простой путь из a в b, имеющий длину не менее B?
РЁБЕРНОЕ ПОКРЫТИЕ УСЛОВИЕ. Заданы: граф $G = (V, E)$, и положительное целое число K. ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $\lvert E'\rvert \leqslant K$ и для каждой вершины $v \in V$ имеется такое ребро $e \in E'$, что $v \in e$?	**ВЕРШИННОЕ ПОКРЫТИЕ** УСЛОВИЕ. Заданы: граф $G = (V, E)$, и положительное целое число K. ВОПРОС. Существует ли подмножество $V' \subseteq V$, такое, что$\lvert V'\rvert \leqslant K$ и для любого ребра $e \in E$ имеется такая вершина $v \in V'$, что $v \in e$?
ТРАНЗИТИВНАЯ РЕДУКЦИЯ УСЛОВИЕ. Заданы: ориентированный граф $G = (V, A)$, и положительное целое число K. ВОПРОС. Существует ли подмножество $A' \subseteq V \times V$, такое, что $\lvert A'\rvert \leqslant K$ и для всех $u, v \in V$ граф $G' = (V, A')$ содержит путь из u в v тогда и только тогда, когда граф G содержит такой путь?	**МИНИМАЛЬНЫЙ ЭКВИВАЛЕНТНЫЙ ОРИЕНТИРОВАННЫЙ ПОДГРАФ** УСЛОВИЕ. Заданы: ориентированный граф $G = (V, A)$, и положительное целое число K. ВОПРОС. Существует ли подмножество $A' \subseteq A$, такое, что $\lvert A'\rvert \leqslant K$ и для всех $u, v \in V$ граф $G' = (V, A')$ содержит путь из u в v тогда и только тогда, когда граф G содержит такой путь?
РАСПИСАНИЕ ДЛЯ ПРЯМОГО ДЕРЕВА ЗАДАНИЙ УСЛОВИЕ. Заданы: множество T заданий с единичной длительностью, директивный срок $d(t) \in Z^+$ для каждого $t \in T$, частичный порядок $\lessdot$ на множестве T, относительно которого каждое задание имеет не более одного непосредственно следующего за ним, положительное целое число m. ВОПРОС. Можно ли составить для множества T расписание на m процессорах, на каждом из которых задания выполняются согласно заданному частичному порядку, так что все директивные сроки окажутся выполненными?	**РАСПИСАНИЕ ДЛЯ ОБРАТНОГО ДЕРЕВА ЗАДАНИЙ** УСЛОВИЕ. Заданы: множество T заданий с единичной длительностью, директивный срок $d(t) \in Z^+$ для каждого $t \in T$, частичный порядок $\lessdot$ на множестве T, относительно которого каждое задание имеет не более одного непосредственного предшественника, положительное целое число m. ВОПРОС. Можно ли составить для множества T расписание на m процессорах, на каждом из которых задания выполняются согласно заданному частичному порядку, так что все директивные сроки окажутся выполненными?

4.1. Пары близких задач, одна из которых принадлежит классу P, а другая NP-полна.

Если интуитивно мы склоняется в пользу одной из возможностей, то очень соблазнительно сосредоточить усилия в этом направлении. Однако в подобных вопросах интуиция оказывается особенно обманчивой, ибо зачастую задачи, разрешимые за полиномиальное время, очень незначительно отличаются от *NP*-полных задач. Например, мы уже видели, что задачи 3-ВЫПОЛНИМОСТЬ (3-ВЫП) и ТРЁХМЕРНОЕ СОЧЕТАНИЕ (3-С) *NP*-полны, а близкие задачи 2-ВЫПОЛНИМОСТЬ и ПАРОСОЧЕТАНИЕ разрешимы за полиномиальное время. На рис. 4.1. указано несколько других пар задач, одна из которых принадлежит классу P, а другая *NP*-полна. Наша интуиция, как правило, основана на опыте работы с близкими задачами. По этой причине мы серьёзно рискуем сбиться с правильного пути, если позволим себе, доверяясь интуиции, сосредоточиться на исследовании только одной возможности.

Таким образом, при анализе задач лучше пользоваться двусторонним подходом. Пытаясь с одной стороны доказать *NP*-полноту задачи, одновременно целесообразно искать полиномиальный алгоритм её решения. Конечно, в каждый данный момент мы выбираем направление исследований исходя из того, что представляется нам перспективнее, но нужно быть готовым сменить направление исследований. В действительности, по мере того как мы будем переходить от одного направления к другому и обратно, оба направления, взаимодействуя друг с другом, сливаются в единое целое. Неудачи в доказательстве *NP*-полноты задачи могут породить идею алгоритма её решения, а неудачи в построении алгоритма могут указать путь к доказательству *NP*-полноты задачи. Любые частичные результаты, полученные в процессе подобных исследований, особенно результаты о «нормальной форме» решений, могут оказаться в равной степени полезными как для доказательства *NP*-полноты задачи, так и для построения эффективного алгоритма её решения.

Само собой разумеется, что успешное применение на практике такого двустороннего подхода требует от исследователя мастерства как в отыскании доказательств *NP*-полноты, *так и* в построении полиномиальных алгоритмов. О методах доказательства *NP*-полноты было уже достаточно много сказано в гл. 3. С методами построения эффективных алгоритмов читатель может ознакомиться в стандартных книгах по построению алгоритмов, см. например, [9, 445]. В настоящей главе мы сосредоточим внимание на следующем вопросе: если доказана *NP*-полнота некоторой задачи (что бывает очень часто), как использовать рассмотренный выше двусторонний подход для более глубокого анализа этой задачи?

В разд. 4.1 обсуждается вопрос о том, каким образом можно глубже прозондировать сложность *NP*-полной задачи, изучая её подзадачи и пытаясь разграничить полиномиально разрешимые и *NP*-полные подзадачи. В разд. 4.2 рассматриваются подзадачи специального вида, часто заслуживающие внимание в тех случаях, когда в исходной задаче существенную роль играют целые числа. Такой подход приводит к понятиям «алгоритм с псевдополиномиальным временем работы» и «сильная *NP*-полнота», а также к дополнению списка «основных» *NP*-полных задач седьмой задачей.

Заканчивает главу разд. 4.3, в котором кратко обсуждается использование подзадач при изучении влияния на сложность задачи её индивидуальных параметров (а не только общей «длины входа»).

4.1. Анализ подзадач

Предположим, нам только что удалось доказать, что интересующая нас задача *NP*-полна. При этом мы получаем полные ответы на оба вопроса, с которых начинался анализ вопросов. Однако это лишь первый шаг на пути решения задачи. Задача, которой мы занимаемся, часто получается как идеализация менее привлекательной прикладной задачи, и некоторые детали, опущенные в процессе идеализации, могут изменить задачу в такой степени, что она станет разрешимой за полиномиальное время. Даже если это не так, у задачи могут иметься некоторые важные частные подзадачи, которые могут быть решены за полиномиальное время. Возможно также, что индивидуальные задачи, плохо поддающиеся решению, встречаются относительно редко и имеют легко выявляемые особенности, позволяющие заблаговременно опознать такие задачи. Эти возможности могут быть изучены с помощью анализа подзадач интересующей нас задачи.

Согласно принятому нами описанию массовых задач распознавания, каждая из них состоит из двух частей: множества D всех индивидуальных задач и множества Y индивидуальных задач из D с ответом «да». ***Подзадачей*** (или «частным случаем») задачи $\Pi = (D, Y)$ называется такая задача $\Pi' = (D', Y')$, что $D' \subseteq D$ и $Y' \subseteq Y \cap D'$. Другими словами, Π' есть подзадача задачи Π, если в ней сформулирован тот же вопрос, что и в Π, но только на некотором подмножестве множества индивидуальных задач из Π.

Таким образом, подзадачи возникают всякий раз, когда на множество индивидуальных задач налагаются дополнительные ограничения. В задачах из теории графов, например, можно ограничиваться индивидуальными задачами, в которых графы планарны, или двудольны, или ацикличны, или обладают некоторыми из этих свойств одновременно. В задачах, содержащих множества, можно ограничиться индивидуальными задачами, в которых мощность множеств не превосходит заданной величины, или задачами, в которых все элементы содержатся не более чем в ограниченном числе множеств. Можно потребовать, чтобы любые величины, сопоставляемые элементам множества, принадлежали некоторому ограниченному множеству допустимых значений. Из всех этих возможных задач для детального анализа обычно выбираются те, которые имеют известные приложения или в каком-то смысле являются «естественными» подзадачами, возникновение которых можно ожидать в приложениях.

Совершенно ясно, что, если даже задача Π является *NP*-полной, её подзадачи могут быть как *NP*-полными, так и полиномиально разрешимыми. Конечно, если задача Π принадлежит классу P, то любая её подзадача, индивидуальные задачи которой распознаваемы за полиномиальное время (а мы будем рассматривать только подзадачи, обладающие последним свойством), также должна лежать в P. Мы уже отмечали отличающиеся в этом отношении две подзадачи ВЫПОЛНИМОСТЬ – а именно задачи 3-ВЫПОЛНИМОСТЬ и 2-ВЫПОЛНИМОСТЬ. В предположении, что $P \neq NP$, можно считать, что подзадачи любой NP-полной задачи лежат по разные стороны воображаемой «границы», разделяющей полиномиально разрешимые и труднорешаемые задачи. При анализе задач наша цель состоит в выяснении, какие подзадачи по какую сторону этой границы лежат.

В действительности, по-видимому, лучше будет в каждый отдельный момент времени представлять себе «пограничную зону» между подзадачами, разрешимость которых за полиномиальное время уже *известна*, и подзадачами, для которых установлена NP-полнота. Подобная пограничная зона состоит из подзадач, NP-полнота которых остаётся под вопросом. На рис. 4.2 схематически изображено возможное «состояние знаний» о наборе подзадач задачи П.

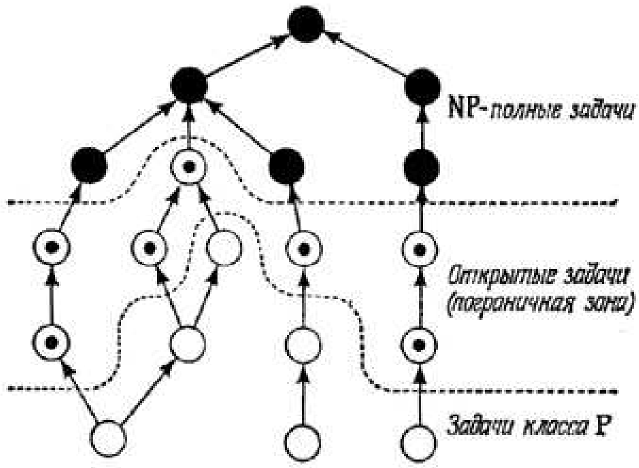


Рис. 4..1: *Одна из возможных ситуаций состояния знаний о подзадачах NP-полной задачи* П. *Задачи изображены кружками. Тёмные кружки соответствуют NP-полным задачам, светлые – задачам класса P, а кружки с точкой - «открытым» задачам. Стрелка от задачи* Π_1 *к задаче* Π_2 *указывает на то, что* Π_1 *есть подзадача* Π_2.

Если нам удаётся определить, что некоторая открытая задача лежит в P или NP-полна, то тем самым сужается пограничная зона и увеличивается изученная часть класса NP. Конечно, если задача неразрешима за полиномиальное время, нам никогда не удастся полностью исчерпать пограничную зону, кроме, возможно, случая, когда представляет интерес только ограниченное число подзадач. Даже когда мы ограничиваемся фиксированным, конечным набором подзадач, некоторые из них могут оказаться принадлежащими к группе промежуточных задач, которые ни NP-полны, ни принадлежат классу P (как мы уже отмечали, если $P \neq NP$, то такие задачи должны существовать). Тем не менее, когда мы выясняем место одной из задач, оставшейся открытой, можно считать, что мы приближаемся к воображаемой «пограничной линии».

Чтобы конкретизировать сказанное выше, рассмотрим задачу составления расписания выполнения заданий, имеющих равную длительность и подчинённых отношению предшествования (эта задача сама по себе представляет частный случай некоторых более общих задач теории расписаний).

РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ

УСЛОВИЕ. Задано множество T «заданий» (предполагается, что все они имеют длительность 1) и частичный порядок $\lessdot$ на множестве T, число «процессоров» m и общий директивный срок $D \in Z^+$.

ВОПРОС. Существует ли такое «расписание» $\sigma : T \to \{0, 1, \dots, D\}$, что для каждого $i \in \{0, 1, \dots, D\} |\{t \in T : \sigma(t) = i\}| \leqslant m$ и если $t \lessdot t'$, то $\sigma(t) < \sigma(t')$?

В качестве гипотетического «приложения» этой задачи рассмотрим следующую ситуацию. Предположим, что вы ассистент факультета прикладной математики Государственного университета и только что получили задание помочь поступившим первокурсникам составить план своих занятий на всё время обучения. Студенты представляют вам список всех курсов, которые они намереваются прослушать, и число m – максимальное число курсов, которые они могут слушать одновременно, D – число семестров обучения. Университет достаточно большой, так что каждый курс лекций читается каждый семестр и никакие два курса не пересекаются по времени. Однако некоторые курсы являются вспомогательными для других и поэтому должны быть прослушаны раньше последних ($t \lessdot t'$ означает, что t является вспомогательным для t'). Предположим, что вы решили написать программу для ЭВМ, которая, пользуясь этой информацией, составляла бы для каждого студента соответствующее расписание.

Задача РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ NP-полна [518], поэтому мало вероятно, что вам удастся построить полиномиальный алгоритм составления расписания в общем случае. Тем не менее, возможно, найдутся некоторые естественные ограничения, приемлемые для большинства студентов и облегчающие решение задачи. Поскольку работоспособность студентов ограничена, то вполне вероятно, что m можно ограничить сверху некоторым числом, например 6. Вспомогательные курсы лекций также могут удовлетворять дополнительным ограничениям. Многие из ваших студентов могли выбрать столь разнообразную программу обучения, что ни один из выбранных ими курсов лекций не потребует вспомогательных курсов (в этом случае условие порядка отсутствует). В некоторых случаях может оказаться, что для каждого данного курса лекций требуется только один «явный» вспомогательный курс, а все остальные курсы, вспомогательные для данного, являются вспомогательными также и для «явного» вспомогательного. В этом случае частичный порядок представляет собой древовидную структуру. Возможны и другие ограничения, но остановимся только на перечисленных. Таким образом, возникает семейство подзадач, представленное на рис. 4.3. На этом же рисунке приведены известные сведения о сложности этих подзадач.

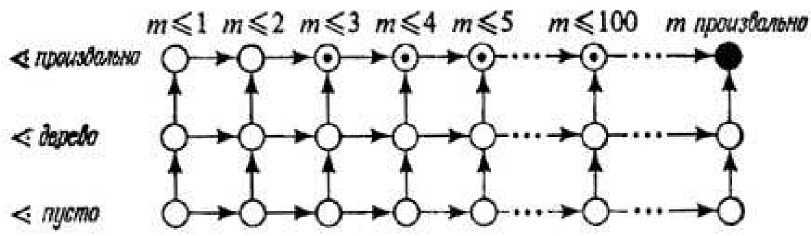


Рис. 4..2: *Современное состояние знаний о семействе подзадач задачи РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ (в обозначениях рис. 4.2).*

Заметим, что все возможности, указанные на рис. 4.2, включая пограничную зону, встречаются на рис. 4.3. В случае когда имеется подобная однородная иерархия подзадач, очень часто ту же самую информацию можно задать более кратко, указав «минимальную» *NP*-полную и «максимальную» разрешимую за полиномиальное время подзадачу. Если задано семейство C подзадач некоторой *NP*-полной задачи, то задача $\Pi \in C$ является (в настоящее время) *минимальной NP*-полной подзадачей, если известна *NP*-полнота задачи Π и у неё нет подзадачи Π', которая была бы *NP*-полна и принадлежала семейству C. Задача $\Pi \in C$ является (в настоящее время) *максимальной* полиномиально разрешимой подзадачей, если известна её принадлежность классу P и в семействе C нет задачи Π', которая была бы разрешима за полиномиальное время и содержала бы Π в качестве подзадачи. Аналогичным образом можно определить минимальную и максимальную открытые подзадачи семейства C.

Для класса подзадач задачи РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ, представленного на рис. 4.3., две подзадачи, определяемые ограничениями «$\lessdot$ – произвольно, $m \leqslant 2$» и «$\lessdot$ – дерево, m – произвольно» являются (в настоящее время) *минимальной NP*-полной подзадачей. Минимальная открытая подзадача определяется ограничениями «$\lessdot$ – произвольно, $m \leqslant 3$», а максимальной открытой подзадачи не существует, поскольку все задачи с ограничениями вида «$\lessdot$ – произвольно, $m \leqslant J$» открыты при всех целых $J \geqslant 3$.

Вполне естественным представляется изучение пограничной области задачи с помощью общей схемы двустороннего подхода, описанной выше. Мы по очереди рассматриваем подзадачи, представляющиеся наиболее вероятными кандидатами на принадлежность классу P, и подзадачи, кажущиеся

NP-полными. Используя в качестве отправных точек подзадачи, очевидным образом разрешимые за полиномиальное время, и подзадачи, *NP*-полнота которых следует непосредственно из *NP*-полноты общей задачи, мы постепенно увеличиваем множество индивидуальных задач первой подзадачи и уменьшаем множество индивидуальных задач второй. В отличие от случая, когда изучается одна отдельная задача с переходами от построения алгоритма к доказательству *NP*-полноты, при анализе подзадач у нас появляется возможность изменять также и сами задачи, от более ограничительных к менее ограничительным.

Техника, используемая для доказательства *NP*-полноты подзадач, по существу, аналогична описанной в гл. 3 технике доказательства *NP*-полноты изолированных задач. Однако имеется одно весьма важное различие. Когда мы пытаемся доказать *NP*-полноту подзадачи, нам уже известно доказательство *NP*-полноты некоторой обобщённой версии этой подзадачи. Это даёт нам хорошего кандидата на «известную» *NP*-полную задачу (чем можно воспользоваться в искомом доказательстве), а также доказательство *NP*-полноты, которое можно было бы модифицировать, чтобы получить соответствующее доказательство для рассматриваемой подзадачи. Хотя это и не всегда облегчит достижение нашей цели, но, по крайней мере в отличие от доказательства *NP*-полноты изолированной задачи, нам будет от чего оттолкнуться.

Преимущества такого подхода можно удачно проиллюстрировать на некоторых хорошо известных задачах из теории графов. Поскольку среди *NP*-полных задач очень часто встречаются задачи из теории графов, а ограничения, которые мы будем рассматривать для таких задач, часто оказываются существенными, то в действительности различные приёмы, используемые нами для доказательства результатов об *NP*-полноте при этих ограничениях, заслуживают рассмотрения сами по себе.

В дальнейшем мы будем обсуждать много разнообразных задач из теории графов, а в качестве первого примера возьмём следующую задачу (уже встречавшуюся в гл. 3) о «раскраске графа»:

РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Верно ли, что G раскрашиваем в 3 цвета? Другими словами: «Существует ли такая функция $f : V \to \{1, 2, 3\}$, что $f(u) \neq f(v)$, если $\{u, v\} \in E$»?

Эта задача связана со знаменитой проблемой четырёх красок (недавно решённой Аппелем и Хакеном [19, 20] и возникающей в связи с задачами теории расписаний и разбиений). Задача сама по себе представляет собой частный случай задачи РАСКРАШИВАЕМОСТЬ ГРАФА В K ЦВЕТОВ, в которой f принимает значение в множестве $\{1, \ldots, K\}$, а K входит в условие задачи. *NP*-полнота задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА была доказана *Стокмейером* [496] (доказательство опубликовано также в работе [157]).

Первое из рассматриваемых нами ограничений состоит в ограничении максимальной степени вершин (*степенью* вершины называется число содержащих её рёбер). Большинство задач из теории графов может быть решено за полиномиальное время, если максимум степеней вершин графа достаточно мал. Например, если потребовать, чтобы все вершины графа имели степень не более 2, то задачи ГАМИЛЬТОНОВ ЦИКЛ, ВЕРШИННОЕ ПОКРЫТИЕ и РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА (а также почти любая задача из теории графов, которую можно себе вообразить) тривиально решаются за полиномиальное время. Таким образом, возникает вопрос: «каково самое сильное ограничение на степени вершин, при котором задача всё ещё остаётся *NP*-полной»?

Задача КЛИКА – одна из задач для которой никакое ограничение на степени вершин не сохраняет свойства *NP*-полноты. Это объясняется тем, что если степени вершин ограничены числом D, то граф не может содержать клику на более чем $D+1$ вершинах. Следовательно, наибольшая клика может быть найдена перебором всех подмножеств вершин, состоящих не более чем из $D+1$ элементов, а поскольку D фиксировано, то число таких подмножеств ограничено полиномом. Заметим, что хотя (в предположении $P \neq NP$) это соображение предупреждает наши попытки доказывать *NP*-полноту ограниченной версии задачи КЛИКА, алгоритм, который можно построить на его основе при больших D, не представляет практического интереса.

Для многих других задач из теории графов *имеются* *NP*-полные подзадачи с ограниченной степенью вершин. На рис. 4.4 представлены некоторые результаты этого типа. Отметим, что все границы являются наилучшими возможными (в предположении $P \neq NP$), поскольку их уменьшение всего лишь на 1 приводит к полиномиально разрешимой подзадаче. В действительности задачи становятся тривиальными. (Для задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА теорема Брукса [56] утверждает, что связный граф с максимальной степенью вершин, равной 3, раскрашиваем в 3 цвета тогда и только тогда, когда он не является полным графом на четырёх вершинах). (Последнее условие легко проверяется).

	В P при $D \leqslant$	*NP*-полная при $D \geqslant$
ВЕРШИННОЕ ПОКРЫТИЕ	2	2
ГАМИЛЬТОНОВ ЦИКЛ	2	3
РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА	3	4
МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ	2	3

Рис. 4.4. Классификация (относительно разрешимости за полиномиальное время и NP-полноты) подзадач, получаемых рассмотрением индивидуальных задач, у которых степень вершин не превосходит D.

Каждый из этих результатов об *NP*-полноте подзадач на графах с ограниченной степенью вершин может быть получен из общей задачи с помощью метода локальной замены. Основную идею, которая заключается в «замене вершин», мы проиллюстрируем на примере задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА.

Теорема 4.1. *Задача РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА для графов, все вершины которых имеют степень не более 4, остаётся NP-полной.*

Доказательство 1. Принадлежность ограниченной задачи классу *NP* следует немедленно из аналогичного результата для общей задачи. Поэтому предположим, что $G = (V, E)$ – произвольная индивидуальная задача общей задачи. Необходимо построить соответствующий граф $G' = (V', E')$, не имеющий вершин степени более 4 и раскрашиваемый в 3 цвета тогда и только тогда, когда граф G раскрашиваем в 3 цвета.

Приём «замена вершин» основан на использовании представленного на рис. 4.5(а) графа H_3 на восьми вершинах. Этот граф имеет три «разъёма»,

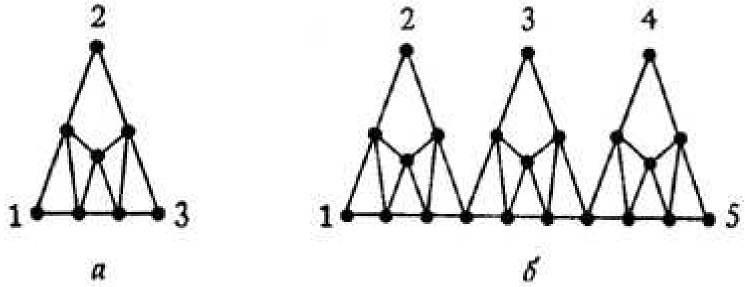


Рис. 4..3: *Граф H_3 и граф H_5 (образованный тремя экземплярами графа H_3), используемые для доказательства NP-полноты задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА с ограниченными степенями вершин.*

помеченные на этом рисунке числами 1, 2 и 3. При $k \geqslant 4$ вершина, инцидентная k рёбрам, заменяется графом H_k, имеющим k разъёмов, который получается из графа H_{k-1} подсоединением экземпляра графа H_3 (причём к $(k-1)$-му разъёму графа H_{k-1} подсоединяется первый разъём графа H_3). Разъёмы графа H_{k-1} – вершины степени два. Все разъёмы графа H_{k-1} сохраняют свои метки, а разъёмы с номерами 2 и 3 подсоединяемого графа H_3 получают метки $k-1$ и k соответственно. На рис. 4.5(б) представлен граф H_5.

Легко видеть, что для всех $k \geqslant 3$ имеют место следующие утверждения:

(1) Граф H_k имеет $7(k-2)+1$ вершин, включая k помеченных разъёмов.

(2) В графе H_k нет вершин степени более 4.

(3) Степень каждого разъёма равна 2.

(4) Граф H_k раскрашиваем в 3 цвета, но не раскрашиваем в 2 цвета, а любая его раскраска в 3 цвета сопоставляет всем разъёмам один и тот же цвет.

Перенумеруем произвольным образом r вершин $v_1, v_2, \ldots, v_r$ заданного графа G, степени которых превосходят 4. Теперь последовательность графов

$$G = G_0, G_1, G_2, \ldots, G_r = G'$$

строится следующим образом. Граф G_i, $1 \leqslant i \leqslant r$, строится по графу G_{i-1}. Пусть d – степень вершины v_i в графе G_{i-1}, а $\{u_1, v_i\}, \{u_2, v_i\}, \ldots, \{u_d, v_i\}$ – рёбра, содержащие эту вершину. Для построения графа G_i из графа G_{i-1} удаляем вершину v_i, заменяя её экземпляром графа H_d, и при этом каждое ребро $\{u_j, v_i\}$ заменяется ребром, соединяющим u_j с j-м разъёмом графа H_d.

Из построения и перечисленных выше фактов вытекает, что при $0 \leqslant k \leqslant r$ граф G_k имеет $r - k$ вершин степени более 4 и граф G_k раскрашиваем в три цвета тогда и только тогда, когда граф G раскрашиваем в три цвета. Отсюда следует, что граф $G' = G_r$ обладает необходимым свойством.

■

Для других задач используются иные замены вершин и подчас требуется большая изобретательность, чтобы отыскать соответствующую замену, сохраняющую все нужные свойства. Однако поскольку на практике такие ограничения встречаются достаточно часто (например, ограничение на число входов и ветвлений выходов в логических схемам), то имеет смысл изучить их влияние на сложность любой общей задачи теории графов.

Другое общее ограничение для задач из теории графов связано с планарностью графов. Граф называется *планарным*, если его можно вложить в плоскость посредством отождествления каждой вершины с точкой плоскости, а ребра – с линией, соединяющей его конечные точки (причём никакие две линии не должны иметь общих точек, кроме, быть может, концов). Например при составлении карт и при конструировании печатных схем возникают планарные графы, поэтому вполне естественно рассмотреть влияние планарности на сложность задачи. КЛИКА снова даёт нам пример задачи, которая при наложении такого ограничения упрощается, ибо планарный граф не может содержать полного подграфа более чем на четырёх вершинах. Более интересен пример следующей задачи:

МАКСИМАЛЬНЫЙ РАЗРЕЗ

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $\omega(e) \in Z^+$ для каждого ребра $e \in E$, а также целое положительное число K.

ВОПРОС. Можно ли разбить множество V на такие два непересекающихся подмножества V_1 и V_2, чтобы сумма весов рёбер из E, имеющих концы в разных подмножествах, была не меньше K?

Для произвольных графов эта задача *NP*-полна, даже если потребовать, чтобы веса всех рёбер были равны [157]. Однако в работах [406, 195] показано, как теория паросочетаний может быть использована для решения этой задачи на планарных графах без каких-либо ограничений на веса рёбер.

С другой стороны, многие задачи на графах (например, все задачи, обсуждавшиеся в гл. 3) остаются *NP*-полными даже для планарных графов.

К доказательству результатов подобного типа имеется два подхода. Первый из них состоит в том, чтобы найти задачу, для которой *NP*-полнота на планарных графах уже известна, и использовать сводимость, сохраняющую планарность. Второй, более общий приём состоит в применении к исходной задаче метода локальной замены, для чего конструируется «переплетение», которое можно было бы использовать вместо любого переплетения рёбер, возникающего при вложении графа (не обязательно планарного). Для иллюстрации этого приёма опять вернёмся к задаче РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА.

Теорема 4.2. *Задача РАСКРАШИВАЕМОСТЬ ПЛАНАРНОГО ГРАФА В 3 ЦВЕТА NP-полна.*

Доказательство 2. Принадлежность этой задачи классу *NP* очевидна (планарность графа можно распознать за полиномиальное время, применив, например, линейный по времени алгоритм Хопкрофта и Тарьяна [213]). Поэтому предположим, что $G = (V, E)$ – произвольная индивидуальная задача РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА. Требуется указать способ построения соответствующего планарного графа $G' = (V', E')$, такого, что G' раскрашиваем в 3 цвета тогда и только тогда, когда G раскрашиваем в 3 цвета.

«Переплетение», используемое в доказательстве,– это граф, представленный на рис. 4.6. У него разъёмы x, x', y и y' помечены. Это переплетение было предложено М.Дж.Фишером, оно проще переплетения, использовавшегося в исходном доказательстве Стокмейера [496]. Граф H имеет 13

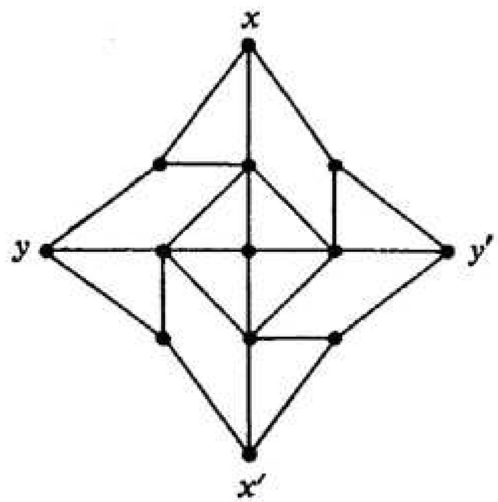


Рис. 4..4: *Переплетение H, используемое в док-ве NP-полноты задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА для планарных графов.*

вершин, 24 ребра и обладает следующими свойствами (проверку которых мы оставляем читателю):

1. Любая раскраска графа H в три цвета удовлетворяет условиям $f(x) = f(x')$ и $f(y) = f(y')$.

2. Существуют такие раскраски f_1 и f_2 графа H в три цвета, что выполняются свойства

$f_1(x) = f_1(x') = f_1(y) = f_1(y')$.

$f_2(x) = f_2(x') \neq f_2(y) = f_2(y')$. Граф G' строится по G следующим образом:

(а) Вкладываем граф G в плоскость, позволяя рёбрам пересекаться, но не допуская прохождения ребра через вершину, отличную от концов, и не разрешая пересечения более чем двух рёбер в одной точке (за исключением вершин). Это может быть легко сделано за полиномиальное время.

(б) Для каждого ребра $\{u, v\} \in E$ назовём его представление на плоскости «{u, v}-линией». Для каждой линии, «пересекаемой» другими линиями, добавим новые вершины по одной между каждым концом и ближайшей к нему точкой пересечения и по одной между каждой парой соседних точек пересечения – см. рис. 4.7 (а) и (б).

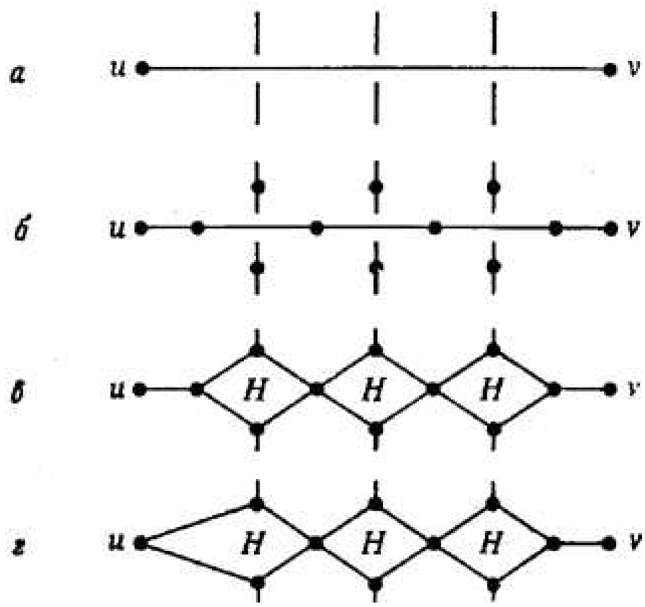


Рис. 4..5: *Построение по заданному графу G планарного графа G', использующее переплетение H, представленное на рис. 4.6 и сохраняющее свойство раскрашиваемости в 3 цвета.*

(в) Заменяем каждую точку пересечения линий в графе экземпляром графа H, отождествляя разъёмы x и x' с ближайшими двумя точками, на одной из пересекающихся линий, а разъёмы y и y' с ближайшими двумя точками на другой – см. рис. 4.7 (в).

(г) Для каждого ребра $\{u, v\} \in E$ выберем одну из его *выделенных конечных точек* и сольём её с ближайшей *новой* точкой на $\{u, v\}$-линии – см. рис. 4.7 (г). Если на $\{u, v\}$-линии нет новых точек, то слияний не происходит и все исходные вершины (множества V) сохраняются неизменными. Ребро между другой концевой точкой $\{u, v\}$-линии и ближайшей к ней новой точкой на $\{u, v\}$-линии будет называться *оперантным ребром* $\{u, v\}$-линии.

На этом описание графа G заканчивается. Нетрудно видеть, что граф G' планарный и может быть построен за полиномиальное время. Остаётся показать, что G' раскрашиваем в три цвета тогда и только тогда, когда G раскрашиваем в три цвета.

Предположим, что $f : V' \to \{1, 2, 3\}$ – произвольное раскрашивание графа G' в три цвета. Утверждается, что ограничение f на V есть раскрашивание в 3 цвета графа G. Предположим противное. Тогда должно существовать ребро $\{u, v\} \in E$, такое, что $f(u) = f(v)$. Рассмотрим $\{u, v\}$-линию в графе G' и (без ограничения общности) предположим, что u – выделенный конец этого ребра, выбранный на шаге (г) построения. В силу свойства 1 графа H все новые точки $\{u, v\}$-линии должны быть окрашены в тот же «цвет», что и вершина u. Следовательно, оба конца оперантного ребра этой линии должны быть окрашены в одинаковый цвет, а это противоречит предположению, что f являлось раскрашиванием графа G' в 3 цвета.

Обратно, предположим, что $f : V \to \{1, 2, 3\}$ – произвольное раскрашивание графа G' в три цвета. Это раскрашивание может быть продолжено до раскрашивания в три цвета графа G' следующим образом. Для каждого ребра $\{u, v\} \in E$ окрасим каждую новую точку $\{u, v\}$-линии в цвет $f(u)$, где u – выделенная вершина этой линии. Это обеспечивает то, что оба конца каждого оперантного ребра оказываются окрашенными в разные цвета (поскольку $f(u) \neq f(v)$). В силу свойства 3 графа H, эта частичная раскраска графа G' может быть продолжена до раскрашивания графа G' в 3 цвета с помощью выбора соответствующей окраски вершин каждого из переплетений. Отсюда вытекает требуемый результат.

■

Списки задач из приложения ознакомят читателя с большим числом других результатов об NP-полноте ограниченных версий задач из теории графов. В этих списках мы пытались дать как можно больше информации о сложности различных подзадач каждой задачи. Списками можно воспользоваться в качестве одного из источников возможных ограничений, которые для данной задачи можно подвергнуть анализу. Другие ограничения возникнут из контекста, в котором формулируются задачи. Индивидуальные задачи, возникающие в конкретных приложениях, часто будут удовлетворять специальным ограничениям. Эти ограничения могут оказать влияние на сложность задачи, хотя они на первый взгляд могут показаться несущественными. В следующем разделе мы обсудим ограничения специального вида, часто представляющие интерес в задачах с числовыми параметрами.

4.2. Задачи с числовыми параметрами и сильная NP-полнота

Необходимость анализа подзадач NP-полной задачи особенно ярко проявляется в задачах с числовыми параметрами. Причины этого явления можно проиллюстрировать на нижеследующем примере применения метода «динамического программирования» для решения задачи РАЗБИЕНИЕ.

j i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	T	T	F	F	F	F	F	F	F	F	F	F	F	F
2	T	T	F	F	F	F	F	F	F	T	T	F	F	F
3	T	T	F	F	F	T	T	F	F	T	T	F	F	F
4	T	T	F	T	T	T	T	F	T	T	T	F	T	T
5	T	T	F	T	T	T	T	F	T	T	T	T	T	T

Рис. 4.8. Таблица значений $t(i,j)$ для индивидуальной задачи РАЗБИЕНИЕ, которая определяется множеством $A = \{a_1, a_2, a_3, a_4, a_5\}$ и весами $s(a_1) = 1, s(a_2) = 9, s(a_3) = 5, s(a_4) = 3$ и $s(a_5) = 8$. Рассматриваемой индивидуальной задаче соответствует ответ «да», поскольку $t(5,13) = T$ в силу того, что $s(a_1) + s(a_2) + s(a_4) = 13 = 26/2$.

Пусть множество $A = \{a_1, a_2, \dots, a_n\}$ и размеры его элементов $s(a_1), s(a_2), \dots, s(a_n) \in Z^+$ определяют произвольную индивидуальную задачу РАЗБИЕНИЕ. Положим $B = \sum_{a\in A} s(a)$. Если B нечётно, то, очевидно, не существует такого подмножества $A' \subseteq A$, что

$$\sum_{a\in A} s(a) \subseteq \sum_{A\setminus A'} s(a),$$

и, следовательно, рассматриваемой индивидуальной задаче соответствует ответ «нет». Если же B чётно, то пусть $t(i,j)$ (где $1 \leqslant i \leqslant n, 0 \leqslant j \leqslant B/2$) обозначает значение истинности утверждения: «существует такое подмножество множества $\{a_1, a_2, \dots, a_i\}$, что сумма размеров его элементов равна в точности j». Значения $t(i,j)$ можно расположить в виде таблицы, как это сделано на рис. 4.8.

«Изюминка» предлагаемого метода заключается в очень простой процедуре, которой можно воспользоваться для заполнения таблицы. Процедура позволяет заполнить таблицу последовательно, строчка за строчкой сверху вниз. Совершенно ясно, что в первой строке $t(i,j) = T$ тогда и только тогда, когда либо $j = 0$, либо $j = s(a_1)$. Каждая следующая строка заполняется с использованием значений, расположенных в предыдущей строке. Для $1 < j \leqslant n, 0 \leqslant j \leqslant B/2$, равенство $t(i,j) = T$ имеет место только в двух случаях: либо если $t(i, j-1) = T$, либо если $s(a_i) \leqslant j$ и $t(i-1, j - s(a_i)) = T$.

После того как вся таблица будет заполнена, рассматриваемая индивидуальная задача РАЗБИЕНИЕ будет решена, поскольку ей соответствует ответ «да» тогда и только тогда, когда $t(n, B/2) = T$.

У читателя не должно возникнуть трудностей при написании итеративного алгоритма заполнения таблицы согласно описанному выше способу; время работы алгоритма ограничено полиномом низкой степени от числа клеток таблицы (т.е. полиномом от nB). На первый взгляд может даже показаться, что это даёт полиномиальный алгоритм решения задачи РАЗБИЕНИЕ, откуда следует, что $P = NP$ и в результате чего настоящая книга вообще становится ненужной. Однако это, конечно, не так. Объяснение можно найти, вспомнив о требовании «сжатости» разумных схем кодирования задач, при которых каждое целое число $s(a_i)$ должно быть представлено на входе словом длины порядка $O(\log s(a_i))$. Следовательно, длина записи всей индивидуальной задачи РАЗБИЕНИЕ будет величиной порядка $O(n \log B)$, а nB *не* ограничено никаким полиномом от этой величины. Следовательно, описанный выше алгоритм не позволяет решить задачу РАЗБИЕНИЕ за полиномиальное время.

Тем не менее наличие такого алгоритма показывает, что NP-полнота задачи РАЗБИЕНИЕ (*а также* её предполагаемая труднорешаемость) в значительной степени зависит от того обстоятельства, что на входе могут присутствовать чрезвычайно большие числа. Если бы на величину таких чисел заранее было наложено какое-либо ограничение, например в виде полинома от $Length[I]$, то приведённый выше алгоритм для таким образом ограниченной задачи был бы полиномиальным. (В дальнейшем для выделения алгоритмов, обладающих этим свойством, мы введём термин «алгоритм с всевдополиномиальным временем работы»). Естественно ожидать, что подобное ограничение выполнено во многих задачах, встречающихся на практике.

Например, в задачах теории расписаний (где числа представляют собой длительности заданий) появление слишком больших чисел мало вероятно, поскольку мы собираемся выполнять задания реально, и если какое-то из них требует чрезвычайно большого времени, то мы просто будем не в состоянии его закончить. В других задачах, где числа представляют собой эмпирически измеренные величины, доступная точность измерения ограничивает пределы изменения чисел, к которым наш алгоритм должен быть применим.

Более того, алгоритм с псевдополиномиальным временем работы может быть полезен даже в тех случаях, когда нельзя надеяться на естественное ограничение величины входных чисел. Такие алгоритмы будут работать «экспоненциально долго» только для тех индивидуальных задач, в которых имеются «экспоненциально большие» числа. Однако может оказаться, что в интересующих нас приложениях подобные индивидуальные задачи встречаются редко. Если это так, то алгоритм этого типа может служить нашим целям почти столь же хорошо, как и полиномиальный по времени алгоритм.

Таким образом, отыскание псевдополиномиального алгоритма для решения *NP*-полной задачи с числовыми параметрами – вопрос достойный исследования. В дальнейшем мы увидим, что не все числовые задачи в этом отношении подобны задаче РАЗБИЕНИЕ. С помощью теории *NP*-полноты можно показать, что если $P \neq NP$, то для решения некоторых из них не может существовать даже псевдополиномиального алгоритма. В разд. 4.2.1 вводится новая терминология, а также закладывается фундамент для доказательства подобных результатов о «сильной» *NP*-полноте. В разд. 4.2.2 иллюстрируется техника доказательств и приводится седьмая из числа «основных» *NP*-полная задача.

4.2.1. Некоторые дополнительные определения

В следующих ниже определениях участвуют подзадачи, получаемые при наложении ограничений на величины чисел, входящих в индивидуальную задачу. Эти ограничения будут формулироваться с помощью двух не зависящих от кодирования функций $Length : D_\Pi \to Z^+$ и $Max : D_\Pi \to Z^+$, которые будут связываться с каждой задачей распознавания П. Хотя теоретически эти две функции могут быть совершенно произвольными (так же как схемы кодирования), значимость определяемых с их помощью понятий будет зависеть от того, в какой мере эти функции адекватно отражают следующий вкладываемый в них смысл. Как говорилось в разд. 2.1, назначение функции *Length* состоит в сопоставлении любой индивидуальной задаче I некоторого целого числа $Length[I]$, соответствующего числу символов, используемых для описания I при некоторой разумной схеме кодирования задачи П. Назначение функции *Max* (не обсуждавшейся ранее) состоит в сопоставлении любой индивидуальной задаче I целого числа $Max[I]$, соответствующего величине максимального числа в I.

Типы результатов, которые мы будем доказывать в дальнейшем, будут носить достаточно общий характер и будут справедливы для достаточно широкого класса «полиномиально эквивалентных» функций *Length* и *Max*. Будем говорить, что две функции длины записи *Length* и *Length'* для задачи П *полиномиально эквивалентны*, если существуют такие полиномы p и p', что для всех индивидуальных задач $I \in G_\Pi$ выполнены соотношения

$$Length(I) \leqslant p'(Length'[I]);$$
$$Length'(I) \leqslant p\,(Length[I]).$$

Будем говорить, что пара функций $(Length, Max)$ *полиномиально эквивалентна* паре функций $(Length', Max')$, если, во-первых, функции *Length* и *Lenth'* эквивалентны в смысле данного выше определения и, во-вторых, существуют такие полиномы от двух переменных q и q', что для всех $I \in D_\Pi$ выполнены соотношения

$$Max[I] \leqslant q'(Max[I], Length'[I]);$$
$$Max'[I] \leqslant q\,(Max[I], Length[I]).$$

Все формулируемые ниже результаты будут справедливы для любых функций $Length$ и Max, полиномиально эквивалентных тем функциям, которыми мы пользуемся.

В качестве примера рассмотрим задачу РАЗБИЕНИЕ, в которой индивидуальная задача I определяется конечным множеством A и размерами $s(a) \in Z^+$ каждого элемента $a \in A$. Любая из приводимых ниже функций подошла ба в качестве функции $Length$ для задачи РАЗБИЕНИЕ:

$$Length[I] = |A| + \sum_{a \in A} \lceil \log_2 s(a) \rceil;$$

$$Length[I] = |A| + \max\{\lceil \log_2 s(a) \rceil : a \in A\};$$

$$Length[I] = |A| \cdot \left\lceil \log_2 \sum_{a \in A} s(a) \right\rceil .$$

пар функций $Length$ и Max, получаемых комбинацией функций подошла бы в качестве функции Max для задачи РАЗБИЕНИЕ:

$$Max[I] = \max\{s(a) : a \in A\};$$

$$Max[I] = \sum_{a \in A} s(a);$$

$$Max[I] = \left\lceil \left(\sum_{a \in A} s(a) \right) \Big/ |A| \right\rceil .$$

Мы оставляем читателю проверку того, что любая из девяти пар функций $Length$ и Max, получаемых комбинацией функций из приведённых двух списков, полиномиально эквивалентна любой другой такой же паре функций.

Неопределённость, допускаемая нами при выборе функций $Length$ и Max, позволит избежать явного их указания для задачи Π, поскольку их с достаточной точностью можно восстановить по описанию условия задачи. Подходящая функция $Length$ определяется понятием «разумная схема кодирования» задачи, а оно уточняется при описании условия задачи посредством стандартных соглашений, принятых в разд. 2.1. Подходящая функция Max возникает при указании того, что некоторые объекты задачи представляют собой числа (в отличие от множеств, последовательностей, графов, элементов с именами и т.д.). Обычно это будут целые числа, а более сложное «число», входящее в описание индивидуальной задачи, будет рассматриваться как образование, полученное из одного или нескольких отдельных целых чисел, аналогично рассмотренному выше случаю рациональных чисел. Примем следующее соглашение: в качестве $Max[I]$ будет выбираться либо величина наибольшего числа, входящего в I, либо 0, если в I вообще не входят целые числа.

От функции *Length* и *Max* будет требоваться выполнение ещё одного, последнего условия. Оно заключается в том, что при любой фиксированной разумной схеме кодирования задачи П должны существовать две ДМТ-программы с полиномиальным временем работы, которые, принимая на входе код каждой индивидуальной задачи $I \in D_\Pi$, выдают на выходе двоичные записи величин $Length[I]$ и $Max[I]$. Это свойство необходимо нам лишь по той причине, что мы будем рассматривать ограничения на индивидуальные задачи, формируемые с помощью $Length[I]$ и $Max[I]$, и нам нужно уметь распознавать коды индивидуальных задач, удовлетворяющие этим ограничениям. Любые естественно возникающие функции $Length[I]$ и $Max[I]$ наверняка обладают этим свойством.

Нижеследующие определения предполагают, что с любой задачей распознавания П связаны функции *Length* и *Max*, для которых выполняются перечисленные выше требования. Формально говоря, для рассуждения на уровне языков необходимо также, чтобы для каждой задачи П была предъявлена некоторая схема кодирования. Однако гораздо удобнее формулировать определения на уровне задач без этой оговорки, действуя согласно стандартным предположениям об использовании разумных схем кодирования. У читателя не должно возникнуть трудностей в восполнении всех деталей при полной формализации этих определений на уровне языков, и будет более естественно и более информативно продолжать наши рассуждения на уровне задач.

Алгоритм решения задачи П будет называться *псевдополиномиальным по времени алгоритмом* (или просто псевдополиномиальным алгоритмом), если его временна́я функция ограничена сверху полиномом от двух аргументов $Length[I]$ и $Max[I]$. По определению любой полиномиальный по времени алгоритм является также псевдополиномиальным по времени алгоритмом, поскольку время его работы ограничено полиномом от одного аргумента – $Length[I]$. Однако мы уже ознакомились с примером псевдополиномиального, но не полиномиального алгоритма для задачи РАЗБИЕНИЕ. Этот пример показывает, что, несмотря на то, что *NP*-полнота задачи П исключает возможность её решения полиномиальным алгоритмом (при условии $P \neq NP$), *NP*-полнота не исключает возможности решения задачи П псевдополиномиальным алгоритмом.

Точнее говоря, *NP*-полнота задачи *не обязательно* исключает возможность её решения псевдополиномиальным алгоритмом. Многие из рассмотренных нами задач распознавания обладали тем свойством, что величина $Max[I]$ ограничена полиномом от $Length[I]$, и, следовательно, для этих задач нет различия между полиномиальными и псевдополиномиальными алгоритмами. Например, единственный числовой параметр, входящий в условие задачи КЛИКА,– это граница J, которая не может быть больше числа вершин, заданного в условии графа. Задача ВЫПОЛНИМОСТЬ вообще не содержит чисел, за исключением лишь индексов у литералов и дизъюнкций, которыми можно пренебречь, поскольку они фактически играют роль «имён», а не «чисел». (Принятые нами соглашения о схемах кодирования гарантируют, что такие числовые «имена» всегда будут ограничены полиномом от $Length[I]$.) Интересующие нас здесь вопросы не имеют отношения к задачам такого типа. Выделим класс задач, к которым эти вопросы *имеют отношение.* Будем называть задачу Π задачей с *числовыми параметрами*, если не существует такого полинома p, что $Max[I] \leqslant p(Length[I])$ для всех $I \in D_\Pi$. Среди шести основных *NP*-полных задач единственной задачей с числовыми параметрами является задача РАЗБИЕНИЕ.

Непосредственно на основе данных определений можно сделать следующее замечание:

Замечание 4.1. *Если NP-полная задача* Π *не является задачей с числовыми параметрами, то* Π *не может быть решена псевдополиномиальным алгоритмом (если* $P \neq NP$*).*

Таким образом, в предположении $P \neq NP$ *NP*-полные задачи, для которых имеется потенциальная возможность решения псевдополиномиальным алгоритмом,– это задачи с числовыми параметрами.

Для произвольной задачи распознавания Π и полинома p (с целыми коэффициентами) обозначим через Π_p подзадачу, получаемую из Π рассмотрением только тех индивидуальных задач I, для которых выполнено соотношение $Max[I] \leqslant p(Length[I])$. Задача Π_p, таким образом, не является задачей с числовыми параметрами. Кроме того, если задача Π разрешима псевдополиномиальным алгоритмом, то Π_p должна быть разрешима полиномиальным алгоритмом. Если задано входное слово x, то достаточно проверить, что x кодирует индивидуальную задачу I, для которой $Max[I] \leqslant p(Length[I])$, и если это так, то применить к I псевдополиномиальный алгоритм решения задачи Π. В силу сделанных предположений, величины $Max[I]$ и $Length[I]$ могут быть вычислены за полиномиальное время, поэтому требуемое неравенство может быть проверено за полиномиальное время. По определению псевдополиномиального алгоритма для задачи Π, он будет полиномиальным для тех индивидуальных задач из Π, для которых выполняется указанное неравенство. Эти соображения приводят нас к следующему определению.

Задачу Π назовём *NP-полной в сильном смысле*, если Π принадлежит *NP* и существует такой полином p с целыми коэффициентами, что задача Π_p является *NP*-полной. В частности, если задача Π *NP*-полна и не является задачей с числовыми параметрами, то задача Π автоматически *NP*-полна в сильном смысле.

Таким образом, получаем обобщение замечания 4.1.

Замечание 4.2. *Если задача* Π *NP-полна в сильном смысле, то она не может быть решена псевдополиномиальным алгоритмом (если* $P \neq NP$*).*

Благодаря этому замечанию мы получаем средства для применения теории *NP*-полноты к вопросам существования псевдополиномиальных алгоритмов. Задача РАЗБИЕНИЕ не может быть *NP*-полной в сильном смысле, поскольку, как нам уже известно, она может быть решена псевдополиномиальным алгоритмом. Однако у нас ещё не было примеров задач с числовыми параметрами, *NP*-полных в сильном смысле. Этот пробел будет восполнен в следующем разделе, в котором будет показано, как можно получать результаты о сильной NP-полноте.

4.2.2. Доказательство результатов о сильной NP-полноте

Наиболее прямой путь доказательства сильной NP-полноты задачи Π с числовыми параметрами заключается просто в доказательстве того, что для некоторого конкретного полинома p задача Π_p является NP-полной. Например, задача КОММИВОЯЖЕР (КМ), определённая в разд. 2.1, представляет собой задачу с числовыми параметрами, поскольку ни расстояния между городами $d(i,j)$, ни граница B в ней не ограничены. Мы доказали NP-полноту задачи КМ, сведя к ней задачу ГАМИЛЬТОНОВ ЦИКЛ. Более того, при этом сведении получаются только такие индивидуальные задачи из КМ, в которых расстояния между городами равны либо 1, либо 2, а граница B равна числу городов m. Таким образом, если взять в качестве $Max[I]$ максимум из границы B и наибольшего расстояния между городами и положить

$$Length[I] = m + \lceil \log_2 B \rceil + \sum_{i,j} \lceil \log_2 d(i,j) \rceil,$$

то все индивидуальные задачи, получаемые в описанной выше сводимости, будут удовлетворять ограничению

$$Max[I] \leqslant Length[I].$$

Другими словами, эта сводимость фактически показывает, что подзадача задачи КМ, состоящая из всех индивидуальных задач, для которых выполняется указанное неравенство, сама является NP-полной. Отсюда следует, что задача КОММИВОЯЖЕР NP-полна в сильном смысле.

В противоположность рассмотренному примеру доказательства NP-полноты задач РЮКЗАК, РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ и УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ, приведённых в разд. 3.2, оставляют открытым вопрос о возможности решения перечисленных задач псевдополиномиальными алгоритмами. Оказывается, что задачу РЮКЗАК *можно* решить за псевдополиномиальное время с помощью метода динамического программирования. Этот метод, аналогичный методу, использовавшемуся при решении задачи РАЗБИЕНИЕ, изложен в работе [98]. Все известные нам алгоритмы с псевдополиномиальным временем работы основаны на этом методе. Пример использования этого метода читатель может найти в работах [221, 328, 332, 465].

Задачи РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ и УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ, напротив, NP-полны в сильном смысле. Для доказательства этого утверждения полезно иметь задачу с числовыми параметрами, NP-полную в сильном смысле и в определённом отношении «более числовую», чем любая из задач, рассмотренных до сих пор. Этой задачей будет седьмая «основная» *NP*-полная задача 3-РАЗБИЕНИЕ, определяемая следующим образом:

3-РАЗБИЕНИЕ

УСЛОВИЕ. Заданы множество A, состоящее из $3m$ элементов, граница $B \in Z^+$ и «размеры» $s(a) \in Z^+$ всех элементов, $a \in A$, причём $B/4 < s(a) < B/2$ и $\sum_{a \in A} s(a) = mB$.

ВОПРОС. Можно ли A так разбить на m непересекающихся подмножеств $S_1, S_2, \dots, S_m$, что для $1 \leqslant i \leqslant m \quad \sum_{a \in S_i} s(a) = B$.

(Отметим, что указанные в условии ограничения на размеры элементов приводят к тому, что каждое из множеств S_i должно содержать *ровно* три элемента из множества A.)

Мы будем доказывать сильную *NP*-полноту задачи 3-РАЗБИЕНИЕ в два этапа. Вначале докажем, что близкая к ней задача 4-РАЗБИЕНИЕ *NP*-полна в сильном смысле. Задача 4-РАЗБИЕНИЕ формулируется точно так же, как задача 3-РАЗБИЕНИЕ, с той лишь разницей, что A содержит $4m$ элементов, а каждый размер $s(a)$ должен подчиняться условию $B/5 > s(a) < B/3$. Отсюда следует, что каждое множество искомого разбиения будет содержать ровно *четыре* элемента.

`Теорема 4.3.` *Задача 4-РАЗБИЕНИЕ NP-полна в сильном смысле.*

Доказательство 1. Поскольку за полиномиальное время легко проверить, что заданное разбиение множества A удовлетворяет необходимым требованиям, то легко видеть, что задача 4-РАЗБИЕНИЕ лежит в классе *NP*. Сведём задачу ТРЁХМЕРНОЕ СОЧЕТАНИЕ (3-С) к ограниченной версии задачи 4-РАЗБИЕНИЕ, в которой размеры всех элементов ограничены полиномом от общего числа элементов и, следовательно, полиномом от $Length[I]$. В частности, положив $Max[I]$ равным $\max\{s(a) : a \in A\}$, мы увидим, что задача 4-РАЗБИЕНИЕ *NP*-полна даже в том случае, если ограничиться индивидуальными задачами, такими, что $Max[I] \leqslant 2^{16}|A|^4$.

Пусть $W = \{w_1, w_2, \dots, w_q\}$, $X = \{x_1, x_2, \dots, x_q\}$, $Y = \{y_1, y_2, \dots, y_q\}$ и $M \subseteq W \times X \times Y$ определяют произвольную индивидуальную задачу из 3-С. Не ограничивая общности, можно считать, что $|M| \geqslant q$. В соответствующей индивидуальной задаче из 4-РАЗБИЕНИЕ множество A будет иметь $4M$ элементов, по одному для каждого вхождения элемента множества $W \cup X \cup Y$ в тройку из M и по одному элементу для каждой тройки из M.

Элементы, соответствующие фиксированному $z \in W \cup X \cup Y$ обозначим через $z[1], z[2], \dots, z[N(z)]$, где $N(z)$ – число троек из M, в которые входит z. Назовём $z[1]$ «истинным элементом, соответствующим z», а $z[2], \dots, z[N(z)]$ – «фиктивными элементами, соответствующими z». Размеры этих элементов зависят от того, какому множеству – W, X или Y – принадлежит z, а также от номера элемента z в соответствующем множестве. Веса определяются следующим образом (r принято равным $32q$):

$$
\begin{array}{llll}
s(w_i[1]) & = 10r^4 + ir + 1, & 1 \leqslant i \leqslant q; & \\
s(w_i[l]) & = 11r^4 + ir + 1, & 1 \leqslant i \leqslant q, & 2 \leqslant l \leqslant N(w_i); \\
s(x_j[1]) & = 10r^4 + jr^2 + 2, & 1 \leqslant j \leqslant q; & \\
s(x_j[l]) & = 11r^4 + jr^2 + 2, & 1 \leqslant j \leqslant q, & 2 \leqslant l \leqslant N(x_j); \\
s(y_k[1]) & = 10r^4 + kr^3 + 4, & 1 \leqslant k \leqslant q; & \\
s(y_k[l]) & = 8r^4 + kr^3 + 4, & 1 \leqslant k \leqslant q; & 2 \leqslant k \leqslant N(y_k).
\end{array}
$$

Единственный элемент, соответствующий каждой тройке $m_i = \{w_i, x_j, y_k\} \in M$, обозначается через u_i, а его размер зависит от индексов элементов тройки следующим образом:

$$s(u_i) = 10r^4 - kr^3 - jr^2 - ir + 8.$$

Заметим, что если прибавить к $s(u_i)$ размеры трёх элементов, соответствующих w_i, x_j и y_k, то результатом будет $40r^4 + 15$ всякий раз, когда все три элемента «истинные» или когда все они «фиктивные». Возьмём это число в качестве границы B, т.е. положим

$$B = 40r^4 + 15.$$

Нетрудно проверить, что описанная выше сводимость полиномиальна, размеры всех элементов лежат строго между $B/3$ и $B/5$, а также что сумма размеров всех элементов, как и требуется, равна $|M|{\cdot}B$. Более того, заметим, что размер каждого элемента ограничен сверху величиной

$$12r^4 \leqslant 12 \cdot 8^4 |A|^4 < 2^{16} |A|^4.$$

Поэтому для доказательства сильной *NP*-полноты задачи 4-РАЗБИЕНИЕ достаточно показать, что искомое 4-разбиение существует тогда и только тогда, когда M содержит трёхмерное сочетание.

Предположим вначале, что имеется трёхмерное сочетание $M' \subseteq M$. Соответствующее 4-разбиение образуется из $|M|$ подмножеств, каждое из которых содержит 4 элемента (сокращённо 4-подмножеств) – один вида u_l, один вида $w_i[\cdot]$, один вида $x_j[\cdot]$ и один вида $y_k[\cdot]$, где $(w_i, x_j, y_k) = m_l \in M$. Если $m_l \in M'$, то группируем u_l с элементами $w_i[1], x_j[1]$ и $y_k[1]$. Если $m_l \in M \setminus M'$, то группируем u_l с «фиктивными» элементами, соответствующими w_i, x_j и y_k. Нетрудно видеть, что в нашем распоряжении имеется достаточно фиктивных элементов и такую группировку можно осуществить. В силу сделанных выше замечаний, сумма размеров элементов в каждом из образованных множеств в точности равна B. В результате получается искомое 4-разбиение.

Предположим теперь, что задано 4-разбиение нужного вида. Рассмотрим любое 4-множество из этого 4-разбиения. Рассматривая последовательно вычеты суммы размеров элементов этого множества по модулям r, r^2, r^3, r^4 и r^5 покажем, что это 4-множество должно содержать по одному элементу, соответствующему каждой из составляющих тройки, причем все три элемента одновременно являются или «истинными» или «фиктивными».

Поскольку $r \geqslant 4 \cdot 8 = 32$, то сумма вычетов по модулю r размеров элементов, входящих в 4-множество, будет равна вычету по модулю r суммы этих элементов и, следовательно, равна $B(mod\ r) = 15$. Для того чтобы сумма вычетов по модулю r размеров элементов, входящих в 4-множество, равнялась 15, имеется единственная возможность: 4-множество содержит по одному элементу, отвечающему некоторому элементу множеств W, X и Y соответственно, а также один элемент, отвечающий некоторой тройке из M. Пусть w_i, x_j и y_k обозначают соответствующие элементы множеств W, X и Y, а $m_l = (w_{i'}, x_{j'}, y_{k'})$ обозначает соответствующую тройку из M. Тогда вычет рассматриваемой суммы размеров элементов по модулю r^2 должен быть равен $((i-i')r+15)(mod\ r^2)$, а поскольку $(i-i')r+15 < r^2$, то должно иметь место равенство

$$B(mod\ r^2) = 15 = (i-i')r + 15.$$

Отсюда следует, что $i = i'$. Аналогично, поскольку $(j-j')r^2 + 15 < r^3$, то должно выполняться равенство

$$B(mod\ r^3) = 15 = (j-j')r^2 + 15.$$

Отсюда следует, что $j = j'$, а поскольку $(k-k')r^3 + 15 < r^4$, то должно иметь место равенство

$$B(mod\ r^4) = 15 = (k-k')r^3 + 15$$

и, значит, $k = k'$. Таким образом, на самом деле w_i, x_j и y_k – элементы, входящие в тройку m_i, и коэффициент при r^4 в сумме размеров элементов равен сумме коэффициентов при r^4 каждого из слагаемых. Эти коэффициенты были выбраны нами так, что их сумма равна 40 только в том случае, когда все три элемента одновременно «истинные» или «фиктивные».

Таким образом, в q из наших 4-множеств должно содержаться $3q$ «истинных» элементов, по одному для каждого элемента из множества $W \cup X \cup Y$, а каждое из 4-множеств состоит из одного элемента, соответствующего тройке из M, и трёх «истинных» элементов, соответствующих этой тройке. Следовательно, эти q троек из M образуют искомое трёхмерное сочетание. ■

Теорема 4.4. *Задача 3-РАЗБИЕНИЕ NP-полна в сильном смысле.*

Доказательство 2. Нетрудно видеть, что задача 3-РАЗБИЕНИЕ принадлежит классу *NP*. Сведем к задаче 3-РАЗБИЕНИЕ подзадачу задачи 4-РАЗБИЕНИЕ, составленную из индивидуальных задач, для которых

$$\max\{s(a) : a \in A\} \leqslant 2^{16}|A|^4;$$

при этом сохраняется ограниченность размеров элементов полиномом от общего числа элементов.

Пусть $A = \{a_1, a_2, \dots, a_{4n}\}$, граница B и размеры $s(a)$ элементов (для которых выполнены неравенства $B/5 < s(a) < B/3$ и $s(a) \leqslant 2^{16}|A|^4$) задают произвольную индивидуальную задачу 4-РАЗБИЕНИЕ. Соответствующая индивидуальная задача из 3-РАЗБИЕНИЕ будет содержать $24n^2 - 3n$ элементов, по одному для каждого элемента A, по два элемента для каждой пары элементов из A и $8n^2 - 3n$ «заполняющих» элементов.

Каждому отдельному элементу $a_i \in A$ сопоставим «регулярный» элемент w_i, размер определяется равенством

$$s'(w_i) = 4(5B + s(a_i)) + 1,$$

где $s'(\cdot)$ – функция размера в конструируемой индивидуальной задаче из 3-РАЗБИЕНИЕ. Паре элементов $a_i, a_j \in A$ соответствуют два «спаривающих» элемента $u[i,j]$ и $\bar{u}[i,j]$, размеры которых определяются так:

$$s'(u[i,j]) = 4(6B - s(a_i) - s(a_j)) + 2,$$

$$s'(\bar{u}[i,j]) = 4(5B + s(a_i) + s(a_j)) + 2.$$

Наконец, для $1 \leqslant k \leqslant 8n^2 - 3n$ имеется «заполняющий» элемент u_k^* размера $s'(u_k^*) = 20B$. В качестве границы B' для конструируемой индивидуальной задачи 3-РАЗБИЕНИЕ возьмем $64B + 4$.

Опять-таки читателю нетрудно будет убедиться в том, что эта сводимость является полиномиальной, а также, что размеры всех элементов заключены строго между $B'/4 = 16B+1$ и $B'/2 = 32B+2$ и что сумма размеров всех элементов равна $(2n^2 - n)B'$. Более того, поскольку размеры элементов множества A ограничены величиной $2^{16}|A|^4$, то размеры элементов получаемой индивидуальной задачи 3-РАЗБИЕНИЕ также будут ограничены полиномом от $|A|$, и, следовательно, полиномом от числа элементов задачи I', и, значит, полиномом от $Length[I]$. Таким образом, чтобы закончить доказательство *NP*-полноты в сильном смысле задачи 3-РАЗБИЕНИЕ, достаточно показать, что в построенной индивидуальной задаче I' 3-разбиение существует тогда и только тогда, когда в исходной индивидуальной задаче существует 4-разбиение.

Вначале предположим, что для исходной индивидуальной задачи имеется 4-разбиение. Соответствующее 3-разбиение строится следующим образом. Каждое из 4-множеств $\{a_i, a_j, a_k, a_l\}$ разбиваем произвольно на два 2-множества, скажем на $\{a_i, a_j\}$ и $\{a_k, a_i\}$. Тогда искомое 3-разбиение будет .содержать два 3-множества $\{w_i, w_j, u[i,j]\}$ и $\{w_k, w_l, \bar{u}[i,j]\}$. (Заметим, что можно было бы использовать $\bar{u}[k,l]$ вместо $u[i,j]$ и $u[k,l]$ вместо $\bar{u}[i,j]$.) Сумма размеров элементов каждого 3-множества будет равна B', поскольку $s(a_i) + s(a_j) + s(a_k) + s(a_l) = B$. Выполняя эту процедуру для каждого из n заданных 2-множеств, получим $2n$ 3-множеств, содержащих все «регулярные» элементы и n связанных пар, состоящих из «спаривающих» элементов. После этого остаются свободными $8n^2 - 3n$ связанных пар «спаривающих» элементов и $8n^2 - 3n$ «заполняющих» элементов. Поскольку сумма размеров двух связанных «спаривающих» элементов равна $44B + 4 = B - 20B$, то каждая такая связанная пара может быть сгруппирована с одним из оставшихся «заполняющих» элементов, в результате чего получается 3-разбиение.

Предположим теперь, что для построенной индивидуальной задачи имеется 3-разбиение. Из рассмотрения вычетов размеров элементов по модулю 4 получаем, что ни одно из 3-множеств не может содержать нечётное число «регулярных» элементов, ни одно из 3-множеств не может содержать три «спаривающих» элемента и ни одно из 3-множеств не может содержать два «регулярных» и один «заполняющий» элемент. Отсюда следует, что заданное 3-разбиение включает $2n$ 3-множеств, каждое из которых содержит два «регулярных» и один «спаривающий» элемент, а также $8n^2 - 3n$ 3-множества, каждое из которых содержит два «спаривающих» и один «заполняющий» элемент.

Рассмотрим любое из 3-множеств последнего типа. Пусть $u[i,j]$ (или $\bar{u}[k,l]$) является одним из двух «спаривающих» элементов этого множества. Если другой спаривающий элемент $u[s,t]$ этого множества отличен от $\bar{u}[i,j]$ (или $u[k,l]$), то $u[s,t]$ должен иметь тот же размер, что и $\bar{u}[i,j]$, и поэтому элементы $u[s,t]$ и $\bar{u}[i,j]$ можно поменять местами, чтобы получилось эквивалентное 3-разбиение. Эту операцию можно повторять до тех пор, пока не получится 3-разбиение, в котором *каждый* «заполняющий» элемент окажется в одном 3-множестве со связанной парой $u[i,j]$ и $\bar{u}[i,j]$. Таким образом, оставшиеся «спаривающие» элементы, входящие совместно с регулярными элементами в одно из множеств 3-разбиения, также разбиваются на пары. В результате этого 3-множества, содержащие «регулярные» элементы, разбиваются на n пар. Поскольку два «спаривающих» элемента в каждой такой паре 3-множеств оказываются связанными, сумма их размеров равна $44B + 4$, следовательно, сумма четырёх «регулярных» элементов, входящих в эти два 3-множества, должна быть равна $84B + 4$. Отсюда следует, что соответствующие четыре элемента множества A образуют 4-множество, сумма размеров элементов которого равна B. Следовательно, n пар рассматриваемых 3-множеств порождают искомое 4-разбиение. ■

Заметим, что если эту сводимость рассматривать как сводимость общей задачи 4-РАЗБИЕНИЕ к задаче 3-РАЗБИЕНИЕ, то она не доказывала бы сильную *NP*-полноту задачи 3-РАЗБИЕНИЕ. Чтобы получить результат о сильной *NP*-полноте, необходимо рассмотреть *NP*-полную подзадачу задачи 4-РАЗБИЕНИЕ, в которой величина $\max\{s(a)\}$ ограничена полиномом. Однако легко понять, что конкретный вид полинома в оценке несущественен. На самом деле было бы удобнее, если бы у нас была возможность оперировать со сводимостями подобного типа без необходимости вдаваться в детали подзадач и конкретных полиномов. Этого можно достичь с помощью определения и леммы, приводимых ниже.

Пусть Π и Π' – произвольные задачи распознавания; D_Π и $D_{\Pi'}$ – множества их индивидуальных задач; Y_Π и $Y_{\Pi'}$ – множества индивидуальных задач с ответом «да»; $Max, Length, Max'$ и $Length'$ – соответствующие функции максимума и длины записи. Говорят, что задача Π *псевдополиномиально сводится* к задаче Π', если существует функция $f : D_\Pi \to D_{\Pi'}$ обладающая следующими свойствами:

(a) для любого $I \in D, I \in Y$ тогда и только тогда, когда $f(I) \in Y_{\Pi'}$;

(б) функция f может быть вычислена за время, ограниченное полиномом от двух переменных $Max[I]$ и $Length[I]$;

(в) существует такой полином q_1, что для всех $I \in D_\Pi$ имеет место неравенство

$$q_1(Length'[f(I)]) \geqslant Length(I);$$

(г) существует такой полином q_2 от двух переменных, что для всех $I \in D_\Pi$ выполнено неравенство

$$Max'[f(I)] \leqslant q_2(Max[I], Length[I]).$$

Говорят также, что функция, обладающая перечисленными свойствами, реализует псевдополиномиальную сводимость Π к Π'.

Лемма 4.1. *Если* Π *–* *NP-полная в сильном смысле задача,* $\Pi' \in NP$ *и* Π *псевдополиномиально сводится к* Π'*, то* Π' *–* *NP-полная в сильном смысле задача.*

Доказательство 3. Пусть f – функция, реализующая псевдополиномиальную сводимость, (q_1 и q_2 – соответствующие полиномы, описанные выше. Без потери общности можно предполагать, что q_1 и q_2 – полиномы с целыми положительными коэффициентами, ибо эти полиномы можно привести к такому виду, не уменьшая принимаемых ими значений. Поскольку задача II *NP*-полна в сильном смысле, то существует такой полином p, что задача Π_p является *NP*-полной. Более того, p можно выбрать так, что у него будут только целые неотрицательные коэффициенты, поскольку если p_0 – любой полином с целыми коэффициентами, для которого при всех x выполняется неравенство $p_0(x) \geqslant p(x)$, то задача Π_{p0} будет содержать все индивидуальные задачи из Π_p, и, следовательно, задача Π_{p0} будет *NP*-полной, если Π_p была *NP*-полной. Определим полином $\hat{p}$ следующим образом:

$$\hat{p}(x) = q_2(p(q_1(x)), q_1(x)).$$

Мы утверждаем, что функция f, будучи ограниченной на индивидуальные задачи из Π_p, осуществляет полиномиальное сведение задачи Π_p к задаче Π'_p, что доказывает *NP*-иолноту задачи Π'_p. Проверим вначале, что любая индивидуальная задача I из Π_p при отображении f переходит в индивидуальную задачу из Π'_p. По определению задачи Π_p и неравенств, которым удовлетворяют многочлены q_1 и q_2, получаем, что для каждой индивидуальной задачи I из Π_p имеют место неравенства

$$\begin{gathered} Max'[f(I)] \leqslant q_2(Max[I], Length[I]) \leqslant \\ \leqslant q_2(p(Length[I]), Length[I]) \leqslant \\ \leqslant q_2(p(q_1(Length'[f(I)])), q_1(Length'[f(I)])) = \\ = \hat{p}(Length'[f(I)]). \end{gathered}$$

Таким образом, индивидуальная задача $f(I)$ принадлежит задаче Π'_p. Из условий (а) и (б) определения псевдополиномиальной сводимости и того факта, что для любой индивидуальной задачи I из Π_p выполнено неравенство $Max[I] \leqslant p(Length[I])$, немедленно следует, что f удовлетворяет всем необходимым требованиям и реализует полиномиальную сводимость. Следовательно, задача Π'_p *NP*-полна, откуда следует сильная *NP*-полнота задачи Π'. ■

Эта лемма освобождает нас от необходимости иметь дело с конкретными подзадачами Π_p при доказательстве результатов о сильной *NP*-полноте, что представляет собой большое удобство, поскольку мы редко интересуемся конкретным видом соответствующего полинома. Однако сложное определение псевдополиномиальной сводимости может показаться труднопреодолимым препятствием при использовании этого подхода. В действительности оно не столь сложно, как кажется. Условие (а) совпадает с одним из условий определения обычной полиномиальной сводимости. Условие (б) почти совпадает со вторым условием, однако оставляет нам несколько больше свободы при выборе сложности сводящей функции. Условие (в) будет выполнено почти для всех сводимостей, исключая самые экзотические, поскольку требуется лишь, чтобы сводимость не приводила к значительному *уменьшению* длины входа. Наиболее важным условием определения является условие (г), назначение которого заключается в том, чтобы величина наибольшего числа в конструируемой индивидуальной задаче не росла экспоненциально в зависимости от функций Max и $Length$ исходной индивидуальной задачи.

Например, конструкцию, которой мы пользовались для доказательства теоремы 4.4, можно рассматривать как псевдополиномиальное сведение задачи 4-РАЗБИВАНИЕ к задаче 3-РАЗБИЕНИЕ. Последняя задача заслужила название седьмой «основной *NP*-полной задачи» в связи с тем, что она легко может быть псевдополиномиально сведена к другим задачам. Например, можно воспользоваться такой сводимостью для доказательства сильной *NP*-полноты задачи УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ, *NP*-полнота которой была доказана в разд. 3.2.2.

Теорема 4.5. *Задача УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ NP-полна в сильном смысле.*

Доказательство 4. Напомним, что в условии этой задачи дано множество заданий T, длительность $l(t) \in Z^+$ каждого задания $I \in T$ и временно́й интервал $[r(t), d(t)]$, в пределах которого оно должно быть выполнено. Спрашивается, можно ли так упорядочить задания, чтобы были удовлетворены все условия и в каждый момент времени в процессе выполнения находилось только одно задание. В разд. 3.2.2 было доказано, что эта задача *NP*-полна и, следовательно, принадлежит классу *NP*. Построим псевдополиномиальное сведение задачи 3-РАЗБИЕНИЕ к задаче УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ.

Пусть $A = \{a_1, a_2, \dots, a_{3m}\}$, $B \in Z^+$ и $s(a_1), s(a_2), \dots, s(a_{3m})$ определяет произвольную задачу из 3-РАЗБИЕНИЕ. Соответствующая индивидуальная задача из УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ определяется следующим образом:

$$T = A \cup \{t_i : 1 \leqslant i < m\};$$

$$l(t) = \begin{cases} 1, \text{ если } t = t_i, 1 \leqslant i < m; \\ s(a_j), \text{ если } t = a_j \in A; \end{cases}$$

$$r(t) = \begin{cases} iB + i - 1, \text{ если } t = t_i, 1 \leqslant i < m; \\ 0, \text{ если } t = a_j \in A; \end{cases}$$

$$r(t) = \begin{cases} iB + i, \text{ если } t = t_i, 1 \leqslant i < m; \\ mB + m - 1, \text{ если } t = a_j \in A; \end{cases}$$

Эта сводимость, очевидно, может быть выполнена за полиномиальное время только по отношению к длине записи исходной информации, а длина индивидуальных задач, получаемых при таком сведении, полиномиально эквивалентна длине исходных задач. Таким образом, условия (б) и (в) из определения полиномиальной сводимости выполняются. Более того, наибольшее число в построенной индивидуальной задаче равно $mB + m - 1$, поэтому выполняется также условие (г). Остается только показать, как в обычных доказательствах NP-полноты, что выполнено условие (а).

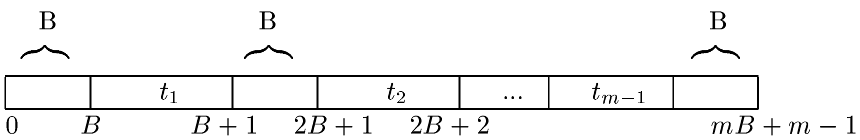


Рис. 4.9. Вид последовательности, необходимый для того, чтобы были выполнены все ограничения индивидуальной задачи из УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ, получаемой из индивидуальной задачи 3-РАЗБИЕНИЕ в доказательстве теоремы 4.5.

В любой последовательности заданий, удовлетворяющей перечисленным выше условиям, задание $t_i, 1 \leqslant i < m$, должно выполняться в промежутке времени от $iB+i-1$ до $iB+1$, как указано на рис. 4.9. При этом остается m отдельных блоков времени продолжительностью B каждый, но поскольку времени в этих блоках ровно столько, чтобы можно было успеть выполнить все задания, то все эти блоки должны быть заполнены полностью. Следовательно, эти блоки играют ту же роль, что и множества $S_1, S_2, \dots, S_m$ в искомом разбиении множества A. Отсюда вытекает, что искомая последовательность существует тогда и только тогда, когда существует искомое разбиение в исходной индивидуальной задаче из 3-РАЗБИЕНИЕ.

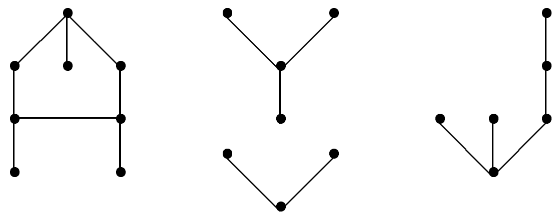
Условие (а), следовательно, также выполняется, и мы действительно получаем псевдополиномиальную сводимость задачи 3-РАЗБИЕНИЕ к задаче УПОРЯДОЧЕНИЕ ВНУТРИ ИНТЕРВАЛОВ. По лемме 4.1 отсюда следует, что последняя задача NP-полна в сильном смысле. ■

В качестве упражнения читателю можно предложить построить аналогичное сведение задачи 3-РАЗБИЕНИЕ к задаче РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ, определенной в разд. 3.2.1. В наших списках NP-полных задач имеется большое число других задач, сильная NP-полнота которых доказывается сравнительно несложно, небольшой модификацией приведенных ранее доказательств посредством замены в них задачи РАЗБИЕНИЕ задачей 3-РАЗБИЕНИЕ. Простота выполнения соответствующих модификаций указывает на полезность задачи 3-РАЗБИЕНИЕ.

Настоящий раздел мы закончим примером того, как псевдополиномиальная сводимость задачи 3-РАЗБИЕНИЕ может быть использована для доказательства NP-полноты задачи, не являющейся задачей с числовыми параметрами. В действительности эта задача вообще не будет содержать числовых параметров!

Напомним формулировку задачи ИЗОМОРФИЗМ ПОДГРАФУ из разд. 2.1: «Заданы два графа G и H, верно ли, что H изоморфен некоторому подграфу графа G?» В разд. 3.2.1 мы доказали NP-полноту этой задачи, указав, что она содержит в качестве частного случая задачу КЛИКА. Однако имеется одна важная подзадача задачи ИЗОМОРФИЗМ ПОДГРАФУ, принадлежность которой классу P уже установлена. Это задача ИЗОМОРФИЗМ ПОДДЕРЕВУ. В ней требуется, чтобы G и H были деревьями (дерево – это связный граф, не содержащий циклов). Полиномиальный алгоритм решения этой задачи был получен Эдмондсом и Матулой [111] (см. также [449]).

Исходя из нашей методологии, состоящей в возможно большем сужении пограничной области между трудными и легкими подзадачами *NP*-полной задачи, можно сформулировать следующий вопрос: «Что будет, если потребовать, чтобы только *один* из графов G или H был деревом?» В одном случае ответ очевиден. Вариант задачи ИЗОМОРФИЗМ ПОДГРАФУ, в котором требуется, чтобы G был деревом, содержит в качестве подзадачи задачу ГАМИЛЬТОНОВ ПУТЬ и тем самым является *NP*-полным, Вариант, в котором требуется, чтобы H было деревом, более интересен. В этом случае нам известно, что если граф H не ацикличен (ациклчный – означает, не содержит циклов), то H не может быть подграфом графа G, но отсюда отнюдь не следует, что граф H должен быть деревом (поскольку H может оказаться несвязным). В общем случае ацикличный граф называется *лесом*, при этом связные леса являются деревьями (рис. 4.10).



G F T

Рис. 4.10. Примеры произвольного графа G, леса F и дерева T. Граф G не есть лес, а лес F не есть дерево. Лес F не есть подлес дерева T, но каждое дерево из F есть поддерево дерева T (граф – Graph, лес – Forest, дерево – Tree).

Назовем подзадачу задачи ИЗОМОРФИЗМ ПОДГРАФУ, в которой G – дерево, а H – лес, задачей ИЗОМОРФИЗМ ПОДЛЕСУ. Несмотря на то, что последняя задача аналогична задаче ИЗОМОРФИЗМ ПОДДЕРЕВУ, справедлива следующая теорема.

Теорема 4.6. *Задача ИЗОМОРФИЗМ ПОДЛЕСУ является NP -полной.*

Доказательство 5. Принадлежность рассматриваемой задачи классу *NP* следует из аналогичного результата для задачи ИЗОМОРФИЗМ ПОДГРАФУ. Укажем псевдополиномиальное сведение задачи 3-РАЗБИЕНИЕ к задаче ИЗОМОРФИЗМ ПОДЛЕСУ. В силу леммы 4.1 отсюда будет следовать утверждение теоремы.

Пусть $A = \{a_1, a_2, \dots, a_m\}$, $B \in Z^+$ и $s(a_1), s(a_2), \dots, S(a_m) \in Z^+$ определяет произвольную индивидуальную задачу из 3-РАЗБИЕНИЕ. Соответствующая индивидуальная задача из ИЗОМОРФИЗМ ПОДЛЕСУ представлена на рис. 4.11.

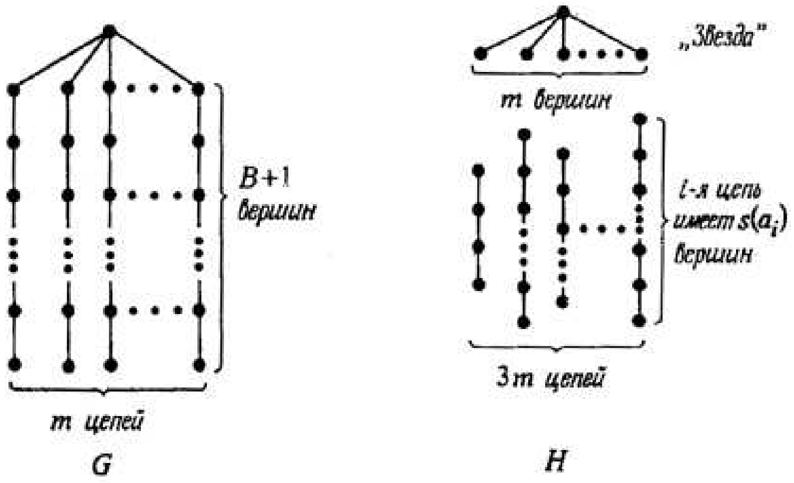


Рис. 4..6: *Дерево G и лес H, соответствующие индивидуальной задаче 3-РАЗБИЕНИЕ в доказательстве теоремы 4.6.*

Дерево G состоит из m цепей, по $B+1$ вершин в каждой, концы которых соединены с дополнительной общей вершиной. Лес H состоит из $3m+1$ дерева, в их число входит одна «звезда», содержащая $m+1$ вершину и $3m$ цепей, каждая из которых соответствует конкретному элементу $a \in A$ и имеет $s(a)$ вершин.

Любой изоморфизм графа H с подграфом графа G должен отображать центр звезды в единственную вершину высокой степени графа G [1]). Все m вершин, соседних с центром звезды из H должны, следовательно, отображаться в вершины графа G, соседние этой вершине. При этом в графе G остаются свободными m цепей (причем каждая состоит из B вершин), в которые при изоморфизме должны отобразиться оставшиеся $3m$ цепей графа H. Отображение этих цепей графа H в оставшиеся свободными цепи графа G соответствует разбиению множества A на m подмножеств и, согласно нашей конструкции, может быть осуществлено тогда и только тогда, когда сумма размеров элементов каждого множества равна B. Таким образом, искомый изоморфизм графа H с подграфом графа G существует тогда и только тогда, когда существует искомое 3-разбиение множества A.

Рис. 4.11. Дерево G и лес H, соответствующие индивидуальной задаче 3-РАЗБИЕНИЕ в доказательстве теоремы 4.6.

Отсюда следует, что выполнено условие (а) определения псевдополиномиальной сводимости. Легко видеть, что эта сводимость может быть осуществлена за полиномиальное время по отношению к m и B, так что условие

[1]Это утверждение верно только при $m > 2$. При $m \leqslant 2$ сводимость модифицируется очевидным образом. – Прим. ред.

(б) также выполнено. Общее число вершин в графах G и H одинаково и равно $2(mB+1)$, так что выполнено и условие (в). Наконец, в конструируемой индивидуальной задаче числовые параметры отсутствуют, поэтому и условие (г) выполняется. Следовательно, по лемме 4.1 задача ИЗОМОРФИЗМ ПОДЛЕСУ *NP*-полна в сильном смысле, откуда следует, что она *NP*-полна в обычном смысле, что и требовалось доказать. ■

4.3. Временн*á*я сложность как функция натуральных параметров

До сих пор в настоящей главе изучение подзадач мотивировалось в основном тем, что на практике истинной нашей целью является решение подзадачи, а не самой общей задачи. Зная очертания пограничной области между *NP*-полными и полиномиально разрешимыми подзадачами, при возникновении конкретной подзадачи мы оказываемся лучше подготовленными к концентрации исследований по поиску алгоритмов в наиболее перспективном направлении.

Результаты, полученные относительно подзадач, могут быть также использованы в качестве вспомогательного ориентира при поиске алгоритма решения общей задачи. Если общая задача оказывается *NP*-полной, то для её решения (если $P \neq NP$), как нам известно, могут существовать только экспоненциальные алгоритмы, но разные алгоритмы могут быть в различной степени «экспоненциальными», и одни могут оказаться для нас предпочтительнее других. Это особенно очевидно для практических задач, когда временн*á*я сложность выражается в терминах естественных параметров задачи, а не в терминах искусственно созданной «длины входа».

Рассмотрим, например, задачу РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ из разд. 3.2.1. В этой задаче набор естественных параметров включает следующие величины: число заданий n, число процессоров m и длительность L самого продолжительного задания. Обычный результат об *NP*-полноте этой задачи, полученный в разд. 3.2.1, утверждает, что если $P \neq NP$, то задача РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ не может быть решена за время, ограниченное полиномом от трёх параметров: n, m и $\log L$. Однако можно задаться вопросом: «Существует ли алгоритм, временн*á*я сложность которого ограничена полиномом от m^n и $\log L$, или полиномом от n^m и $\log L$, или полиномом от n, m и L, или даже полиномом от $(nL)^m$?

Полученные нами результаты о сложности подзадач проливают некоторый свет на эти вопросы. Исходный результат об *NP*-полноте задачи РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ показывает в действительности, что подзадача, в которой m ограничено числом 2, является *NP*-полной. По этой причине (если $P \neq NP$) для решения задачи не может существовать алгоритма, временн*á*я сложность которого ограничена полиномом от n^m и $\log L$, поскольку для указанной подзадачи этот алгоритм был бы полиномиальным. Полученные нами результаты не исключают существование алгоритма, временна́я сложность которого была бы ограничена полиномом от m^n и $\log L$, и в действительности алгоритмы полного перебора, имеющие такую временну́ю сложность, могут быть построены. Аналогичным образом сильная *NP*-полнота задачи РАСПИСАНИЕ ДЛЯ МУЛЬТИПРОЦЕССОРНОЙ СИСТЕМЫ, о которой говорилось в разд. 4.2.2, исключает существование алгоритмов, полиномиальных по n, m и L (если $P \neq NP$).

Однако этот результат оставляет открытым вопрос о существовании алгоритма, полиномиального по $(nL)^m$ (который давал бы псевдополиномиальный алгоритм при каждом *фиксированном* значении m), и опять-таки можно показать, что такой алгоритм существует.

Таким образом, рассматривая подзадачи, возникающие из исходной задачи при ограничении одного или нескольких из её естественных параметров, мы получаем полезную информацию о возможных типах алгоритмов решения общей задачи. Однако, чтобы обеспечить выразимость $Length(I)$ в виде полинома от параметров, выбранных нами в качестве достаточных представителей размера индивидуальной задачи, должны быть приняты особые меры предосторожности. (Чтобы для исходной задачи класс ранее рассмотренных полиномиальных алгоритмов совпадал с классом алгоритмов, полиномиальных относительно выбранных параметров.) А в остальном можно выбирать любые параметры, которые представляются естественными и адекватно отражающими смысл задачи. Из общего результата об *NP*-полноте, следовательно, будет вытекать, что задачу нельзя решить за время, ограниченное полиномом от всех выбранных параметров, а информация, получаемая при ограничении этих параметров, может иметь большое значение в связи с существованием общих алгоритмов других типов.

Хотя вопросы, касающиеся сильной *NP*-полноты и существования псевдополиномиальных алгоритмов, относятся в основном к задачам с числовыми параметрами, однако аналогичные методы можно плодотворно применять и к другим задачам. Это объясняется тем, что в любой задаче имеются числовые параметры, такие, как мощности множеств, значения границ и т.д. Например, *NP*-полнота задачи 3-ВЫПОЛНИМОСТЬ не допускает (при условии $P \neq NP$) существования для решения задачи ВЫПОЛНИМОСТЬ алгоритма, временна́я сложность которого была бы ограничена полиномом от $(nm)^M$ (где m – число дизъюнкций, n – число литералов, M – максимум числа литералов в дизъюнкции). Напротив, для задачи КЛИКА существует алгоритм с оценкой n^D (где n – число вершин графа, D – максимальная степень вершины). Таким образом, теория *NP*-полных задач может служить ориентиром не только при поиске полиномиальных, но и экспоненциальных алгоритмов.

Глава 5.

NP-трудные задачи

В этой главе мы завершим рассмотрение основных понятий и применений теории NP-полных задач. Глава состоит из двух частей. В первой, основной, части показано, как сфера приложений этой теории может быть расширена за пределы класса NP при использовании сводимости задач более общего вида. Во второй части даётся краткая историческая справка о развитии теории NP-полноты и приводятся некоторые альтернативные определения и термины, получившие распространение в литературе.

5.1. Сводимость по Тьюрингу и NP-трудные задачи

Хотя до сих пор мы ограничивались в основном задачами из класса NP, очевидно, что методы, используемые для доказательства NP-полноты задач, можно также применять для установления трудности задач, лежащих вне класса NP. Пусть Π – произвольная задача распознавания свойств, к которой сводится некоторая NP-полная задача. Тогда независимо от того, принадлежит ли задача Π классу NP или нет, она обладает тем свойством, что её невозможно решить за полиномиальное время, если $P \neq NP$. Будем говорить, что такая задача Π является «NP-трудной», поскольку она в некотором смысле не менее трудна, чем любая NP-полная задача.

Ниже будет дано более общее определение NP-трудной задачи. Это объясняется тем, что определение полиномиальной сводимости задач можно обобщить на более широкий класс задач, чем задачи распознавания, причем все они окажутся не менее трудными, чем NP-полные задачи. Как и в гл. 2, все определения будут даны и формально – в терминах языков и машин Тьюринга, и неформально – терминах задач и алгоритмов.

Этот более общий класс задач, к которым будут применимы наши определения, – класс так называемых «переборных задач», или «задач поиска». *Переборная задача* Π определяется как множество D_Π конечных объектов, называемых *индивидуальными задачами*, причём для каждой индивидуальной задачи $I \in D_\Pi$ известно множество $S_\Pi[I]$ конечных объектов, называемых ***решениями*** задачи I. Будем говорить, что алгоритм ***решает*** переборную задачу II, если, получив в качестве входа произвольную индивидуальную задачу $I \in D_\Pi$, этот алгоритм выдаёт ответ «нет», если $S_\Pi[I]$ пусто, и вырабатывает некоторое решение $s \in S_\Pi[I]$ в противном случае.

Например, множество решений в индивидуальной задаче о коммивояжере (в оптимизационной постановке) состоит из *всех* маршрутов, имеющих минимально возможную длину. От алгоритма, который решает эту переборную задачу, требуется только умение находить для любой индивидуальной задачи один из таких маршрутов. Другой пример – когда множество $S_\Pi[I]$ может оказаться пустым – даёт нам так называемая «задача конструирования гамильтонова цикла», в которой множество решений для заданного графа G состоит из всех гамильтоновых циклов в G. Алгоритм решения этой задачи должен выдавать «нет», когда G не содержит гамильтонова цикла; в противном случае он должен указать один такой цикл для G. Заметим, что любая задача распознавания может быть сформулирована как переборная задача следующим образом:

$$S_\Pi[I] = \begin{cases} \{\text{"да"}\}, & \text{если } I \in Y_\Pi; \\ \varnothing, & \text{если } I \notin Y_\Pi; \end{cases}$$

Полезно отметить, что все задачи распознавания ранее были сформулированы именно таким образом, поэтому любая задача распознавания может рассматриваться просто как частный случай переборной задачи.

В формальных терминах переборной задаче соответствует понятие «*словарного отношения*». Пусть имеется конечный алфавит Σ. Словарным отношением над алфавитом Σ называется бинарное отношение $R \subseteq \Sigma^+ \times \Sigma^+$, где $\Sigma^+ = \Sigma^* \backslash \{\varepsilon\}$ – множество всех непустых слов над Σ. Язык L над Σ можно отождествлять со словарным отношением

$$R = \{(x, s) : x \in \Sigma^+ \text{и } x \in L\},$$

где s – любой фиксированный символ из Σ. (Заметим, что при этом игнорируется вопрос, принадлежит ли языку пустое слово, однако это не влияет на вычислительные вопросы, которыми мы занимаемся.) Мы говорим, что функция $f : \Sigma^* \to \Sigma^*$ *реализует* словарное отношение R тогда и только тогда, когда для каждого $x \in \Sigma^+$ имеет место $f(x) = \varepsilon$, если не существует $y \in \Sigma^+$, такого, что $(x, y) \in R$; в противном случае $f(x)$ равняется некоторому $y \in \Sigma^+$ для которого $(x, y) \in R$. ДМТ-программа M *разрешает* словарное отношение R, если функция f_M, вычислимая программой M, реализует R.

Соответствие между переборными задачами и словарными отношениями осуществляется снова с помощью схем кодирования, только теперь схема кодирования для задачи Π должна выдавать как слово, являющееся кодом каждой индивидуальной задачи $I \in D_\Pi$, так и слово, являющееся кодом каждого решения $s \in S_\Pi[I]$. При такой схеме кодирования e переборная задача Π соответствует словарному отношению $R[\Pi, e]$, определённому следующим образом:

$$R[\Pi, e] = \left\{ (x, y) : \begin{array}{l} x \in \Sigma^{+}\text{является кодом индивидуальной} \\ \text{задачи } I \in D_{\Pi}, \text{ а } y \in \Sigma^{+} - \text{кодом} \\ \text{некоторого решения } s \in S_{\Pi}[I] \\ \text{при схеме кодирования } e \end{array} \right\}.$$

Будем говорить, что задача Π (при схеме кодирования e) разрешима полиномиальным алгоритмом, если существует полиномиальная ДМТ-программа, которая «разрешает» $R[\Pi, e]$.

Необходимость обобщения понятия полиномиальной сводимости вызвана тем, что любое полиномиальное сведение задачи распознавания II к задаче распознавания Π' порождает алгоритм A для решения задачи Π при помощи гипотетической «подпрограммы» решения задачи Π'. Если дана переборная задача Π и её индивидуальная задача I, то этот алгоритм сначала конструирует эквивалентную индивидуальную задачу I' из Π', затем применяет упомянутую подпрограмму к I' и наконец выдаёт ответ, выработанный этой подпрограммой, поскольку он будет также правильным ответом для I. За исключением времени работы подпрограммы, алгоритм A работает в течение полиномиального времени. Таким образом, если бы подпрограмма представляла собой полиномиальный алгоритм решения задачи Π', вся описанная процедура целиком была бы полиномиальным алгоритмом для решения задачи Π.

Заметим, что последнее утверждение останется справедливым, если алгоритм A будет использовать подпрограмму для Π' несколько (но не более чем полиномиальное число) раз, и даже если предполагаемая подпрограмма будет предназначена для решения переборной задачи, а не задачи распознавания. Это является основанием для предлагаемого ниже обобщения. Будем говорить, что переборная задача Π сводится за полиномиальное время к переборной задаче Π' в смысле Тьюринга (или, для краткости, *сводится по Тьюрингу*), если существует алгоритм A, который решает задачу Π с помощью гипотетической подпрограммы S решения задачи Π', такой, что если S была бы полиномиальным алгоритмом для Π', то A был бы полиномиальным алгоритмом для Π.

Это определение можно формально выразить в терминах так называемых *оракулъных* машин. Для определенности мы пользуемся моделью машины Тьюринга, хотя аналогичные определения можно было бы сделать с помощью любой стандартной модели вычислений. Оракульная машина Тьюринга (ОМТ) состоит из стандартной детерминированной машины Тьюринга, снабженной дополнительной *оракульной лентой* с ячейками, которые занумерованы числами $\ldots - 2, -1, 0, 1, 2 \ldots$, и читающей/ пишущей *оракульной головкой* для работы с этой лентой. Схематически эта машина изображена на рис. 5.1.

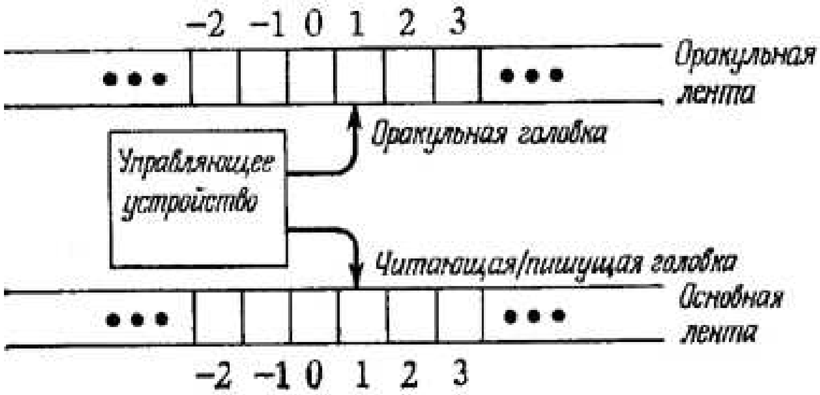


Рис. 5..1: *Схематическое представление оракульной машины Тьюринга (ОМТ).*

Программа для ОМТ аналогична программам для детерминированной машины Тьюринга и характеризуется следующими данными:

(1) конечным множеством Γ *ленточных символов*, содержащим некоторое подмножество $\Sigma \in \Gamma$ *входных символов* и выделенный *пустой символ* $b \in \Gamma - \Sigma$;

(2) конечным множеством *состояний* Q, среди которых выделены некоторое *начальное состояние* q_0, *конечное состояние* q_h, выделенное состояние *вопроса к оракулу* q_c и *резюмирующее состояние* q_r;

(3) *переходной функцией*

$$\delta : (Q - \{q_h, q_c\}) \times \Gamma \times \Sigma \to Q \times \Gamma \times \Sigma\{-1, +1\} \times \{-1, +1\}.$$

Вычисление программы на ОМТ, имеющей вход $x \in \Sigma^*$, очень сходно с вычислением на ДМТ, за исключением того, что если управляющее устройство находится в состоянии q_c, то поведение на следующем шаге зависит от конкретной оракульной функции $g : \Sigma^* \to \Sigma^*$. В начальный момент вычисления символы слова x записаны в ячейках с номерами $1, \dots, |x|$ основной ленты, при этом остальные ячейки этой ленты и все ячейки оракульной ленты пусты, каждая головка находится над ячейкой 1 на своей ленте и управляющее устройство находится в состоянии q_0. Вычисления осуществляются шаг за шагом, причем на каждом шаге может возникнуть одна из трёх ситуаций:

а. Если текущее состояние есть q_h, то вычисление заканчивается, и на этом работа машины прекращается.

б. Если текущее состояние $q \in Q \setminus \{q_h, q_c\}$, то поведение зависит от символов, прочитанных на лентах, и от переходной функции δ. Пусть s_1 – это символ в ячейке, просматриваемой в текущий момент основной головкой, s_2 – символ в ячейке, просматриваемой в этот момент оракульной головкой, и $(q', s'_1, s'_2, \Delta_1, \Delta_2)$ – значение $\delta(q_1, s_1, s_2)$. Тогда машина переходит из состояния q в состояние q'; основная головка записывает s_1' вместо s_1 и сдвигается на 1 ячейку вправо, если $\Delta_1 = +1$, или на 1 ячейку влево, если $\Delta_1 = -1$; оракульная головка. записывает s_2' вместо s_2 и сдвигается аналогично в соответствии со значением Δ_2. Таким образом, всё происходит как на одном шаге в обычной ДМТ, за исключением того, что используются две ленты.

в. Если текущее состояние есть q_c, то поведение зависит от содержания оракульной ленты и от оракульной функции q. Пусть $y \in \Sigma^*$ – слово, записанное на ленте, начиная с 1-ой ячейки и кончая той, в которой расположена оракульная головка, причем $y = \varepsilon$, если эта головка находится левее ячейки [1]. Тогда за один шаг оракульная лента изменяется так, что в ячейках $1, \dots, |z|$ оказывается записанным слово r, а остальные ячейки пусты; оракульная головка перемещается в позицию для просмотра ячейки 1, и машина переходит из состояния q_c в состояние q_r. Этот шаг не изменяет содержимого основной ленты и положения основной головки.

Главное различие между ОМТ и ДМТ проявляется в последней ситуации, которая обеспечивает возможность ОМТ «консультироваться с оракулом». Если ОМТ записывает вопрос y на оракульной ленте и затем принимает состояние вопроса к оракулу, ответное слово $z = g(y)$ выдаётся за один шаг вычисления. Это соответствует обращению к гипотетической подпрограмме вычисления функции g. Вычисление ОМТ-программы M на входе x зависит как от x, так и от используемой оракульной функции g.

Мы будем использовать символ M_g для обозначения «релятивизированной» ОМТ-программы, получаемой путем комбинирования программы m с оракулом g. Если M_g останавливается на всех входах $x \in \Sigma^*$, её можно рассматривать как программу, вычисляющую функцию $f_M^g : \Sigma^* \to \Gamma^*$, определяемую точно так же, как для случая ДМТ. Будем называть M_g *полиномиальной ОМТ-программой*, если существует полином p, такой, что M_g останавливается на $p(|x|)$ шагов для любого $x \in \Sigma^*$.

Пусть R и R' – два произвольных словарных отношения над алфавитом Σ. Определим полиномиальную сводимость R к R' в смысле Тьюринга как ОМТ-программу M с входным алфавитом Σ, такую, что для любой функции $g : \Sigma^* \to \Sigma^*$, реализующей отношение R', релятивизированная программа M_g будет полиномиальной ОМТ-программой, а функция f_M^g, вычисленная программой M_g, будет реализовывать отношение R. Если R таким образом сводится к R', то это обозначается следующим образом: $R \propto_T R'$ (читается «R сводится к R' по Тьюрингу»). Заметим, что $\propto_T$, подобно $\propto$, транзитивно.

Теперь можно определить понятие «*NP*-трудной» задачи. Словарное отношение R назовем *NP*-трудным, если существует *NP*-полный язык L (в

[1] В первом издании книги y определялось так: пусть $y \in \Sigma^*$ – это слово, записанное в ячейках $1, \dots, |y|$ оракульной ленты, где ячейка $|y| + 1$ – первая пустая ячейка вправо от ячейки 0 и пусть $z \in \Sigma^*$ – значение $g(y)$. Авторы благодарят Дж. Лукера за его замечание, позволившее уточнить определение слова y. (Можно показать в качестве упражнения, что некоторые *NP*-трудные задачи решались бы за полиномиальное время, если использовать определение, представленое в 1-м издании.) – *Дополнение авторов.*

свою очередь представленный как словарное отношение описанным выше способом), такой, что $L \propto_T R$. Переборная задача Π (при схеме кодирования e) называется *NP*-трудной, если словарное отношение $R[\Pi, e]$ является *NP*-трудным. Неформально можно сказать, что переборная задача Π является *NP*-трудной, если существует некоторая *NP*-полная задача Π', которая сводится по Тьюрингу к задаче Π. Нетрудно видеть, что если словарное отношение R (или переборная задача Π) является *NP*-трудной, то задача Π не может быть решена за полиномиальное время, если $P \neq NP$.

Заметим, что если R – произвольное *NP*-трудное словарное отношение и при этом R сводится по Тьюрингу к словарному отношению R', то, в силу транзитивности отношения $\propto_T$, словарное отношение R' также должно быть *NP*-трудным. Более того, используя соответствие между языками и словарными отношениями, а также соответствие между задачами распознавания и переборными задачами, немедленно получаем, что все *NP*-полные языки и все *NP*-полные задачи являются *NP*-трудными.

Мы вернемся к ОМТ в гл. 7, а сейчас продолжим обсуждение на уровне задач, применяя введённые выше понятия сводимости по Тьюрингу, и *NP*-трудных задач обычным неформальным образом. Первое и наиболее тривиальное применение касается задач, дополнительных к *NP*-полным. Задача Π^c называется дополнительной к задаче распознавания Π, если её область определения – это множество D_Π, а множеством положительных ответов является $D_\Pi \setminus Y_\Pi$. В гл. 2 мы отметили, что, вообще говоря, не известно, можно ли утверждать, что из принадлежности задачи Π классу *NP* следует, что $\Pi^c \in NP$. Далее, совсем не очевидно, что задачу Π можно полиномиальным образом свести к задаче Π^c из-за несимметричности ответов «да» и «нет». Однако совершенно тривиально, что задача Π сводится по Тьюрингу к Π^c (и обратно); поэтому, если задача Π является *NP*-полной или *NP*-трудной, то и Π^c должна быть *NP*-трудной.

Второе применение относится к таким переборным задачам, как оптимизационная задача о коммивояжере или задача о конструировании гамильтонова цикла, у которых соответствующие задачи распознавания *NP*-полны.

Всякий раз, когда можно воспользоваться полиномиальным алгоритмом переборной задачи для решения соответствующей задачи распознавания за полиномиальное время, существует сводимость по Тьюрингу между ними, и, следовательно, любой результат, касающийся *NP*-полноты задачи распознавания может быть переформулирован как результат, устанавливающий *NP*-трудность переборной задачи. Поэтому аргументы, представленные в гл. 2 в пользу того, что КОММИВОЯЖЕР не труднее, чем оптимизационная задача о коммивояжере, в совокупности с тем фактом, что КОММИВОЯЖЕР – это *NP*-полная задача, составляют доказательство того, что эта оптимизационная задача *NP*-трудна.

Ни один из этих примеров не иллюстрирует всех возможностей сводимости по Тьюрингу, поскольку в каждом из них требуется лишь однократное обращение к оракулу (или к соответствующей подпрограмме). Рассмотрим теперь пример, который в более полной мере использует сводимость по Тьюрингу.

K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО

УСЛОВИЕ. Заданы конечное множество A, размер $s(a) \in Z^+$ для каждого $a \in A$ и два неотрицательных числа $B \leqslant \sum_{a\in A} s(a)$ и $K \leqslant 2^{|A|}$.

ВОПРОС. Существует ли не менее K различных подмножеств $A' \subseteq A$ удовлетворяющих условию $s(A') \leqslant B$

$$\text{(где по определению } s(A') = \sum_{a\in A'} s(a))?$$

Было показано [322], что эта задача может быть решена за псевдополиномиальное время, точнее за время, ограниченное полиномиальной функцией от $|A| \cdot K\lceil \log s(a)\rceil$. Таким образом, для любого *фиксированного* значения K задачу можно решить за полиномиальное время. Вопрос состоит в том, чтобы выяснить, может ли она быть решена за полиномиальное время в общем случае.

Представляется, что эта задача не принадлежит классу P. Более того, совсем не очевидно, что она принадлежит *NP*, поскольку естественный путь её решения недетерминированным алгоритмом требует угадывания K подмножеств множества A и, по-видимому, нет способа записать угаданные K подмножеств в виде слова, длина которого будет полиномом от $|A|\cdot\lceil\log K\rceil\lceil\log s(A)\rceil$. С другой стороны, не известно, сводится ли (в смысле сводимости $\propto$) какая-либо *NP*-полная задача к этой задаче. Однако *Джонсон* и *Кашдан* [260] показали, что *NP*-полная задача РАЗБИЕНИЕ сводится по Тьюрингу к задаче K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО.

В самом деле, положим, что $S[A, s, B, K]$ – это подпрограмма для решения задачи K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО. При этом пусть $A = \{a_1, a_2, \dots, a_n\}$, $s : A \to Z^+$, $B \leqslant s(A)$, $K \leqslant 2^n$. Соответствующий алгоритм для решения задачи РАЗБИЕНИЕ начинается с вычисления величины $s(A)$, где параметры $A = \{a_1, a_2, \dots, a_n\}$, и $s : A \to Z^+$ описывают заданную индивидуальную задачу РАЗБИЕНИЕ. Если $s(A)$ нечетно, то алгоритм немедленно отвечает «нет» для этой индивидуальной задачи. В противном случае он полагает величину B равной $s(A)/2$ и применяет приводимую ниже дихотомическую процедуру, использующую введённую выше процедуру $S[A, s, B, K]$ и позволяющую определить число L^* подмножеств $A' \subseteq A$, удовлетворяющих неравенству $s(A') \leqslant b$.

Шаг 1. Положить $L_{min} \leftarrow 0$, $L_{max} \leftarrow 2^n$.

Шаг 2. Если $L_{max} - L_{min} = 1$, положить $L^* \leftarrow L_{min}$ и остановиться.

Шаг 3. Положить $L \leftarrow (L_{max} + L_{min})/2$ и вызвать подпрограмму $S[A, s, b, L]$. Если ответ «да», положить $L_{min} \leftarrow L$ и перейти к шагу 2. В противном случае положить $L_{max} \leftarrow L$ и перейти к шагу 2.

Эта процедура определяет L^*, вызывая подпрограмму S ровно n раз. Требуется ещё один дополнительный вызов для определения ответа в заданной задаче РАЗБИЕНИЕ, а именно вызывается подпрограмма $S[A, s, b-1, L^*]$. Если ответ в этой подпрограмме будет «да», то все подмножества

$A' \subseteq A$, удовлетворяющие неравенству $s(A') \leqslant b$, должны удовлетворять и условию $s(A') \geqslant b-1$, поэтому ответом к задаче РАЗБИЕНИЕ будет «нет». Если же ответом подпрограммы будет «нет», то должно существовать некоторое подмножество $A' \subseteq A$, для которого $s(A') = b$, поэтому ответ в задаче РАЗБИЕНИЕ будет «да».

Легко видеть, что описанная процедура была бы полиномиальным алгоритмом для задачи РАЗБИЕНИЕ, если бы процедура S была полиномиальным алгоритмом для задачи K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО. Таким образом, мы имеем желаемую тьюрингову сводимость задачи РАЗБИЕНИЕ к задаче K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО. Отсюда следует, что последняя задача NP-трудна и не может быть решена за полиномиальное время, если $P \neq NP$.

Основываясь на этом примере, легко понять, что можно использовать понятие NP-трудности для анализа сложности задач почти таким же образом, как ранее использовалось понятие NP-полноты. Все типы вопросов, обсуждаемых в гл. 4, применимы к NP-трудным задачам. В частности, можно говорить о сложности подзадач и о таких смежных вопросах, как псевдополиномиальные алгоритмы и «сильная» NP-трудность (определяемая аналогично сильной NP-полноте: переборная задача называется NP-трудной в сильном смысле, если она содержит NP-трудную подзадачу, удовлетворяющую полиномиальной границе для $Max[I]$). На эти результаты накладывается единственное ограничение: NP-полные задачи разрешимы за полиномиальное время *тогда и только тогда*, когда $P = NP$, в то время как все, что можно сказать о NP-трудной задаче, – это то, что она неразрешима за полиномиальное время, *если* $P \neq NP$.

Мы покажем, что и это различие часто можно устранить, что будет ещё одной иллюстрацией использования сводимости по Тьюрингу. Вспомним, что, когда мы говорили в гл. 2 о том, что КОММИВОЯЖЕР не труднее, чем соответствующая оптимизационная задача, мы упомянули также, что она в некотором смысле и «не легче». Смысл этого высказывания теперь можно пояснить.

Если рассмотреть пару задач, одна из которых – задача распознавания, а другая – соответствующая оптимизационная задача, то не только задача распознавания сводима по Тьюрингу к задаче оптимизации, но и задача оптимизации также сводима по Тьюрингу к задаче распознавания. Таким образом, каждая задача рассматриваемой пары может быть решена за полиномиальное время тогда и только тогда, когда другая задача этой пары полиномиально разрешима. Поскольку задача распознавания является NP-полной, то отсюда следует, что рассматриваемая оптимизационная задача может быть решена за полиномиальное время тогда и только тогда, когда $P = NP$.

Чтобы убедиться в этом, рассмотрим следующую задачу, занимающую промежуточное место между задачами распознавания и оптимизационными.

ДОСТРОЙКА МАРШРУТА КОММИВОЯЖЕРА (ДМК)

УСЛОВИЕ. Заданы конечное множество $C = \{c_1, c_2, \dots, c_m\}$ городов, расстояния $d(c_i, c_j) \in Z^+$ для каждой пары городов $c_i, c_j \in C$, граница $B \in Z^+$ и «частичный» маршрут $\Theta =< c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(K)} >$ между K различными городами из C, $1 \leqslant K \leqslant m$.

ВОПРОС. Можно ли так достроить Θ, чтобы получить маршрут $< c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(K)}; c_{\pi(K+1)}, ..., c_{\pi(m)} >$, проходящий через все города и имеющий длину не более B?

Легко видеть, что эта задача принадлежит *NP* и, следовательно, по определению *NP*-полноты, ДМК $\propto$ КОММИВОЯЖЕР. Поскольку сводимость $\propto$ – это лишь частный случай сводимости по Тьюрингу, отсюда сразу следует, что ДМК $\propto_T$ КОММИВОЯЖЕР. Обозначим через КО оптимизационную задачу о коммивояжере и покажем, что КО $\propto_T$ ДМК, тогда в силу транзитивности этого факта будет достаточно для доказательства того, что КО $\propto$ КОММИВОЯЖЕР.

Итак, положим, что $S[C, d, \Theta, B]$ – это подпрограмма для решения задачи ДМК, где параметр C обозначает множество городов, d – функция расстояний между парами городов из C, Θ – частичный маршрут, $B \in Z^+$ – граница. Пусть C и d определяют некоторую индивидуальную задачу ДМК, а B^* обозначает длину оптимального маршрута в этой индивидуальной задаче (какая бы она ни получилась). Поскольку каждый город должен войти в оптимальный маршрут и циклическая перестановка городов в маршруте не изменяет его длину, то существует оптимальный маршрут, начинающийся из города c_1. Далее, мы знаем, что величина B^* лежит между $B_{min} = m$ и $B_{max} = m \cdot (\max\{d(c_i, c_j) : c_i, c_j \in C\})$. Тогда, пользуясь дихотомической процедурой, такой же, как в задаче K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО, мы сможем найти величину B^*, обращаясь не более $\lceil \log_2 B_{max} \rceil$ раз к подпрограмме $S[C, d, < c_1 >, B]$, каждый раз придавая величине B различные значения.

Определив величину B^* и используя подпрограмму S, мы сможем построить оптимальный маршрут. Назовем последовательность Θ различных городов из C *продолжаемым частичным маршрутом*, если его можно продолжить до полного маршрута, имеющего общую длину B^*. Очевидно, что $< c_1 >$ – это продолжаемый частичный маршрут. Поскольку $< c_1 >$ допустим, существует по крайней мере один город $c_j \in C \setminus \{c_1\}$, такой, что $< c_1, c_j >$ – продолжаемый частичный маршрут. Мы можем найти этот город c_j, произведя не более $m-2$ вызовов подпрограммы S, которая каждый раз имеет вид $S[C, d, < c_1, c_j >, B^*]$, где $C_j \in C \setminus \{c_1\}$. (Если первые $m-2$ вызова не дают результата, то ясно, что оставшийся город *обязательно* подойдёт, поэтому для него не нужно применять нашу подпрограмму.)

В общем случае, если $< c_{\pi(1)}, ..., c_{\pi(K)} >$ – продолжаемый частичный маршрут, где $K < m$, то мы сможем найти другой продолжаемый частичный маршрут $< c_{\pi(1)}, ..., c_{\pi(K)}, c_{\pi(K+1)} >$, включающий ещё один город, при этом к подпрограмме S делается не более $m - K - 1$ обращений. Таким образом, можно построить полный маршрут для множества C, совершив не более $(m-1)(m-2)/2$ вызовов подпрограммы S (помимо тех вызовов, которые использовались для нахождения величины B^*). Поскольку в качестве подходящей функции $Length$ в задаче о коммивояжере можно взять функцию

$$Length[I] = m + \log_2 B_{max},$$

то очевидно, что описанная процедура даёт (полиномиальную) сводимость по Тьюрингу задачи КО к задаче ДМК. Следовательно, КО $\propto_T$ ДМК $\propto_T$ КОММИВОЯЖЕР, т. е. КО $\propto_T$ КОММИВОЯЖЕР, что и требовалось доказать.

Заметим, что главное звено здесь – это сводимость КО $\propto_T$ ДМК. Выяснив, что КО сводится по Тьюрингу к *какой-то* задаче из *NP*, мы получаем, что она может быть решена за полиномиальное время, если $P = NP$, и, следовательно, она «не труднее», чем *NP*-полные задачи. Мы назовем переборную задачу Π *NP-лёгкой*, если существует задача $\Pi' \in NP$, такая, что $\Pi \propto_T \Pi'$.

После того как читатель усвоил технику сводимости КО $\propto_T$ ДМК, ему будет не трудно понять, как этим же подходом можно воспользоваться, чтобы доказать *NP*-лёгкость многих переборных задач, особенно это относится к тем задачам, которым соответствуют задачи распознавания, являющиеся *NP*-полными. Можно определить задачу Π' из *NP* так, как это было сделано для задачи ДМК, с тем, чтобы последовательно построить требуемое решение (см., например, [520]). Для оптимизационных задач процедуре построения решения предшествует предварительная дихотомическая процедура, в процессе которой находится оптимальное значение целевой функции. В качестве упражнения мы предлагаем читателю попытаться доказать, что переборные задачи, соответствующие шести базовым *NP*-полным задачам из разд. 3.1, все являются *NP*-лёгкими, т. е. предлагается найти: (1) набор, присваивающий значение «истина», (2) трёхмерное сочетание, (3) вершинное покрытие минимальной мощности, (4) клику максимальной мощности, (5) гамильтонов цикл и (6) разбиение множества «взвешенных» элементов на два подмножества одинакового размера.

В действительности теперь мы видим, что изучение в основной теории только задач распознавания не привело к существенной потере общности, поскольку в большинстве случаев переборные задачи, у которых соответствующие задачи распознавания оказались *NP*-полными, сами все являются *NP*- лёгкими и, следовательно, эквивалентными по сложности. В то время как мы строили наш класс эквивалентности – класс *NP*-полных задач, одновременно получался значительно более широкий класс эквивалентных переборных задач, а именно тех задач, которые одновременно и *NP*-трудные и *NP*-лёгкие (и поэтому мы их называем *NP-эквивалентными*). Хотя этот более широкий класс содержит много задач, не принадлежащих *NP*, он сохраняет знакомое свойство, присущее *NP*-полным задачам: ни одна задача из этого класса не может быть решена за полиномиальное время, если $P \neq NP$, а если $P = NP$, то все задачи из этого класса могут быть решены за полиномиальное время.

5.2. Историческая справка

Заканчивая наше описание теории *NP*-полноты, уместно кратко остановиться на истории развития основных идей и терминологии, используемой для их описания.

В гл. 1 мы уже упоминали две ранние работы, в которых обсуждалось значение полиномиальной сложности. *Кобхэм* [80] рассматривал широкий спектр математических функций, вычислимых за полиномиальное время, и заметил, что класс всех таких функций не изменяется при различных моделях вычислений. *Эдмондс* [106] неформально отождествлял термин «хороший алгоритм» с понятием полиномиального алгоритма. В другой ранней работе [107] было введено неформальное понятие, близкое к классу *NP*. В этой работе предлагалось считать, что задача имеет «хорошую характеризацию», если для каждого её решения найдётся проверяемое за полиномиальное время «доказательство» того, что это действительно решение.

Все эти ранние работы были написаны в терминах функций или задач, которые мы назвали переборными. Фундаментальный вклад Кука состоял, в частности, в том, что он увидел возможность свести обсуждения к языкам и соответствующим задачам распознавания. В работе Кука [91] впервые были введены классы, которые теперь называют классами P и *NP*, и была доказана фундаментальная теорема (теорема Кука). Однако эта статья отличается от всех последующих работ в двух важных аспектах. Во-первых, основное понятие сводимости, введенное Куком и названное им «P-сводимостью», было близко к полиномиальной сводимости по Тьюрингу, а не к полиномиальной сводимости $\propto$. Поэтому в некоторых работах полиномиальную сводимость по Тьюрингу называют «сводимостью по Куку». Во-вторых, класс эквивалентных задач, которые Кук связал с задачей ВЫПОЛНИМОСТЬ (он не дал названия этому классу), не был ограничен задачами, сейчас называемыми *NP*-полными, а включал все задачи распознавания, которые мы назвали «*NP*-эквивалентными».

Теория *NP*-полноты приняла свой современный вид в работе *Карпа* [280], хотя некоторые термины появились позднее. Карп ввёл термины «класс P» и «класс *NP*» и заметил, что теорема Кука останется справедливой, если понятие сводимости по Тьюрингу заменить более простой и удобной полиномиальной сводимостью (transformability). Этот тип сводимости иногда называют «сводимостью по Карпу». Иногда его также называют «многооднозначной» сводимостью, поскольку она является многооднозначной функцией. Сам Карп просто пользовался термином «сводимость». Он дал современное определение «*NP*-полной задачи», хотя при этом пользовался термином «полиномиально полная задача» (термин «полный» восходит к теории рекурсивных функций, где язык называют полным по отношению к классу с данным типом сводимости, если он принадлежит этому классу и каждый элемент класса сводится к нему). В свете различия между двумя классами, определенными Куком и Карпом, ссылки на «класс Кука – Карпа», иногда встречающиеся в литературе, выглядят забавно (обычно при этом имеется в виду класс задач, определенный Карпом).

Статья Карпа, а также его сообщения на конференциях до публикации этой статьи вызвали широкий интерес. И естественно, что наряду с новыми научными результатами время от времени стали появляться новые термины. Вместо термина «polynomial complete» предлагалось писать «polynomially complete», несмотря на громоздкость последнего термина. Далее, *Сахни* [463] ввёл термин «P-полная задача», это был аналог «полиномиально полных задач», обобщённый на более широкий класс задач (аналогичных нашим «переборным задачам»). Он же предложил термин «P-трудная задача» для тех задач, которые по крайней мере столь же сложны, как NP-полные. В то же время *Левин* независимо доказал теорему, аналогичную теореме Кука, и ввёл термин «универсальная переборная задача» [340], который относится (так же как P-полные задачи, рассматриваемые Сахни) к более широкому классу задач, чем только задачи распознавания. В частности, используемое нами определение термина «переборная задача» заимствовано из работы [340]. Общепринятая в настоящее время терминология (NP-полная и NP-трудная задачи, полиномиальная сводимость) в большой степени является результатом деятельности *Дональда Кнута*. В 1973 г., подготавливая к печати четвертый том своей монографии «Искусство программирования» и нуждаясь в кратком обозначении задач, «которые по крайней мере столь же сложны, как полиномиально полные», Кнут провёл по своей инициативе опрос среди большой группы специалистов, занимающихся рассматриваемой тематикой. Он предложил им три варианта названия и просил выбрать наилучший с их точки зрения. При этом он отверг термин «P-трудная задача» из семантических соображений, поскольку если $P \neq NP$, то полиномиально полные задачи окажутся на самом деле намного труднее, чем задачи из P. Опрашиваемые специалисты в свою очередь отвергли все три названия, предложенные Кнутом, и выдвинули целый ряд своих названий, некоторые из них носили не совсем серьезный характер (например, *Шен Лин* предложил полиномиально полные задачи называть «PET-задачами», во-первых, потому, что это излюбленная тема его исследований [2], а во-вторых, потому, что сочетание PET можно рассматривать как сокращение для «Probably Exponential Time» (возможно экспоненциальное время)). Более того, если проблема $P = NP$ будет решена, предлагаемое сокращение получит новую интерпретацию: если $P \neq NP$, то PET можно трактовать как «provably exponential time» (доказуемо экспоненциальное время); если же $P = NP$, то PET может обозначать «previously exponential time» (время, ранее считавшееся экспоненциальным).

Интересный отчет об этом опросе и последовавших дискуссиях можно найти в статье Кнута [294]. В результате проведённого опроса наибольшее число голосов получили термины «NP-полная задача» и «NP-трудная задача» (оба они отсутствовали в первоначальном списке Кнута), при этом «NP-полная задача» заменила термин «полиномиально полная задача», а «NP-трудной задачей» была названа такая задача распознавания Π, что ВЫПОЛНИМОСТЬ $\propto \Pi$. Термин «NP-полная» был принят, поскольку он,

[2] *Pet*– излюбленная тема, излюбленный предмет. – *Прим. перев*

во-первых, короче, чем «полиномиально полная», и, во-вторых, он имеет больше смысла с теоретической точки зрения (*NP*-полная задача – это такая задача, которая полна в *NP* относительно полиномиальной сводимости). Аналогично термин «*NP*-трудная задача» имеет тот смысл, что это задача не проще, чем «самая трудная в *NP*».

С тех пор эти термины стали применяться почти повсеместно, хотя по-прежнему нередко появляется и термин «полиномиально полная задача». Термин «*NP*-полная задача» употребляется достаточно стабильно, а значение термина «*NP*-трудная задача» меняется. В заметке [295] Кнут предложил применять термин «полиномиальное сведение» (polynomial reduction) вместо термина «полиномиальное преобразование» (polynomial transformation). Кроме того, он предложил более широкое определение *NP*-трудной задачи как задачи распознавания, к которой ВЫПОЛНИМОСТЬ сводится по Тьюрингу (а не в смысле сводимости $\propto$). По этой причине, например, задачи, дополнительные к *NP*-полным, могут быть названы *NP*-трудными. Термин «*NP*-трудная задача» встречается в литературе и в том и в другом смысле, пожалуй, одинаково часто.

В этой книге мы сделали ещё один шаг по сравнению с работами Кнута, обобщив определение *NP*-трудной задачи, расширив его, а также понятие сводимости по Тьюрингу, чтобы включить переборные задачи. По этой причине новые термины найдут более широкое применение. Исходя из аналогичных соображений, мы ввели понятие «*NP*-лёгких» и «*NP*-эквивалентных» задач, причем в последнем понятии мы попытались формализовать кое-что из того, что Сахни имел в виду, введя «*P*-полные задачи», а Левин – «универсальные переборные задачи».

Кроме того, авторы ввели понятие «сильной *NP*-полноты» (разд. 4.2), причем впервые этот термин появился в работе [151]. Другое определение этого же понятия было предложено в [317], где задача «*NP*-полная в сильном смысле» называлась «унарной *NP*-полной», а задача «*NP*-полная в обычном смысле» называлась «бинарной *NP*-полной». Эта терминология основана на том факте, что если задача *NP*-полная в сильном смысле, то соответствующий язык является *NP*-полным даже в случае, когда кодирующая схема представляет числа в «неэкономном» унарном коде (т. е. последовательность из n единиц представляет число n).

За исключением нескольких новых терминов, введенных выше, и некоторых несущественных отличий в предлагаемых моделях ЭВМ, мы пытались следовать как можно ближе принятой в настоящее время терминологии, поэтому настоящая книга достаточно хорошо согласована с современной научной литературой. Читатель, выбравший эту книгу в качестве начального руководства для дальнейшего изучения литературы, без сомнения, найдёт ссылки на многочисленные периферийные вопросы, которые являются «точками роста» в теории *NP*-полноты. В последующих двух главах приводится материал по подобной смежной тематике, так что читатель может представить себе, о каких вопросах идет речь, и, возможно, сам займётся изучением тех вопросов, которые покажутся ему наиболее интересными.

Глава 6.

Подходы к решению NP-полных задач

Вернёмся на короткое время в отдел бандерснечей из разд. 1.1. Напомним, что мы покинули его в несколько неопределённой ситуации. Будучи главным разработчиком алгоритмов, вы удачно избежали возможного обвинения в некомпетентности, доказав NP-полноту задачи о бандерснече. Однако после произнесения этих магических слов задача о бандерснече не исчезла, и вы по-прежнему стоите пред проблемой отыскания приемлемого алгоритма её решения.

В настоящей главе будут обсуждаться некоторые возможные подходы к этой проблеме, главным образом новое направление, связанное с получением доказуемых «оценок погрешности» алгоритмов. Однако, прежде чем перейти к рассмотрению конкретного материала, желательно сделать краткий обзор возможных альтернатив, которые в первом приближении можно было бы разбить на две общие категории.

К первой категории относятся подходы, в которых делается попытка максимального сокращения объёма перебора, хотя при этом и признается неизбежность экспоненциального времени работы. К наиболее широко используемым приёмам сокращения перебора относятся приёмы, основанные на методе «ветвей и границ» или методе «неявного перебора» (см., например, [161]). Эти приёмы состоят в построении «частичных решений», представленных в виде дерева поиска, и применении мощных методов построения оценок, позволяющих опознать бесперспективные частичные решения, в результате чего от дерева поиска на одном шаге отсекается целая ветвь. Известны и другие подходы, когда процесс поиска организован иначе (они используются иногда совместно с методом ветвей и границ). К ним относятся метод динамического программирования (применявшийся в гл. 4 для построения алгоритмов с псевдополиномиальным временем работы), методы отсечений (см., например, [226, 161]) и метод Лагранжа (см., например, [170, 206]). Кроме того, в худшем случае иногда оказывается возможным в значительной степени уменьшить сложность полного перебора за счет более удачного выбора объектов, к которым он применяется. К относительно недавним примерам использования этого подхода относятся алгоритмы решения задачи РАЗБИЕНИЕ, задачи РАСКРАШИВАЕМОСТЬ ГРАФА В K ЦВЕТОВ [325] и задачи НЕЗАВИСИМОЕ МНОЖЕСТВО [513].

Подходы, относящиеся ко второй категории, применимы исключительно к оптимизационным задачам (это однако не является слишком сильным ограничением, поскольку огромное количество задач, возникающих на практике, естественно формулируются как оптимизационные) и включают приём, который можно назвать «снижение требований». Он заключается в отказе от поиска оптимального решения и в нахождении вместо этого «хорошего» решения за приемлемое время. Алгоритмы, основанные на этом приёме, обычно называются «эвристическими», поскольку они используют различные разумные соображения без строгих обоснований. Методы, применяемые для построения алгоритмов такого типа, сильно зависят от особенностей задачи, хотя и удалось выделить несколько руководящих принципов, которые могут служить полезной отправной точкой (см. [347], где проводится блестящее обсуждение этих принципов). Наиболее широко применяется так называемый метод «локального поиска», когда заранее выбранное множество локальных операций используется для последовательного улучшения начального решения до тех пор, пока такое улучшение возможно, в противном случае оказывается достигнутым «локальный оптимум».

Эвристические алгоритмы, построенные этим и другими методами, на практике оказываются весьма удовлетворительными, хотя для получения удовлетворительных характеристик алгоритма обычно требуется довольно большая работа по его «доводке». В результате удаётся только в очень редких случаях заранее предсказать и оценить поведение таких алгоритмов. Вместо этого такие алгоритмы оцениваются и сравниваются на основе сочетания эмпирических данных и аргументов, опирающихся на здравый смысл.

Однако совсем недавно был получен ряд результатов, которые показали, что эвристические алгоритмы не всегда столь сложны для формального анализа. В некоторых случаях удаётся доказать, что решения, получаемые эвристическим алгоритмом, всегда будут отличаться в процентном отношении от оптимального решения не более чем на определённую величину. Подобные результаты можно рассматривать как «оценки погрешности» алгоритмов. В настоящей главе мы остановимся именно на таких результатах.

В разд. 6.1 сделан обзор различных типов оценки погрешности алгоритмов. В разд. 6.2 показано, как теория NP-полноты позволяет дать заключение о наилучшей из возможных для данной задачи оценке погрешности. Наконец, в разд. 6.3 обсуждаются некоторые практические соображения, касающиеся возможностей применения результатов этого типа, а также упоминаются некоторые теоретические работы, в которых делается попытка проанализировать «поведение в среднем» эвристических алгоритмов.

6.1. Оценки погрешности приближённых алгоритмов

Дадим вначале формальное описание понятия «оптимизационная задача».

Комбинаторная оптимизационная задача Π есть либо задача минимизации, либо задача *максимизации* и состоит из следующих трёх частей:

(1) множества D_Π *индивидуальных задач*;

(2) для каждой $I \in D_\Pi$ конечного множества $S_\Pi(I)$ *допустимых решений* индивидуальной задачи I;

(3) функции m_Π, сопоставляющей каждой индивидуальной задаче $I \in D$ и каждому *допустимому решению* $\sigma \in S_\Pi(I)$ некоторое положительное целое число $m_\Pi(I, \sigma)$, называемое *величиной решения* σ.

Если Π – задача минимизации (соответственно максимизации), то *оптимальным решением* индивидуальной задачи $I \in D_\Pi$ является такое допустимое решение $\sigma^* \in S_\Pi(I)$, что для всех $\sigma \in S_\Pi(I)$ выполнено неравенство $m_\Pi(I, \sigma^*) \leqslant m_\Pi(I, \sigma)$ (соответственно $m_\Pi(I, \sigma^*) \geqslant m_\Pi(I, \sigma)$). Для обозначения величины $m_\Pi(I, \sigma^*)$ оптимального решения индивидуальной задачи I будет использоваться символ $OPT_\Pi(I)$ (в тех случаях, когда задача ясна из контекста, индекс Π будет опускаться).

Алгоритм A называется *приближённым алгоритмом* решения задачи Π, если для любой индивидуальной задачи $I \in D_\Pi$ алгоритм A отыскивает некоторое допустимое решение $\sigma \in S_\Pi(I)$. Через $A(I)$ будем обозначать величину $m_\Pi(I, \sigma)$ того возможного решения σ, которое A строит по I. Если $A(I) = OPT(I)$ для всех $I \in D_\Pi$, то A называется точным алгоритмом решения задачи Π.

Приведённые определения можно проиллюстрировать на примере уже известной нам задачи о коммивояжере. Это задача минимизации, множество её индивидуальных задач состоит из всех конечных множеств городов и попарных расстояний между ними. Допустимые решения конкретной индивидуальной задачи представляют собой все возможные перестановки заданных городов. Величина решения для каждой такой перестановки равна длине соответствующего маршрута. Таким образом, приближённый алгоритм для данной задачи должен отыскать только некоторую перестановку городов, а точный алгоритм должен всегда отыскивать перестановку, соответствующую маршруту минимальной длины.

Если оптимизационная задача NP-трудна (что имеет место в данном случае), то, как нам известно, нельзя построить точный полиномиальный алгоритм решения этой задачи (если $P \neq NP$). Более реалистично попытаться найти приближённый алгоритм A, время работы которого ограничено полиномом невысокой степени и который обладает тем свойством, что для всех индивидуальных задач I величина $A(I)$ «близка» к величине $OPT(I)$. Следующий пример иллюстрирует типы результатов, которыми мы будем интересоваться в дальнейшем.

Рассмотрим задачу об «упаковке в контейнеры». Она формулируется следующим образом: Задано конечное множество $U = \{u_1, u_2, \dots, u_n\}$ «предметов» и «размеры» $s(u) \in [0,1]$ каждого предмета $u \in U$ (размер предмета представлен рациональным числом); требуется найти такое разбиение множества U на непересекающиеся подмножества $U_1, U_2, \dots, U_k$, чтобы сумма размеров предметов в каждом подмножестве U_i не превосходила 1 и чтобы k было наименьшим возможным. Можно считать, что предметы, принадлежащие каждому множеству U_i, упаковываются в один контейнер размера 1, а наша цель – упаковать предметы множества U в как можно меньшее число контейнеров.

Рассматриваемая задача NP-трудна в сильном смысле (в частном случае она даёт задачу 3-РАЗБИЕНИЕ), так что мало надежды на отыскание даже псевдополиномиального точного алгоритма решения этой задачи. Однако для её решения имеется несколько заслуживающих внимания простых приближённых алгоритмов.

Один из них известен под названием «в первый подходящий». Предположим, что имеется бесконечная последовательность контейнеров размера 1, каждый из которых в начальный момент времени пуст. Алгоритм заключается в том, чтобы помещать предметы в контейнеры по очереди, в порядке возрастания номеров. Предметы помещаются в контейнеры согласно следующему простому правилу: «Очередной предмет u_i помещается в контейнер с наименьшим номером, у которого сумма размеров помещённых в него предметов не превосходит $1-s(u_i)$». Другими словами, u_i помешается в первый из контейнеров, куда он может попасть (не нарушая ограничения по размеру). На рис. 6.1 показан пример; здесь каждый предмет представлен прямоугольником, имеющим высоту, пропорциональную размеру предмета.

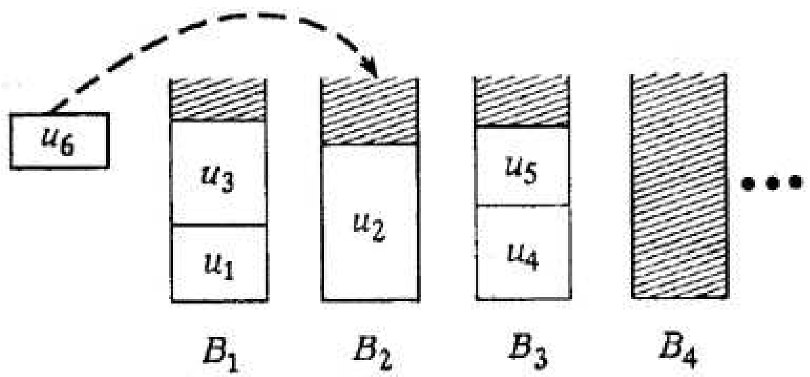


Рис. 6..1: *Пример размещения по правилу «в первый подходящий», где предмет u_6 кладётся в контейнер B_2, поскольку это первый контейнер, в который u_6 может поместиться.*

С интуитивной точки зрения этот алгоритм кажется очень естественным и разумным. Он не начинает заполнять новый контейнер до тех пор, пока все непустые контейнеры не будут слишком сильно заполнены. Что можно сказать о качестве работы этого алгоритма?

Вначале рассмотрим число контейнеров, требующихся для указанного алгоритма, как функцию от параметров задачи. Обозначим эту функцию через $FF(I)$.[1] Имеют место неравенства:

$$FF(I) \geqslant 1$$

$$FF(I) < \left\lceil 1 \sum_{i=1}^{n} s(u_i) \right\rceil .$$

Последнее неравенство вытекает из того, что в результате работы алгоритма не может получиться более одного контейнера, заполненного менее чем на половину. (В противном случае первый же предмет, попавший в такой контейнер с бо́льшим номером, должен был быть помещён алгоритмом «в первый подходящий», т.е. такой же контейнер с меньшим номером.) То, что указанная граница – наилучшая из возможных, следует из рассмотрения индивидуальной задачи $U = \{u_1, u_2, ..., u_n\}$, где $s(u_i) = 1/2 + \varepsilon$, $1 \leqslant i \leqslant n$. Поскольку никакие два предмета не могут попасть в один ящик, то $FF(I) = n$, в то время как сумма размеров предметов равна $n/2 + n\varepsilon$, и если выбрать ε достаточно малым, она будет как угодно близка к $n/2$.

Сделанное замечание позволяет нам также оценить, во сколько раз плохое решение, получаемое алгоритмом «в первый подходящий», может отличаться от оптимального решения задачи. Поскольку ясно, что

$$OPT(I) \geqslant \left\lceil \sum_{i=1}^{n} s(u_i) \right\rceil ,$$

то отсюда вытекает, что для всех индивидуальных задач I выполнено неравенство

$$FF(I) < 2 \cdot OPT(I).$$

Однако, как следует из приведённой ниже теоремы, для $FF(I)$ имеется лучшая оценка, см. [265].

Теорема 6.1. *Для всех индивидуальных задач I об упаковке имеет место неравенство*

$$FF(I) \leqslant \frac{17}{10} OPT(I) + 2.$$

Более того, существуют индивидуальные задачи I, для которых $OPT(I)$ сколь угодно велико и

$$FF(I) \geqslant \frac{17}{10}(OPT(I) - 1).$$

[1] FF – первые буквы английский слов *First Fit* (первый подходящий). – *Прим. перев.*

Таким образом, теорема 6.1 характеризует асимптотическое поведение в худшем случае алгоритма «в первый подходящий». Величина $FF(I)$ никогда не отличается от оптимума более чем на 70%, и в некоторых случаях такое отличие действительно достигается. (Небольшое улучшение верхней оценки, благодаря которому $(\frac{17}{10})OPT(I) + 2$ заменяется на $\lceil(\frac{17}{10}OPT(I)\rceil$, получено в работе [144].) Мы опустим слишком длинное доказательство теоремы 6.1, однако заметим, что близкая оценка поведения алгоритма в худшем случае может быть получена из рассмотрения примеров, представленных на рис. 6.2, для которых $FF(I) = (\frac{5}{3})OPT(I)$. (Доказательство нижней оценки в теореме 6.1 получается в результате некоторого усложнения этих примеров.)

Индивидуальная задача I :

$$U = \{u_1, u_2, \dots, u_{18m}\}, s(u_i) = \begin{cases} \frac{1}{7} + \varepsilon & 1 \leqslant i \leqslant 6m \\ \frac{1}{3} + \varepsilon & 6m \leqslant i \leqslant 12m \\ \frac{1}{2} + \varepsilon & 12m \leqslant i \leqslant 18m \end{cases}$$

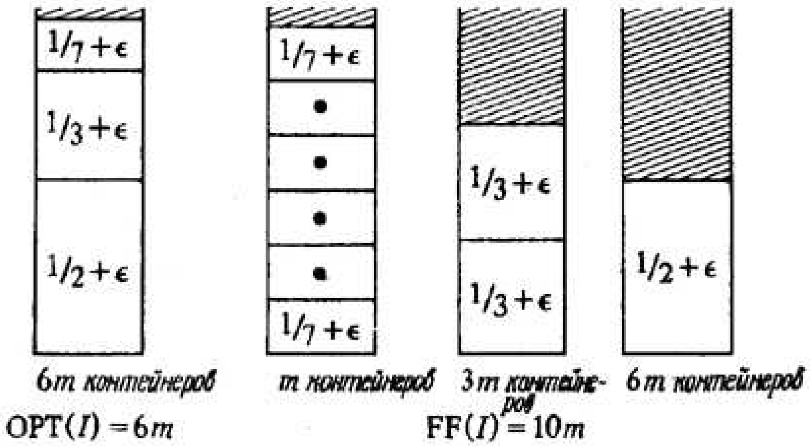


Рис. 6..2: *Индивидуальные задачи I с произвольно большим значением $OPT(I)$, для которых $FF(I)$ равно $\frac{5}{3}OPT(I)$.*

Полученные результаты являются основой для анализа приближённых алгоритмов решения задачи об упаковке. Теперь можно перейти к анализу других алгоритмов с лучшими оценками поведения. Очевидно, алгоритм «в первый подходящий» можно видоизменить, пользуясь, например, следующим более совершенным правилом размещения: каждый следующий предмет u_i помещаем в тот контейнер, содержимое которого ближе всего к величине $1 - s(u_i)$, но не превосходит её (если имеется несколько контейнеров, обладающих этим свойством, то выбираем из них контейнер с наименьшим номером). Этот алгоритм известен под названием «в наилучший из подходящих». К сожалению (и к удивлению многих), алгоритм «в наилучший из подходящих» в худшем случае имеет, по существу, те же характеристики, что и алгоритм «в первый подходящий», см. [265].

Несколько лучший приближённый алгоритм можно получить, если учесть, что наихудшей для алгоритма «в первый подходящий» (а также для алгоритма «в наилучший из подходящих») оказывается ситуация, когда в упорядочении предметов предметы с меньшими размерами идут раньше предметов с бóльшими размерами. Предположим, что предметы из U упорядочены не произвольно, а в порядке уменьшения их размеров и перенумерованы так, что $s(u_1) \geqslant s(u_2) \geqslant \ldots \geqslant s(u_n)$. Алгоритм применения процедуры «в первый подходящий» к переупорядоченному таким образом списку предметов называется «в первый подходящий в порядке убывания» (соответствующая функция обозначается $FFD(I)$ [2]. Работа этого алгоритма характеризуется следующей теоремой Джонсона [262] (набросок доказательства см. в [265]).

Теорема 6.2. *Для всех индивидуальных задач об упаковке выполняется неравенство*

$$FFD(I) \leqslant \frac{11}{9} OPT(I) + 4.$$

Более того, существуют индивидуальные задачи I, для которых $OPT(I)$ произвольно велико и

$$FFD(I) \geqslant \frac{11}{9} OPT(I).$$

Таким образом, мы имеем гарантию, что алгоритм «в первый подходящий в порядке убывания» даже в худшем случае выдаёт решение, отличающееся от оптимального не более чем на 22%. Кроме того, встречаются индивидуальные задачи, для которых эта оценка достигается. Такой же результат имеет место для аналогичного алгоритма «в наилучший из подходящих в порядке убывания». На рис. 6.3 показан класс примеров, позволяющих установить нижнюю оценку для обоих алгоритмов.

Доказательство оценки сверху заключается в тщательном рассмотрении различных случаев, даже краткое изложение которых потребовало бы больше места, чем отведено для всей настоящей главы. (Хотя длинные доказательства для задач, подобных задаче об упаковке, – обычное явление, отметим, что аналогичные результаты для других задач были получены без геркулесовых усилий, а для задачи об упаковке несколько более слабые оценки можно получить гораздо проще.)

Дальнейшие модификации алгоритма «в первый подходящий в порядке убывания» были предложены в работах [262, 542]. Цель этих модификаций состояла в получении полиномиального по времени приближённого

[2] Буква D в *FFD* – это сокращение от *Decreasing*, т.е. в порядке уменьшения. – *Прим. ред.*

Индивидуальная задача I :

$$U = \{u_1, u_2, \dots, u_{30m}\}, s(u_i) = \begin{cases} \frac{1}{2} + \varepsilon & 1 \leqslant i \leqslant 6m \\ \frac{1}{4} + 2\varepsilon & 6m \leqslant i \leqslant 12m \\ \frac{1}{4} + \varepsilon & 12m \leqslant i \leqslant 18m \\ \frac{1}{4} - 2\varepsilon & 18m \leqslant i \leqslant 30m \end{cases}$$

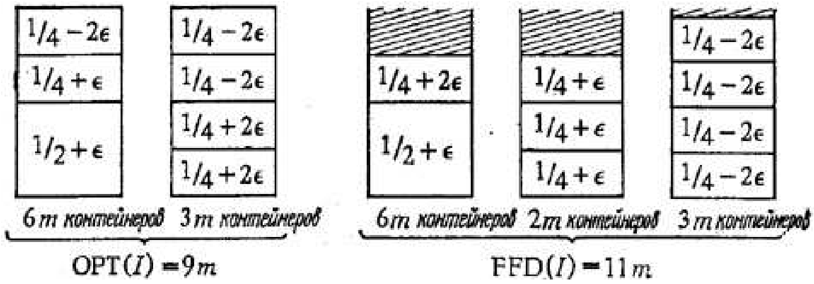


Рис. 6..3: *Индивидуальные задачи I с произвольно большим значением $OPT(I)$, для которых $FFD(I)$ равно $(\frac{11}{9})OPT(I)$.*

алгоритма решения задачи об упаковке с ещё лучшими оценками решения. Однако ни для одной из этих модификаций пока не получена существенно лучшая оценка.

Итак, проведённый анализ приближённых алгоритмов решения задачи об упаковке, можно подытожить следующим образом. Мы начали анализ с изучения оценки погрешности очевидного, но весьма разумного алгоритма. Этот анализ проводился посредством получения оценок работы алгоритма в худшем случае и посредством построения примеров, показывающих, что эти оценки не могут быть улучшены. На основе этого анализа, а также интуитивных соображений о недостатках исходного алгоритма мы стали искать другие алгоритмы (возможно, представляющие собой всего лишь более сложные версии исходного алгоритма) и анализировать их. Оценки погрешностей алгоритмов были выражены в виде отношений, представляющих интерес с точки зрения возможности их сравнения и в некотором смысле отражающих близость результатов к оптимуму. Этот общий подход может служить моделью изучения NP-трудных оптимизационных задач и в действительности широко применяется (хотя, конечно, иногда могут оказаться более адекватными и легче доказуемыми оценки другого типа, см., например, [96,399]).

Для формализации этого подхода дадим ещё несколько определений. Пусть Π – задача минимизации (соответственно максимизации), а I – произвольная индивидуальная задача из D_Π. Определим отношение $R_A(I)$ следующим образом:

$$R_A(I) = \frac{A(I)}{OPT(I)}$$

(соответственно $R_A(I) = \frac{OPT(I)}{A(I)}$). *Погрешностью приближённого алгоритма* A решения задачи Π назовем величину R_A, определяемую следующим образом: [3]

$$R_A = inf \left\{ r \geqslant 1 : \ R_A(I) \leqslant r \ \begin{array}{c} \text{для всех индивидуа-} \\ \text{льных задач } I \in D_\Pi; \end{array} \right\}.$$

Асимптотической погрешностью приближённого алгоритма A назовём величину

$$R_A^\infty = inf \left\{ r \geqslant 1 : \begin{array}{l} \text{найдётся } N \in Z^+, \text{ такое, что } R_A(I) \leqslant r \\ \text{для всех } I \in D_\Pi, \text{для которых выполнено} \\ \text{соотношение } OPT(I) \geqslant N \end{array} \right\}.$$

Заметим, что отношения, определённые выше, обладают тем свойством, что отношение для задачи минимизации обратно отношению для задачи максимизации. Благодаря этому мы получаем однородную шкалу для градации приближённых алгоритмов решения задач разных типов. При этом всегда имеют место соотношения $1 \leqslant R_A \leqslant \infty$ и $1 \leqslant R_A^\infty \leqslant \infty$, причём чем отношение ближе к 1, тем меньше погрешность.

Отметим также, что R_A не обязательно равно R_A^∞. Так, например, теорема 6.2 утверждает, что $R_{FFD}^\infty = \frac{11}{9}$, но легко построить пример такой индивидуальной задачи I, что $OPT(I) = 2$, и $FFD(I) = 3$, и, значит, $R_{FFD} \geqslant \frac{3}{2}$. Для задачи об упаковке асимптотическая погрешность, по-видимому, важнее, хотя для задач другого типа более адекватной величиной может быть $R_A(I)$. Может оказаться, что для интересующих нас приближённых алгоритмов обе характеристики совпадают: $R_A = R_A^\infty$. В любом случае полезно знать обе эти величины, а различие между ними учитывать при анализе приближённого алгоритма.

В качестве другого примера рассмотрим ещё раз задачу о коммивояжере, введя в её условие дополнительное ограничение. Индивидуальная задача I по-прежнему представляет собой множество C городов и список расстояний между ними. В данном случае, однако, потребуем, чтобы расстояния подчинялись «неравенству треугольника», т.е. чтобы для каждой тройки городов a, b, c из множества C было выполнено соотношение

$$d(a, c) \leqslant d(a, b) + d(b, c).$$

[3] Для обозначения величины $R_A(I)$ авторы употребляют термин «absolute performance ratio». Хотя $R_A(I)$ и безразмерная величина, она, однако, не является относительной погрешностью в обычном смысле (определяемой как $\left|\frac{A(I)-OPT(I)}{OPT(I)}\right|$). Избегая значительной переделки текста при переводе, мы предпочли для величины $R_A(I)$ термин «погрешность», хотя и сознавали компромиссность этого решения. – *Прим. ред.*

Это условие выполняется, например, когда расстояния между городами представляют собой кратчайшие пути в некоторой стандартной метрике или если допускаются маршруты, проходящие через некоторые города по нескольку раз, поскольку в последнем случае каждое из заданных расстояний $d(c_i, c_j)$ может быть заменено длиной кратчайшего пути от c_i к c_j. Нетрудно видеть, что и при указанных ограничениях задача остается *NP*-трудной. Более того, для всех алгоритмов, которые мы рассмотрим в дальнейшем, будет выполнено соотношение $R_A = R_A^\infty$, так что мы ограничимся рассмотрением величины R_A.

Рассмотрим естественный эвристический алгоритм, называемый «ближайший сосед». Этот алгоритм обозначается NN [4] и описан, например, в работе [162]. Пусть $C = \{c_1, c_2, \dots, c_m\}$ – заданное множество городов. В качестве первого города $c_{x(i)}$ в искомом маршруте возьмем город c_1. Если частичный маршрут $< c_{\pi(1)}, c_{\pi(2)}, \dots, c_{\pi(k)} >$, $k < m$, уже построен, то в качестве города $c_{\pi(k+1)}$ алгоритм выбирает город c, ближайший к городу $c_{\pi(k)}$, (т.е. город с наименьшим расстоянием $d(c_{\pi(k)}, c)$), из городов, ещё не принадлежащих маршруту (если имеется несколько таких городов, можно выбрать город с наименьшим номером). Следующая теорема Розенкрантца, Стирнза и Льюиса [457] показывает, что этот приближённый алгоритм может вести себя гораздо хуже, чем любой из обсуждавшихся нами алгоритмов для решения задачи об упаковке.

Теорема 6.3. *Для индивидуальных задач о коммивояжере I с m городами и неравенством треугольника выполнено неравенство*

$$NN(I) \leqslant \frac{1}{2}(\lceil \log_2 m \rceil + 1) OPT(I).$$

Более того, для произвольно больших значений m существуют такие индивидуальные задачи I с m городами, что

$$NN(I) > \frac{1}{3}\left(\log_2(m+1) + \frac{4}{3}\right) OPT(I).$$

Наиболее важное утверждение этой теоремы можно сформулировать очень кратко: $R_{nn} = \infty$; оно показывает не слишком большую перспективность алгоритма «ближайший сосед». Пример сложной рекурсивной конструкции, используемой для получения нижней оценки теоремы 6.3, при $m = 15$ представлен на рис. 6.4.

Погрешность алгоритма «ближайший сосед» оставляет желать много лучшего. Мы упомянули об этом алгоритме лишь для того, чтобы показать, что даже весьма разумные эвристики могут приводить к плохим результатам, отличающимся от оптимума в произвольно большое число раз.

К счастью, другие приближённые алгоритмы решения этой задачи работают гораздо лучше. В самом деле, в работе Розенкрантца, Стирнза и

[4]NN –от английских слов *Nearest Neighbor* (ближайший, сосед).– *Прим. перев.*

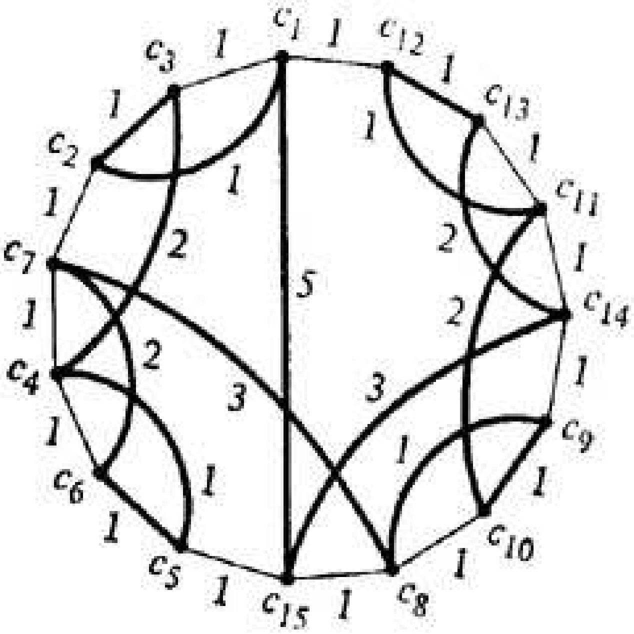


Рис. 6..4: *Схематическое изображение индивидуальной задачи о коммивояжере, для которой алгоритм «ближайший сосед» работает плохо. В этом примере имеется 15 городов, расстояние $d(c_i, c_j)$ определяется как длина кратчайшего пути от c_i к c_j, получаемого с использованием рёбер, изображенных на рисунке. Нетрудно проверить, что эти расстояния подчиняются неравенству треугольника. Периметр даёт оптимальный маршрут длины 15, а алгоритм «ближайший сосед» выдаёт зачернённый маршрут длины 27. Таким образом, $R_{NN}(I) = \frac{27}{25} > \frac{16}{9} = \frac{1}{3}(\log_2(m+1)+\frac{4}{3})$.*

Льюиса [457] приведено несколько приближённых алгоритмов с погрешностью $R_A = 2$. Хорошо известен обладающий этим свойством алгоритм, основанный на понятии «минимальное остовное дерево». Рассмотрим индивидуальную задачу о коммивояжере, представленную в виде полного графа, вершинами которого являются города, а каждое ребро $\{c_i, c_j\}$ имеет «длину» $d(c_i, c_j)$. В таком графе имеется очевидное взаимно-однозначное соответствие между всевозможными машрутами и гамильтоновыми циклами. Остовным деревом называется связный подграф, содержащий все вершины и не содержащий циклов (т. е. между любыми двумя вершинами имеется в точности один путь без повторяющихся рёбер). По индукции легко доказывается, что в связном графе на m вершинах остовное дерево должно иметь $m-1$ ребро. Минимальным остовным деревом называется остовное дерево, сумма длин рёбер которого минимальна.

В отличие от маршрутов коммивояжера минимальной длины минимальные остовные деревья можно легко построить, затратив при этом время, ограниченное полиномом невысокой степени (см. [308] или [9]). Более того, длина минимального остовного дерева для графа задачи о коммивояжере должна быть меньше длины минимального маршрута, потому что остовное дерево (более короткое, чем этот маршрут) может бытъ получено из него путём отбрасывания одного ребра. Это наводит на мысль, что достаточно короткий маршрут можно получить построением минимального остовного дерева с последующим посещением всех городов в результате двойного обхода дерева. На рис. 6.5 (б) такой обход указан для дерева с рис. 6.5 (а). В результате такого обхода некоторые города приходится посещать более одного раза, однако полученный маршрут всегда можно сократить, если каждый раз идти сразу в следующий ещё не посещённый город, как показано на рис. 6.5 (в). Неравенство треугольника обеспечивает, чтобы в результате такого сокращения длина маршрута не увеличивалась.

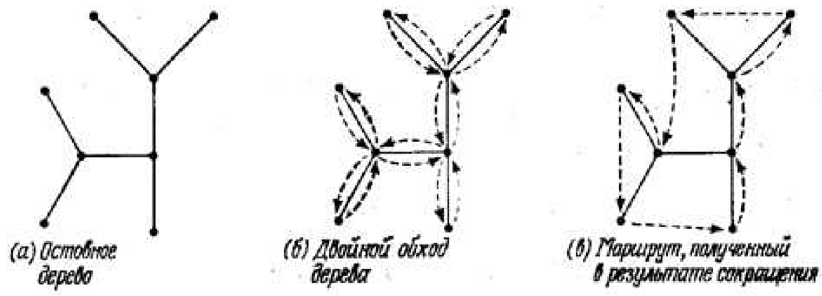


Рис. 6..5: *Преобразование остовного дерева в маршрут для коммивояжера с использованием алгоритма «двойной обход дерева».*

Таким образом, длина маршрута, получаемого в результате изложенной выше процедуры, не превосходит удвоенной длины минимального остовного дерева, что строго меньше удвоенной длины оптимального маршрута. Обозначим алгоритм «двойного обхода минимального остовного дерева» через MST[5]. Только что доказанный результат формулируется следующим образом:

Теорема 6.4. *Для всех индивидуальных задач о коммивояжере I, для которых выполняется неравенство треугольника, имеет место неравенство*

$$MST(I) < 2 \cdot OPT(I).$$

Примеры, показывающие, что полученная для алгоритма MST оценка наилучшая из возможных и, следовательно, $R_{MST} = 2T$, построить нетрудно, и мы оставляем это в качестве упражнения.

[5] От английских слов *Minimum Spanning Tree* (минимальное остовное дерево).– *Прим. перев.*

Идея, лежащая в основе алгоритма MST, была развита *Кристофидисом* [71], в результате чего был построен эвристический алгоритм решения этой задачи с ещё меньшей погрешностью. Для этого используется комбинация техники «паросочетаний» и понятий «эйлеров граф» и «эйлеров маршрут». *Эйлеровым графом* называется граф, у которого все вершины имеют чётные степени. *Эйлеровым маршрутом* в графе называется цикл, проходящий ровно по одному разу через каждое ребро. Нетрудно показать, что для того, чтобы граф G был эйлеровым, необходимо и достаточно существование в графе G эйлерова маршрута (см., например, [357]). Более того, легко построить полиномиальный алгоритм отыскания в таком графе эйлерова маршрута. Кристофидис заметил, что алгоритм MST можно расчленить на четыре стадии: (1) отыскание минимального остовного дерева, (2) преобразование остовного дерева в эйлеров граф посредством дублирования всех рёбер графа, (3) отыскание в построенном графе эйлерова маршрута, (4) преобразование с помощью сокращения полученного эйлерова маршрута в маршрут для коммивояжера. Поскольку общая длина добавляемых рёбер равна длине минимального остовного дерева (и, следовательно, меньше минимальной длины маршрута для коммивояжера) и так как неравенство треугольника гарантирует, что замена двух рёбер одним не увеличит длины маршрута, то длина получающегося маршрута может превосходить минимально возможную длину маршрута самое большее в два раза. Однако Кристофидис показал, что остовное дерево можно превратить в эйлеров граф с ещё меньшими издержками.

Это можно сделать, сосредоточив внимание на множестве $V' = \{a_1, a_2, \ldots, a_{2k}\}$ вершин остовного дерева, имеющих нечётную степень (таких вершин должно присутствовать чётное число). *Паросочетанием* в множестве V' называется его разбиение на k 2-элементных подмножеств. *Весом* такого паросочетания называется сумма расстояний $d(c_i, c_j)$, где $\{c_i, c_j\}$ – подмножество рассматриваемого паросочетания и $i < j$ (для того чтобы избежать дублирования). Любое паросочетание для V' даёт нам k рёбер, добавление которых к остовному дереву превращает его в эйлеров граф. Общая длина этих рёбер равна весу паросочетания. *Паросочетанием минимального веса* называется паросочетание, вес которого минимален среди весов всех возможных паросочетаний. Паросочетания минимального веса могут быть найдены за полиномиальное время стандартными методами (см., например, [324]). В приближённом алгоритме Кристофидиса стадия 2 алгоритма MST заменяется процедурой отыскания минимального паросочетания для множества V' с последующим превращением остовного дерева в эйлеров граф посредством добавления соответствующих рёбер.

Основное наблюдение, позволяющее оценить погрешность этого алгоритма, состоит в том, что вес паросочетания минимального веса не может превосходить половины длины минимального маршрута для коммивояжера. В этом можно убедиться следующим образом. Превратим вначале маршрут минимальной длины в маршрут только на вершинах из V'. Этого можно добиться, исключив из заданного маршрута вершины, не принадлежащие V'. В результате, в силу неравенства треугольника, получится маршрут, длина которого не превосходит длины исходного. Построенный маршрут задаёт на множестве V' два паросочетания, которые получаются, если брать рёбра в маршруте через одно. Наименьший из весов этих двух паросочетаний не превосходит половины длины маршрута. Обозначив построенный алгоритм через MM [6] получаем следующую теорему.

Теорема 6.5. *Для всех индивидуальных задач о коммивояжере, для которых выполняется неравенство треугольника, имеет место неравенство*

$$MM(I) < \frac{3}{2}OPT(I).$$

Как и раньше, нетрудно построить примеры, показывающие, что эта оценка, по существу, наилучшая возможная (см., например, [97]) и поэтому $R_{MM} = \frac{3}{2}$. В настоящее время не известен приближённый полиномиальный алгоритм решения задачи о коммивояжере с неравенством треугольника, погрешность которого была бы меньше.

Полученные выше результаты для задачи о коммивояжере, подобно результатам для задачи об упаковке, представляют собой характерный пример того, на что можно надеяться при решении трудных задач полиномиальными алгоритмами. В обеих задачах имеется гарантия отыскания приближённых решений, довольно близких к оптимальному, хотя ни для одной из них наилучшая оценка погрешности решения не так близка к 1, как хотелось бы.

К сожалению, имеется большое число *NP*-трудных задач другого типа, для которых не известны полиномиальные алгоритмы, даже с такой оценкой погрешности. Один из примерев подобных задач – задача о раскраске графа: «Задан граф $G = (V, E)$, требуется найти такую функцию $f : V \to \{1, 2, \dots, k\}$, что $f(u) \neq f(v)$, если $\{u, v\} \in E$ и k – наименьшее возможное». В данном случае множество допустимых решений состоит из всех таких функций $f : V \to \{1, 2, \dots, |V|\}$, что $f(u) \neq f(v)$, если $\{u, v\} \in E$. Величина допустимого решения f равна $\max\{f(v) : v \in V\}$. В работе *Джонсона* [264] для большого числа полиномиальных приближённых алгоритмов A решения задачи о раскраске графа доказано существование такой константы c и такого бесконечного множества графов $G = (V, E)$, что

$$A(G) > c \cdot |V| \cdot OPT(G).$$

Более того, наилучшая из известных в настоящее время оценок погрешности для полиномиальных алгоритмов A решения задачи о раскраске графа имеет вид

$$A(G) = \frac{c \cdot |V|}{\log |V|} OPT(G)$$

(см. [264]). Иными словами, для задачи о раскраске графа не известен полиномиальный приближённый алгоритм с оценкой погрешности, хотя бы близкой к оценке $R_A < \infty$.

[6] MM– от английских слов *Minimum Matching* (минимальное паросочетание) . – *Прим. перев.*

Задача о раскраске графа, таким образом, представляется более трудной, чем задача об упаковке и задача о коммивояжере с неравенством треугольника. Для этой задачи за полиномиальное время мы не только не можем получить гарантированного оптимального решения (если $P \neq NP$), но даже разумных оценок приближения к оптимальному решению. Две другие задачи, которые представляются в такой же степени не поддающимися решению, – это задача отыскания максимального по мощности множества независимых вершин в графе и задача отыскания минимального маршрута для коммивояжера, если не требовать выполнения неравенства треугольника. (В следующем разделе мы ещё вернемся к этим задачам.)

Следует отметить, что возникающее очевидным образом подразделение задач на хорошо и плохо решаемые приближёнными методами, по-видимому, не имеет отношения к тому факту, что эти задачи тесно связаны посредством полиномиальных сводимостей. Можно было бы думать, что такая сводимость позволяет преобразовать хороший приближённый алгоритм решения одной задачи в хороший приближённый алгоритм решения другой. Однако, вообще говоря, это не так, что можно увидеть из приведённого ниже примера.

Напомним, что в разд. 3.1.3 было установлено, что задачи ВЕРШИННОЕ ПОКРЫТИЕ и НЕЗАВИСИМОЕ МНОЖЕСТВО очень тесно связаны. Для графа $G = (V, E)$ подмножество $V' \subseteq V$ является независимым множеством тогда и только тогда, когда дополнительное подмножество $V \setminus V'$ является вершинным покрытием в G. Более того, эта взаимосвязь переносится и на соответствующие оптимизационные задачи. Множество V' – максимальное независимое множество в G тогда и только тогда, когда $V \setminus V'$ – минимальное вершинное покрытие в G (в рассматриваемых случаях в качестве величины решения берется число вершин в независимом множестве и вершинном покрытии соответственно). Однако, хотя нам не известен приближённый полиномиальный алгоритм с погрешностью $R_A < \infty$ для решения задачи о максимальном независимом множестве (см. [263]), имеется очевидный алгоритм отыскания минимального вершинного покрытия с погрешностью $R_A \leqslant 2$ (см. [167]).

Последний алгоритм основан на идее «максимального паросочетания» в графе. *Паросочетанием* в графе $G = (V, E)$ называется такое множество $E' \subset E$, что никакие два ребра из E' не имеют общей вершины (взаимоотношение между этим понятием и понятием, использовавшимся в алгоритме Кристофидиса для задачи о коммивояжере, совершенно очевидно). Паросочетание E' называется максимальным, если каждое из рёбер, оставшихся в множестве $E \setminus E'$, имеет общую вершину с некоторым элементом множества E'. *Максимальное* паросочетание строится очень просто за полиномиальное время: к некоторому уже построенному паросочетанию последовательно добавляются рёбра, пока это возможно. Более того, заметим, что если E' – максимальное паросочетание для G, то множество всех концов рёбер из E' должно быть вершинным покрытием G (в противном случае E' не максимально). Такое вершинное покрытие имеет мощность, равную $2|E'|$. Важное обстоятельство состоит в том, что любое паросочетание для G должно иметь по крайней мере $|E'|$ вершин, поскольку оно должно включать по крайней мере один конец каждого ребра рассмотренного паросочетания E'. Таким образом, мощность вершинного покрытия, которое строится по паросочетанию E', не превосходит удвоенной мощности минимально возможного паросочетания.

Причина весьма различного поведения задач о максимальном независимом множестве и минимальном вершинном покрытии по отношению к решению полиномиальными приближёнными алгоритмами становится ясна, если мы попытаемся с помощью полиномиального сведения одной задачи к другой преобразовать приведённый выше алгоритм решения одной из них в алгоритм решения другой. Предположим, что имеется граф с 1000 вершинами и минимальное вершинное покрытие мощности 490. Указанный выше приближённый алгоритм гарантирует, что найденное с его помощью вершинное покрытие имеет мощность не более 980, поскольку отношение не превосходит 2, однако отношение мощностей соответствующих независимых множеств может достигать величины

$$\frac{1000-490}{1000-980} = \frac{510}{20} = 25{,}5.$$

Хотя сводимость сохраняет оптимальные решения, она не сохраняет *отношения* величины оптимального решения к величине приближённого. Нет оснований полагать, что близкое к оптимуму допустимое решение одной задачи отображается в близкое к оптимуму допустимое решение другой задачи. Таким образом, для того чтобы сводимости, используемые для доказательства *NP*-полноты можно было использовать для преобразования хороших приближённых алгоритмов решения одной задачи в аналогичные алгоритмы решения другой, необходимо, чтобы сводимости сохраняли величину решений более равномерно.

Рассмотренные нами типы поведения приближённых алгоритмов не исчерпывают всех возможностей. Мы видели, что некоторые задачи решаются приближённо гораздо хуже, чем задача об упаковке. С другой стороны, имеется ряд задач, которые представляются значительно более простыми.

Вернёмся, например, к определённой в разд. 3.2.1 задаче РЮКЗАК. Её оптимизационный вариант формулируется следующим образом: «Задано конечное множество U «предметов», для каждого из $u \in U^+$ «размер» $s(u) \in Z^+$ и «стоимость» $v(u) \in Z^+$, а также положительное число $B \geqslant \max\{s(u) : u \in U\}$ – «ёмкость рюкзака». Требуется найти такое подмножество $U' \subseteq U$, что $\sum_{u \in U'} s(u) \leqslant B$, и величина $\sum_{u \in U'} s(u)$ принимает наибольшее возможное значение». Иными словами, необходимо максимизировать ценность того, что помещается в «рюкзак», при условии, что сумма размеров предметов, помещённых в рюкзак, не превосходит его ёмкости.

Один простой приближённый алгоритм решения этой задачи работает следующим образом. Упорядчиваем множество $U = \{u_1, u_2, \ldots, u_n\}$ по убыванию «удельной ценности», т. е. так, что $v(u_1)/s(u_1) \geqslant v(u_2)/s(u_2) \geqslant \ldots \geqslant v(u_n)/s(u_n)$. Затем, начиная с пустого множества U', последовательно добавляем к нему предметы из списка, причём при добавлении предмета

u_i проверяем, не превосходит ли сумма размеров, уже содержащихся в U' предметов, величины $B - s(u_i)$. Затем сравниваем стоимость решения, полученного этим «жадным алгоритмом» с максимумом стоимости предметов из множества U и выбираем из этих двух величин наибольшую. Не представляет труда доказать, что построенный таким образом составной жадный алгоритм GA[7] имеет погрешность $R_{GA} = 2$.

Однако, включая этот алгоритм в более совершенную процедуру, можно достичь гораздо большего. *Сахни* [464] показал, что для всех $k \geqslant 1$ существует такой приближённый полиномиальный алгоритм A_k, что $R_{A_k} \leqslant 1 + (1/k)$. Основная идея построения такого алгоритма заключается в испытании в качестве исходного множества U' всех возможных множеств, состоящих не более чем из k предметов, и последующего добавления к нему (с использованием алгоритма GA) как можно большего числа предметов. Затем наилучшее из полученных множеств берется в качестве приближённого решения.

К сожалению, хотя для каждого фиксированного значения k время работы соответствующего алгоритма фактически полиномиально зависит от $n, \log V$ и $\log S$ (где $V = \max\{v(u) : u \in U\}$ и $S = \max\{s(u) : u \in U\}$), эти полиномы всегда содержат k в показателе. Чтобы гарантировать решения, очень близкие к оптимальному, пришлось бы обратиться к алгоритму, временна́я сложность которого была бы полиномом очень высокой степени и который мог бы оказаться весьма дорогим для применения на практике. Тем не менее указанные алгоритмы ведут себя, по крайней мере теоретически, лучше, чем алгоритмы для задачи об упаковке, поскольку, как мы видели, для последней задачи не известно приближённого полиномиального алгоритма с относительной погрешностью R_A, существенно лучшей, чем $\frac{11}{9}$. Для задачи о рюкзаке имеются полиномиальные алгоритмы, для которых R_A как угодно близка к 1.

Более того, недавно *Ибарра* и *Ким* [248][8] доказали, что для задачи о рюкзаке может быть построено приближённое решение с аналогичной оценкой погрешности без использования алгоритма, содержащего k в показателе. Этот результат базируется на том, что задача о рюкзаке может быть решена псевдополиномиальным оптимизационным алгоритмом. Ибарра и Ким показали, как за счет потери точности посредством округления и масштабирования подобный оптимизационный алгоритм может быть превращен в полиномиальный.

Для иллюстрации этого метода предположим, что A – псевдополиномиальный оптимизационный алгоритм решения задачи о рюкзаке, временна́я сложность которого имеет порядок, скажем,

$$O(n^2 V \log(nVS)).$$

Каждую индивидуальную задачу о рюкзаке можно модифицировать, заменяя стоимости $v(u)$ всех предметов $u \in U$ новой величиной $v'(u) =$

---

[7] GA – от английских слов *Greedy Algorithm* (жадный алгоритм). – *Прим. перев.*

[8] См. также работы Бабата [13*, 14*]. – *Прим. ред.*

$\lfloor v(u)/K \rfloor$, для некоторого фиксированного $K > 0$. Для применения алгоритма A к возникшей в результате индивидуальной задаче требуется время порядка

$$O(n^2(V/K)\log(nVS)).$$

Более того, поскольку оптимальное решение не может содержать больше чем n предметов, то имеется следующая взаимосвязь между $OPT(I)$, стоимостью оптимального решения исходной задачи, и $OPT(I')$, стоимостью оптимального решения модифицированной задачи:

$$OPT(I) - K \cdot OPT(I') \leqslant Kn.$$

Заметим, что $K \cdot OPT(I)'$ не превосходит стоимости оптимального решения задачи I', выраженной в исходных стоимостях предметов. Таким образом, если взять оптимальное решение U' задачи I' в качестве приближённого решения задачи I, то его стоимость будет отличаться от стоимости оптимального решения задачи I не более чем на Kn.

Желаемый результат получается теперь, если выбирать K зависящим от исходной индивидуальной задачи I. В частности, можно положить $K = V/(k+1)n$, где k – фиксированное положительное целое число. В результате получается приближённый алгоритм A_k, имеющий временну́ю сложность порядка $O(kn^3\log(nVS))$ и, следовательно, ограниченную полиномом от $n, \log V, \log S$ и k (время, затрачиваемое на построение по индивидуальной задаче I задачи I', не превосходит времени работы алгоритма A на входе I'). Более того, поскольку

$$A_k(I) \geqslant OPT(I) - Kn = OPT(I) - V/(k+1)$$

и так как $OPT(I) \geqslant V$, получаем

$$R_{A_k}(I) = \frac{OPT(I)}{A_k(I)} \leqslant \frac{A_k(I) + V/(k+1)}{A_k(I)} \leqslant 1 + \frac{V/(k+1)}{V - V/(k+1)} = 1 + (1/k).$$

что и утверждалось. (Алгоритм Ибарры и Кима несколько сложнее, однако приведённых рассуждений достаточно для иллюстрации основной идеи.)

Аппроксимационные возможности, выявленные на примере этих алгоритмов, не ограничиваются исключительно задачей о рюкзаке, а проявляются также в некоторых других ситуациях. Поэтому для обсуждения результатов подобного типа полезно иметь общую терминологию. Назовем *аппроксимационной схемой* для оптимизационной задачи Π алгоритм A, который, начиная работать на входе, состоящем из двух объектов – индивидуальной задачи $I \in D_\Pi$ и «желаемой точности» $\varepsilon > 0$, выдаёт такое допустимое решение $\sigma \in S_\Pi(I)$, что

$$R_{A_\varepsilon}(I) \leqslant 1 + \varepsilon.$$

Здесь термин «схема» используется по той причине, что в действительности A представляет собой набор приближённых алгоритмов решения задачи Π, по одному для каждого значения $\varepsilon > 0$.

Приближённая схема A называется *приближённой схемой с полиномиальным временем работы* (или просто приближённой полиномиальной схемой), если для каждого $\varepsilon > 0$ соответствующий алгоритм A_ε имеет полиномиальную временну́ю сложность. Приближённая схема A называется *вполне полиномиальной приближённой схемой*, если её временная сложность ограничена полиномом от $Length[I]$ и $1/\varepsilon$. Таким образом, последовательность алгоритмов, предложенная Сахни [464], представляет собой полиномиальную приближённую схему для задачи о рюкзаке, однако она не является вполне полиномиальной приближённой схемой. Последовательность алгоритмов Ибарры и Кима [248] представляет собой вполне полиномиальную приближённую схему для этой задачи.

Некоторые улучшения приближённой схемы Ибарры и Кима обсуждаются в работе [329]. Аналогичные приближённые схемы для задач, похожих на задачу о рюкзаке, а также для некоторых задач из теории расписаний можно найти в работах [465] и [221]. Основная идея всех этих приближённых схем аналогична описанной выше: в качестве основы схемы берется псевдополиномиальный алгоритм, затем применяются методы округления и масштабирования для того, чтобы за счет ограниченного уменьшения точности значительно уменьшить время работы алгоритма. Вполне полиномиальные приближённые схемы во многих отношениях – это лучшее, на что можно рассчитывать при решении NP-трудных задач оптимизации. Если $P \neq NP$, то для решения такой задачи, как нам известно, не удаётся построить приближённого полиномиального алгоритма A, для которого $R_A = 1$. Однако поскольку R_A^∞ – величина, определяемая асимптотически, то вполне возможно существование полиномиального алгоритма с $R_A^\infty = 1$. Например, Липтон и Тарьян [354] указали приближённый полиномиальный алгоритм A для отыскания максимальных независимых множеств в планарных графах (эта подзадача на плоских графах остается NP-трудной) с оценкой отклонения от оптимума

$$|A(I) - OPT(I)| \leqslant O\left(1/\sqrt{\log\,\log OPT(I)}\right) \cdot OPT(I).$$

Другая возможность достичь отношения $R_A^\infty = 1$ состоит в том, чтобы ограничить величину $|A(I) - OPT(I)|$ константой, не зависящей от I. Например, для задачи упаковки можно было бы попытаться получить такой приближённый полиномиальный алгоритм, что $A(I) \leqslant OPT(I) + 1$ для всех индивидуальных задач I. Хотя для задачи об упаковке такой алгоритм не известен, Горовитц и Сахни [222] показали, что его можно построить для NP-трудной задачи об упаковке максимально возможного числа предметов в два контейнера. Подобных результатов о разности $|A(I) - OPT(I)|$ для NP-трудных задач известно немного, однако несколько задач, для решения которых не известно полиномиального алгоритма, удаётся приблизить подобным образом; см. [504, 286, 285].

6.2. Применение теории NP-полноты к отысканию приближённых решений

До настоящего времени мы не привели никаких аргументов в пользу того, что выявленные различия в возможности приближённого решения NP-трудных задач имеют объективные причины, а не являются просто результатом нашего неумения находить хорошие приближённые алгоритмы. В настоящем разделе будет показано, что некоторые из этих различий в действительности– неотъемлемая черта самих задач и что теорией NP-полноты можно воспользоваться для того, чтобы очертить границы возможностей приближённого решения конкретной задачи. Для оптимизационной задачи П назовем *асимптотически наилучшей достижимой относительной погрешностью* величину

$$R_{MIN}(\Pi) = inf \left\{ r \geqslant 1 : \begin{array}{l} \text{для решения задачи } \Pi \text{ существует такой} \\ \text{приближённый полиномиальный} \\ \text{алгоритм } A, \text{ что } R_A^{\infty} = r. \end{array} \right\}.$$

На основе результатов, изложенных в предыдущем разделе, можно предположить, что существуют NP-трудные задачи, для которых $R_{MIN}(\Pi) = \infty$, существуют NP-трудные задачи, для которых $1 < R_{MIN}(\Pi) < \infty$, а также NP-трудные задачи, для которых $R_{MIN}(\Pi) = 1$. Если реализуется последняя возможность, то можно выделить следующие важные подслучаи: П может быть решена (1) полиномиальной приближённой схемой, (2) вполне полиномиальной приближённой схемой, (3) приближённым полиномиальным алгоритмом A, для которого $R_A^{\infty} = 1$, и (4) приближённым полиномиальным алгоритмом, A, для которого при некотором фиксированном K выполнено неравенство $|A(I) - OPT(I)| \leqslant K$. Конечно, возможны и другие типы оценок отклонения от оптимума, но перечисленные четыре типа оценок в настоящее время наиболее изучены, и в дальнейшем мы ограничимся только ими.

При поиске доказательства того, что для конкретной задачи некоторые из перечисленных возможностей не могут быть реализованы, мы, как обычно, наталкиваемся на то обстоятельство, что пока не известно, совпадают ли классы P и NP. Поэтому по-прежнему остается возможным, что все NP-эквивалентные задачи поддаются решению точно за полиномиальное время (все обсуждавшиеся до сих пор оптимизационные задачи являются как NP-лёгкими, так и NP-трудными). По этой причине мы опять ограничимся условными результатами, доказывая, что если $P \neq NP$, то задача П не может быть решена приближённым полиномиальным алгоритмом определённого типа.

Вначале рассмотрим одну из наиболее естественных оценок отклонения от оптимума, а именно результат вида «для всех индивидуальных задач I имеет место оценка $|A(I) - OPT(I)| \leqslant K$, где K – константа». Пример задачи, для которой такая оценка исключается, даёт нам задача о рюкзаке; её оптимизационный вариант был определён в предыдущем разделе.

Теорема 6.6. *Если $P \neq NP$, то не существует полиномиального приближённого алгоритма A для решения задачи о рюкзаке с оценкой*

$$|A(I) - OPT(I)| \leqslant K,$$

где K – некоторая фиксированная константа.

Доказательство 1. Предположим противное, т. е. что A – алгоритм, обладающий этим свойством (не ограничивая общности, можно предполагать, что K – положительное целое число). Покажем, что алгоритм A можно применить для точного решения задачи о рюкзаке за полиномиальное время, что противоречит предположению $P \neq NP$. Соответствующая процедура очень проста. Если задана любая индивидуальная задача о рюкзаке I, построим по I новую индивидуальную задачу I', заменяя стоимость $v(u)$ каждого предмета величиной $(K+1)v(u)$, и применим к задаче I алгоритм A. Очевидно, что все это можно осуществить за полиномиальное время. Более того, допустимые решения задачи I' оказываются точно такими же, как и у задачи I, а величина решения задачи I' ровно в $K+1$ раз больше величины соответствующего решения задачи I. Поскольку величина любого решения задачи I' – целое кратное величины $K+1$, то из соотношения $|A(I') - OPT(I)| \leqslant K$ немедленно следует, что $|A(I') - OPT(I')| = 0$ и, следовательно,

$$|A(I') - OPT(I)| = |A(I') - OPT(I')|/(K+1) = 0,$$

где A' обозначает описанный выше алгоритм. Таким образом, допустимое решение, конструируемое алгоритмом A', обязательно будет оптимальным, так что A' – полиномиальный оптимизационный алгоритм решения задачи о рюкзаке, что противоречит предположению $P \neq NP$. ■

Единственное свойство, использовавшееся в этом доказательстве, заключается в том, что величины всех решений можно умножить на произвольную константу u; при этом множество допустимых решений не изменится. Поскольку это свойство выполняется и для многих других задач, например задачи о коммивояжере (с неравенством треугольника), то легко видеть, что доказательства этого типа широко применимы. Менее очевидно, что эта же идея применима к решению задач совершенно иной природы. Рассмотрим, например, задачу о максимальном независимом множестве. В этой задаче нет чисел, которые можно было бы умножить, поэтому «умножим» весь граф целиком!

Теорема 6.7. *Если $P \neq NP$, то не существует приближённого полиномиального алгоритма для решения задачи о максимальном независимом множестве с оценкой*

$$|A(G) - OPT(G)| \leqslant K$$

для некоторой фиксированной константы K.

Доказательство 2. Предположим, что A – приближённый алгоритм, обладающий этим свойством, и опять предположим, что K – положительное целое число. Как и в предыдущем доказательстве, покажем, что A можно применить для построения полиномиального оптимизационного алгоритма, что противоречит предположению $P \neq NP$. Пусть граф G определяет индивидуальную задачу о вершинном покрытии. Процедура заключается в построении нового графа G', состоящего из $K+1$ экземпляров графа G. Нетрудно видеть, что $OPT(G') = (K+1)OPT(G)$. Более того, для графа G можно построить независимое множество, состоящее по крайней мере из $\lceil A(G')/(K+1) \rceil$ вершин. Для этого достаточно посмотреть, сколько вершин алгоритм A выбирает из каждого экземпляра графа G, и среди этих подмножеств взять подмножество максимальной мощности. Поскольку

$$|A(G') - OPT(G')| - |A(G') - (K+1)OPT(G)| \leqslant K,$$

то независимое множество для G будет содержать в точности $OPT(G)$ вершин. Следовательно, приведённая выше процедура – оптимизационный алгоритм для решения задачи о максимальном независимом множестве. Этот алгоритм полиномиальный по времени, так как K – константа, не зависящая от G. ■

В качестве упражнения читатель может попытаться воспользоваться этим подходом, чтобы доказать аналогичный результат для задачи отыскания минимального покрывающего множества – оптимизационного аналога задачи МИНИМАЛЬНОЕ ПОКРЫТИЕ, определённой в разд. 3.2.1.

Прежде чем от результатов о разности вернуться к результатам об относительной погрешности, отметим, что возможны результаты промежуточного типа. Несколько видоизменив технику, применённую в доказательстве теоремы 6.7, для задачи о максимальном независимом множестве можно показать, что каковы бы ни были K и $\varepsilon > 0$, не существует полиномиального приближённого алгоритма с оценкой

$$|A(G) - OPT(G)| \leqslant K \cdot OPT(G)^{1-\varepsilon}$$

(если $P \neq NP$). Аналогичные результаты для этой и других задач получены в работах [402] и [309]. Заметим, однако, что, как видно из результатов *Липтона* и *Тарьяна* [354], упомянутых в конце предыдущего раздела, результаты этого типа не исключают возможности существования полиномиального приближённого алгоритма с погрешностью $R_A^\infty = 1$. Единственный известный способ доказательства того, что не существует полиномиального приближённого алгоритма A с асимптотической погрешностью $R_A^\infty = 1$ (если $P \neq NP$), состоит в доказательстве того, что из $P \neq NP$ следует $R_{MIN}(\Pi) > 1$, т.е. принципиально более сильное утверждение. Поскольку такие доказательства удобнее считать доказательством нижней оценки для величины $R_{MIN}(\Pi)$ (в случае когда $1 < R_{MIN}(\Pi) < \infty$), то мы временно отложим их обсуждение; вместо этого изучим вопрос о возможности решения задачи Π вполне полиномиальной приближённой схемой.

Напомним, что в предыдущей главе было сказано, что все известные вполне полиномиальные приближённые схемы для решения *NP*-трудных задач имеют источником псевдополиномиальные оптимизационные алгоритмы. В связи с этим не удивительно, что сильная *NP*-полнота оказывается связанной с существованием таких схем. В действительности взаимосвязь псевдополиномиальных алгоритмов с вполне полиномиальными приближёнными схемами носит двусторонний характер. Пусть П – оптимизационная задача, а $Length[I]$ и $Max[I]$ – две, связанные с ней меры индивидуальных задач (см. разд. 4.2), и предположим, что величины всех решений – положительные целые числа. Следующий результат доказан в работах *Гэри* и *Джонсона* [151, 152].

Теорема 6.8. *Если существует такой полином q, зависящий от двух переменных, что для любой индивидуальной задачи $I \in D_\Pi$ выполнено соотношение*

$$OPT(I) < q(Length[I], Max[I]),$$

то из существования вполне полиномиальной приближённой схемы для решения задачи П следует существование псевдополиномиального оптимизационного алгоритма решения задачи П.

Доказательство 3. Предположим, что A – подобная схема для решения задачи П. Соответствующий оптимизационный алгоритм A' работает следующим образом. Пусть I – некоторая индивидуальная задача, положим

$$\varepsilon = q(Length[I], Max[I])^{-1}$$

и применим к задаче I схему A при заданной точности ε. За время, ограниченное полиномом от $Length[I]$, и

$$q(Length[I], Max[I])$$

и следовательно, за псевдополиномиальное время, алгоритм A находит допустимое решение задачи I с погрешностью

$$R_{A_\varepsilon}(I) \leqslant 1 + \varepsilon$$

Предположим, что П – задача максимизации (аналогичные рассуждения применимы и в том случае, когда П – задача минимизации). Тогда, в силу определения ε и предположения о q, имеем

$$OPT(I) \leqslant (1+\varepsilon) A_\varepsilon(I)$$

или

$$OPT(I) - A_\varepsilon(I) \leqslant \varepsilon \cdot OPT(I) < 1.$$

Однако, поскольку значения всех решений – целые числа, отсюда вытекает, что $A_\varepsilon(I) = OPT(I)$ и, следовательно, A' за псевдополиномиальное время находит оптимальное решение, что и требовалось доказать. ∎

В качестве следствия получаем метод доказательства невозможности существования вполне полиномиальной оптимизационной схемы для решения задачи П.

Следствие. *Пусть П – оптимизационная задача с целочисленными величинами решений, удовлетворяющая предположениям теоремы 6.8. Если П является NP-трудной в сильном смысле задачей, то (при $P \neq NP$) П не может быть решена вполне полиномиальной приближённой схемой.*

Два основных условия следствия заключаются в том, что величины решений задачи П – не слишком большие целые числа и что П *NP*-трудна в сильном смысле. Эти условия выполняются для многих представляющих интерес задач, включая, например, задачи об упаковке, раскраске графа, о максимальном независимом множестве и минимальном вершинном покрытии. Таким образом, мы владеем общим весьма сильным методом доказательства того, что не существует вполне полиномиальных приближённых схем.

Несколько труднее доказывать невозможность существования приближённых полиномиальных, но не вполне полиномиальных схем. Простейший способ заключается, по-видимому, просто в доказательстве того, что если $P \neq NP$, то для некоторого $r > 1$ не существует полиномиального приближённого алгоритма A решения задачи П с погрешностью $R_A < r$. Задача о раскраске графа представляет собой пример задачи, для которой существование такого r очевидно. В самом деле, полиномиальный алгоритм A для решения задачи о раскраске графа с погрешностью $R_A < \frac{4}{3}$ должен был бы раскрашивать 3-раскрашиваемый граф не более чем в 3 цвета и, следовательно, решал бы *NP*-полную задачу РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА за полиномиальное время. Таким образом, если $P \neq NP$, то приближённого алгоритма с такой погрешностью не существует и, следовательно, не существует полиномиальной приближённой схемы для задачи о раскраске графа. Изложенную выше идею можно сформулировать в виде несложной теоремы.

Теорема 6.9. *Пусть П – задача минимизации, величины всех решений которой – неотрицательные целые числа. Предположим, что для некоторого фиксированного $K \in Z^+$ задача распознавания: «Задана индивидуальная задача $I \in D\Pi$, верно ли, что $OPT(I) \leqslant K$ является NP-трудной». Тогда если $P \neq NP$, то не существует полиномиального приближённого алгоритма A решения задачи П с погрешностью $R_A < 1 + \frac{1}{K}$ и П не может быть решена полиномиальной приближённой схемой.*

Конечно, для задач максимизации имеет место аналогичный результат. Теорема 6.9 применима также к задаче об упаковке (при $K = 2$), задаче минимизации срока выполнения заданий, возникающей из задачи РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ из разд. 4.1 (при $K = 3$); см. [336], и множеству других задач. Однако имеется целый ряд задач, не содержащих *NP*-трудных подзадач нужного вида, и для них вопрос о существовании полиномиальной приближённой схемы остается открытым.

Вернёмся теперь к вопросу об отыскании нижней оценки величины $R_{MIN}(\Pi)$. Заметим, что теорема 6.9 не даёт возможности получить такую оценку, поскольку в её формулировке участвует только R_A, а наилучшая достижимая нижняя оценка величины R_A не обязательно справедлива для асимптотической погрешности R_A^∞. Это можно пояснить на примере задачи об упаковке. Теорема 6.9 утверждает, что если $P \neq NP$, то для решения задачи об упаковке не может существовать полиномиального приближённого алгоритма A с погрешностью $R_A < \frac{3}{2}$. С другой стороны, как мы уже видели, $R_{FFD}^\infty = \frac{11}{9} < \frac{3}{2}$. На самом деле, даже предполагая, что $P \neq NP$, можно лишь утверждать, что в задаче об упаковке имеет место неравенство $1 \leqslant R_{MIN}(\Pi) \leqslant \frac{11}{9}$.

Тем не менее для некоторых задач оценки, получаемые с помощью теоремы 6.9, могут быть так усилены, что они окажутся справедливыми и для асимптотической погрешности. Поясним это на задаче о раскраске графов.

`Теорема 6.10.` *Если $P \neq NP$, то для решения задачи о раскраске графов не существует полиномиального приближённого алгоритма A с асимптотической погрешностью $R_A^\infty < \frac{4}{3}$.*

Доказательство 4. Покажем, что с помощью любого такого алгоритма A можно было бы получить полиномиальный алгоритм для задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА, что противоречит предположению $P \neq NP$. В конструкции используется понятие «композиция» двух графов. Пусть $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ – два графа. Композицией $G = G_1[G_2]$ этих графов называется граф с множеством вершин $V = V_1 \times V_2$ и множеством рёбер

$$E = \left\{ \{(u_1, u_2), (v_1, v_2)\} : \quad \begin{array}{l} \text{либо } \{u_1, u_2\} \in E_1, \\ \text{либо } u_1 = v_1 \text{ и } \{u_2, v_2\} \in E_2. \end{array} \right.$$

Пример композиции двух графов представлен на рис. 6.6. Композицию $G_1[G_2]$ удобно представлять себе как результат замены каждой вершины графа G_1 экземпляром графа G_2 с последующей заменой каждого ребра графа G_1 полным двудольным подграфом, соединяющим каждую вершину экземпляра, соответствующего одной вершине ребра, с каждой вершиной экземпляра, соответствующего второй вершине этого ребра.

Рис. 6.6.

Рассмотрим следующий алгоритм для задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА. Поскольку $R_A^\infty < \frac{4}{3}$, то существует такое неотрицательное целое число $K \in Z^+$, что для всех графов G с $OPT(G) \geqslant K$ выполняется соотношение $A(G) \leqslant \left(\frac{4}{3}\right) OPT(G)$. Пусть $G = (V, E)$ – произвольная индивидуальная задача РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА. Пусть G' – полный граф на K вершинах. Положим $G^* = G'[G]$. Заметим, что G^* – это в точности K изоморфных экземпляров графа G, причём любые две вершины из различных экземпляров соединены ребром, поэтому

$$OPT(G^*) = K \cdot OPT(G) \geqslant K.$$

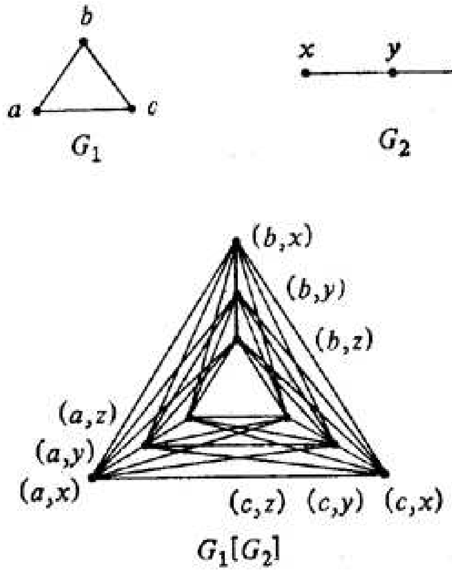


Рис. 6..6: *Композиция* $G_1[G_2]$ *графов* G_1 *и* G_2.

Более того, размер графа G^* и время, необходимое для его построения (поскольку K не зависит от G), ограничены полиномами от размера графа G. Поэтому время работы алгоритма A на графе G^* также будет ограничено полиномом от размера графа G. Однако если G 3-раскрашиваем, то справедливы неравенства

$$A(G^*) \leqslant \frac{4}{3} \cdot OPT(G^*) \leqslant \frac{4}{3} \cdot 3K = 4K.$$

С другой стороны, если G нераскрашивасм в 3 цвета, то

$$A(G^*) \geqslant OPT(G^*) \geqslant 4K.$$

Следовательно, G можно раскрасить в три цвета тогда и только тогда, когда $A(G^*) < 4K$. Поэтому процедура, состоящая в построении графа G^* и применении к нему алгоритма A, даёт полиномиальный алгоритм для решения задачи РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА. Это приводит к противоречию и завершает доказательство теоремы. ■

Аналогичное доказательство оценки $R_{MIN}(\Pi) \geqslant \frac{4}{3}$ можно получить для упоминавшейся выше задачи о построении расписания с отношением предшествования. Действительно, в то время как для задачи о раскраске графов не известен полиномиальный приближённый алгоритм с оценкой $R_A^\infty < \infty$, для задачи о построении расписания дело обстоит гораздо лучше. Для её решения полиномиальный приближённый алгоритм A с оценкой погрешности $R_A^\infty = 2$ был предложен *Грэхемом* [185], и поэтому для этой задачи величина $R_{MIN}(\Pi)$ удовлетворяет неравенствам $\frac{4}{3} \leqslant R_{MIN}(\Pi) \leqslant 2$ (в предположении $P \neq NP$).

Для задачи о раскраске графов нижняя оценка для величины $R_{MIN}(\Pi)$, равная $\frac{4}{3}$, может быть существенно улучшена методом «увеличения» погрешности. Имеет место следующий результат, доказанный в [146].

Теорема 6.11. *Если $P \neq NP$, то для задачи о раскраске графов не существует полиномиального приближённого алгоритма A с погрешностью $R_A^\infty < 2$.*

Доказательство 5. Предположим, что такой полиномиальный приближённый алгоритм A существует. Как и при доказательстве теоремы 6.10, покажем, что с помощью алгоритма A можно решить NP-полную задачу РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА за полиномиальное время. Для этого опять воспользуемся операцией композиции графов. Однако нам понадобится дополнительное понятие «мультираскраски» графа: (k,m)-*мультираскраской графа* $G = (V, E)$ называется функция f, сопоставляющая каждой вершине $v \in V$ множество $f(v) \subseteq \{1, 2, ..., K\}$, такая, что $|f(v)| = m$ для всех вершин v, и если $\{u, v\} \in E$, то $f(u)$ и $f(v)$ не содержат общих элементов. На рис. 6.7 представлен пример (5,2)-мультираскраски. m-*хроматическим числом графа* G, $\chi_m(G)$, называется наименьшее целое число $k \geqslant 1$, такое, что существует (k,r)-мультираскраска графа G. Заметим, что $(k,1)$-мультираскраска соответствует обычной раскраске графа, а величина $\chi(G)$ совпадает с $OPT(G)$.

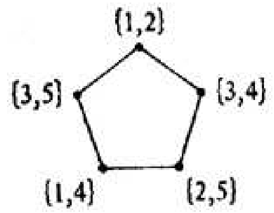


Рис. 6..7: *(5.2)-мультираскрашивание пятиугольника C_5; каждая вершина v помечена соответствующим множеством «цветов» $f(v)$. Можно показать, что $\chi(C_5) = 5$.*

При доказательстве данной теоремы графы $H_n, n \geqslant 6$, определяемые ниже, играют ту же роль, что и полные графы при доказательстве теоремы 6.10. Граф $H_n = (V_n, E_n)$ имеет $n(n-1)(n-2)/6$ вершин, соответствующих различным 3-х элементным подмножествам множества $\{1, 2, \ldots, n\}$. Причем две вершины соединены ребром из E_n тогда и только тогда, когда соответствующие подмножества не пересекаются. Важнейшее свойство этих графов, как доказано в работе [146], заключается в том, что при $n \geqslant 6$ выполнены соотношения

$$\chi_3(H_n) = n \text{ и } \chi_4(H_n) = 2n - 4.$$

Это свойство позволяет использовать графы в качестве «увеличителя погрешности».

Поскольку $R_A^\infty < 2$, то существуют такие $\varepsilon > 0$ и $K \in Z^+$, что для всех графов G с $OPT(G) \geqslant K$ выполнено свойство

$$A(G) \leqslant (2-\varepsilon)OPT(G).$$

Пусть $N \geqslant max\{6, K\}$ - такое целое число, что $2N - 4 > (2-\varepsilon)N$. Рассмотрим следующий алгоритм проверки раскрашиваемости графа в 3 цвета.

Пусть $G = (V, E)$ – произвольный граф и нужно выяснить, можно ли его раскрасить в 3 цвета. Вначале построим граф-композицию $G^* = H_N[G]$ (это можно сделать за полиномиальное время, поскольку N не зависит от G). Затем применим к графу G^* алгоритм A, что опять-таки потребует времени, ограниченного полиномом от размера графа G. Мы утверждаем, что граф G раскрашиваем в 3 цвета тогда и только тогда, когда $A(G^*) < 2N - 4$.

Предположим, что граф G раскрашиваем в 3 цвета. Тогда граф G^* можно раскрасить, используя только N цветов. Это делается следующим образом. Фиксируем некоторое $(N, 3)$-мультираскрашиваиие графа H_N (оно существует потому, что $\chi_3(H_N) = N$) и раскрасим каждый экземпляр графа G, содержащийся в G^*, в три цвета, соответствующих вершине H_N при выбранном мультираскрашивании. Никакие две вершины различных экземпляров графа G, соединенные в G^* ребром, не будут окрашены в один цвет, поскольку соответствующие вершины графа H_N соединены ребром, и, следовательно, соответствующие им при мультираскрашивании множества не пересекаются. Таким образом, мы получаем раскраску графа G^* в N цветов. Из определений N и ε следует, что если G раскрашивается в 3 цвета, то

$$A(G^*) \leqslant (2-\varepsilon)OPT(G^*) \leqslant (2-\varepsilon)N < 2N - 4.$$

С другой стороны, предположим, что G нельзя раскрасить в 3 цвета. Тогда при любой раскраске графа G^* каждый экземпляр графа G должен содержать вершины не менее четырёх разных цветов. По определению графа G^*, любые два экземпляра графа G, соответствующие соседним вершинам графа H_N, не должны содержать вершин одинакового цвета. Поэтому любое раскрашивание графа G^* в k цветов индуцирует $(k, 4)$-мультираскрашивание графа H_N, получаемое посредством сопоставления каждой вершине $v \in V_N$ любой четвёрки цветов, используемых в соответствующем экземпляре графа G. Однако поскольку $\chi_4(H_N) = 2N - 4$, то отсюда следует, что $k \geqslant 2N - 4$, т. е. G^* не может быть раскрашен менее чем в $2N - 4$ цвета. Отсюда вытекает, что если G нельзя раскрасить в 3 цвета, то

$$A(G^*) \geqslant OPT(G^*) \geqslant 2N - 4.$$

Таким образом, граф G раскрашиваем в 3 цвета тогда и только тогда, когда $A(G^*) < 2N - 4$, и построенный выше алгоритм является полиномиальным. Это противоречит предположению, что $P \neq NP$, и теорема доказана. ∎

В качестве следствия из теоремы 6.11 получаем, что если $P \neq NP$, то для задачи о раскраске графов $2 \leqslant R_{MIN}(\Pi) \leqslant \infty$. Более сильных нижних оценок в настоящее время не известно, хотя и существует предположение, что $R_{MIN}(\Pi) = \infty$.

Отметим, что все приведённые ранее нижние оценки величины $R_{MIN}(\Pi)$ решающим образом зависели от того факта, что у задачи Π имелась *NP*-трудная подзадача вида: «Задано $I \in D_\Pi$, верно ли, что $OPT(I) \leqslant K$?» (или «Задано $I \in D_\Pi$, верно ли, что $OPT(I) \geqslant K$?») – при некотором фиксированном числе K. Если Π не обладает этим свойством, то применявшаяся выше схема доказательства не сработает. Тем не менее даже в этом случае некоторые результаты удаётся сохранить. Вернёмся ещё раз к задаче о максимальном независимом множестве. Для этой задачи не известен полиномиальный приближённый алгоритм A с асимптотической погрешностью $R_A^\infty < \infty$, но при любом фиксированном значении K подзадача: «Задан граф G, верно ли, что G содержит независимое множество мощности не менее K?» – разрешима за полиномиальное время. Докажем следующий результат.

`Теорема 6.12.` *Имеет место одна из двух следующих возможностей. Либо задачу о максимальном независимом множестве можно решить полиномиальной приближённой схемой, либо для этой задачи не существует полиномиального приближённого алгоритма A с $R_A^\infty < \infty$.*

Доказательство 6. Предположим, что A – такой полиномиальный приближённый алгоритм решения этой задачи, что $R_A^\infty < \infty$. Поскольку можно считать, что A всегда находит независимое множество мощности не менее 1, то $R_A^\infty < \infty$. Пусть $R_A = r$. Построим приближённую схему следующим образом. Для любого $\varepsilon > 0$ обозначим через N_ε наименьшее целое число N, для которого выполнено неравенство $r^{1/N} < 1 + \varepsilon$. (Поскольку в дальнейших рассуждениях r фиксировано, то для краткости индекс у N_ε будем опускать.) Для произвольного графа $G = (V, E)$ укажем способ отыскания независимого множества мощности $M \geqslant OPT(G)/(1+\varepsilon)$ за время, ограниченное полиномом от размера графа G.

Вначале определим по индукции граф $G_N = (V_N, E_N)$ следующим образом: $G_1 = G$, а для $1 < i \leqslant N$ положим $G_i = G_{i-1}[G]$. Поскольку N зависит только от ε, то при фиксированном ε граф G_N можно построить за полиномиальное время. Более того, нетрудно видеть, что $OPT(G_i) = (OPT(G))^i, 1 < i \leqslant N$, и по любому независимому подмножеству S_N в G_N за полиномиальное время можно построить такое независимое множество S в G, что

$$|S| \geqslant \lceil |S_N|^{1/N} \rceil.$$

Рассмотрим результат применения алгоритма A к графу G_N. Поскольку $R_A = r$, то получаемое для графа G_N независимое множество S_N имеет мощность

$$|S_N| \geqslant OPT(G_N)/r = (OPT(G))^N/r.$$

Используя это множество, для графа G можно построить независимое множество S, такое, что

$$|S| > \lceil ((OPT(G))^N/r)^{1/N} \rceil \geqslant OPT(G)/r^{1/N} \geqslant OPT(G)/(1+\varepsilon)$$

(где последнее неравенство получено согласно определению числа N). При фиксированном ε описанная выше процедура построения множества G может быть выполнена за полиномиальное время. Таким образом, на основе алгоритма A мы построили полиномиальную приближённую схему решения задачи о максимальном независимом множестве. ∎

В качестве следствия теоремы 6.12 получается следующий результат: для задачи о максимальном независимом множестве имеются только две возможности: либо $R_{MIN}(\Pi) = 1$, либо $R_{MIN}(\Pi) = \infty$; т.е. промежуточное значение величины $R_{MIN}(\Pi)$ невозможно. Таким образом, если бы удалось получить хоть *какой-нибудь* полиномиальный алгоритм с асимптотической погрешностью, меньшей ∞, то можно было бы построить полиномиальный алгоритм с *любой*, как угодно близкой к 1 асимптотической погрешностью. С другой стороны, если бы для некоторого $\varepsilon > 0$ нам удалось доказать, что не существует полиномиального приближённого алгоритма с погрешностью $R_A < 1+\varepsilon$, то отсюда вытекало бы, что у всех полиномиальных приближённых алгоритмов $R_A = R_A^\infty = \infty$. Хотя мы и предполагаем, что $R_{MIN}(\Pi) = \infty$, нам не известны никакие перспективные подходы к доказательству этого утверждения.

Таким образом, мы приходим к последнему типу отрицательных результатов. Для некоторых задач Π все-таки удаётся доказать, что $R_{MIN}(\Pi) = \infty$ (в предположении $P \neq NP$). Эти доказательства очень похожи на уже рассмотренные доказательства нижних оценок, отличие лишь в том, что теперь удаётся «увеличить погрешность» в гораздо большей степени. В качестве примера рассмотрим опять задачу о коммивояжере, не предполагая на сей раз, что для расстояний между городами выполнено неравенство треугольника. Следующая теорема доказана в работе [469].

Теорема 6.13. *Если $P \neq NP$, то для задачи о коммивояжере не существует приближённого полиномиального алгоритма с асимптотической погрешностью $R_A^\infty < \infty$.*

Доказательство 7. Предположим, что приближённый алгоритм, удовлетворяющий перечисленным в условии теоремы свойствам, существует. Тогда для некоторого положительного целого числа K существовал бы полиномиальный алгоритм A с погрешностью $R_A \leqslant K$. Покажем, используя вариант сведения ГЦ $\propto$ КМ из гл. 2, как с помощью алгоритма A может быть решена NP-трудная задача ГАМИЛЬТОНОВ ЦИКЛ. Пусть $G = (V, E)$ – произвольный граф. Построим соответствующую индивидуальную задачу о коммивояжере I. Пусть V – множество городов, и пусть расстояние между городами u и v равно

$$d(u,v) = \begin{cases} 1, & \text{если } \{u,v\} \in E; \\ K \cdot |V| & \text{в противном случае.} \end{cases}$$

Ясно, что построить индивидуальную задачу I и применить к ней алгоритм A можно за полиномиальное время (поскольку K не зависит от G). Однако если в G есть гамильтонов цикл, то $OPT(I) = |V|$, а если в G нет гамильтонова цикла, то $OPT(I) > K \cdot |V|$. Таким образом, воспользовавшись оценкой погрешности алгоритма A, получим, что $A(I) \leqslant K \cdot |V|$ тогда и только тогда, когда в G имеется гамильтонов цикл. Следовательно, из существования полиномиального приближённого алгоритма A с погрешностью $R_A < \infty$ следует, что КОММИВОЯЖЕР лежит в классе P, что противоречит предположению $P \neq NP$. ■

Эта теорема указывает на то, что неравенство треугольника существенно для справедливости полученных в предыдущем разделе положительных результатов, относящихся к задаче о коммивояжере. Если неравенство треугольника выполнено, то имеем оценку $R_{MIN}(\Pi) \leqslant \frac{3}{2}$, а без неравенства треугольника $R_{MIN}(\Pi) = \infty$ (при условии $P \neq NP$). Некоторые результаты относительно других задач, аналогичные теореме 6.13, можно найти в работах [469] и [459].

Таким образом, относительно любого из возможных типов поведения приближённых алгоритмов, перечисленных в разд. 6.1, можно доказать негативные результаты, состоящие в том, что если $P \neq NP$, то для конкретной задачи Π не существует приближённого алгоритма с тем или иным поведением. Это наводит на мысль о том, что в предположении различия классов P и NP оптимизационные задачи можно классифицировать согласно значению величины $R_{MIN}(\Pi)$ (а если $R_{MIN}(\Pi) = 1$, то возможна ещё более тонкая классификация). На рис. 6.8 представлены известные факты о такой классификации для некоторых задач, обсуждавшихся выше.

В заключение напомним читателю, что, несмотря на то что мы выбрали для изучения особенно характерные виды погрешностей, возможны также и другие виды погрешностей, рассмотренные в некоторых работах. Кроме того, эта область исследований продолжает развиваться, и в ней постоянно получают новые, как положительные, так и отрицательные результаты. Дальнейшую информацию можно найти в работе Гэри и Джонсона [147], где приведена аннотированная библиография работ в этой области, опубликованных к середине 1976 г.

	Значение R_{MIN} (обозн. см. под таблицей)				
Наименование задачи	1	2	3	4	5
Рюкзак	+		+	+	+
Упаковка в контейнеры		+	+		+
Минимальное вершинное покрытие	+	+			+
Максимальное независимое множество	+	+		+	
Коммивояжер с неравенством треугольника	+	+			+
Раскраска графов	+	+	+		
Коммивояжер	+	+	+	+	
Расписание с отношением предшествования	+	+	+		+

Обозначения столбцов:

1: $R_{MIN}(\Pi) = 1$. Оценка разности.

2: $R_{MIN}(\Pi) = 1$. Вполне полиномиальная схема.

3: $R_{MIN}(\Pi) = 1$. Полиномиальная схема.

4: $1 < R_{MIN}(\Pi) < \infty$

5: $R_{MIN}(\Pi) = \infty$

Рис. 6.8. Классификация нескольких оптимизационных задач относительно значений величины $R_{MIN}(\Pi)$. При $R_{MIN}(\Pi) = 1$ имеются различные возможности. Светлые (пустые) прямоугольники указывают на открытые в настоящее время вопросы. Затененные (со знаком «+») прямоугольники указывают на то, что соответствующая возможность не имеет места либо потому, что это можно доказать при условии $P \neq NP$, либо потому, что уже известен лучший алгоритм.

6.3. Оценки погрешности и поведение алгоритмов «на практике»

Нетрудно понять причину, в силу которой очень желательно иметь такие оценки погрешностей, какие обсуждались выше для эвристических алгоритмов. Даже если конкретная оценка погрешности и не столь сильна, как хотелось бы (решение, отличающееся от оптимального на 50%, часто не может считаться удовлетворительным), тем не менее весьма разумно начать решение задачи с простого алгоритма, имеющего такую оценку погрешности, а затем улучшить её, пользуясь более совершенными эвристиками и методами локальной оптимизации. Кроме того, некоторые доказанные оценки погрешности достаточно хороши и могут быть достигнуты без чрезмерных затрат машинного времени. В частности, весьма удовлетворительными могут оказаться вполне полиномиальные приближённые схемы при значениях ε, меняющихся в разумных пределах. Более того, поскольку погрешности оцениваются в расчёте на худший случай, то на практике приближённые алгоритмы ведут себя гораздо лучше, чем можно было бы предположить на основе оценки их погрешности.

Подчеркнем ещё раз, что это имеет место «на практике». Наибольший интерес для нас представляет то, в какой степени наш алгоритм «на практике» будет приближать оптимальное решение. В качестве альтернативы к анализу погрешности «в худшем случае» можно было бы попытаться оценить погрешность «в среднем». Такой анализ в действительности имеет богатую историю и до недавнего времени осуществлялся преимущественно при помощи эмпирических методов.

Часто изучение поведения алгоритмов «в среднем» сводилось просто к формированию множества предположительно «типичных» индивидуальных задач, прогонке соперничающих алгоритмов на этом множестве и сравнению полученных результатов. Конечно, полезность экспериментов такого рода зависела от того, насколько «типична» рассматриваемая выборка индивидуальных задач, кроме того, не всегда достаточно просто образовать подходящий набор индивидуальных задач.

Если можно считать, что индивидуальные задачи, предположительно встречающиеся на практике, подчиняются некоторому конкретному распределению вероятностей, то можно воспользоваться более совершенными методами. В этом случае может оказаться доступным метод построения «случайных» индивидуальных задач, подчиняющихся заданному распределению вероятностей. На построенной таким образом выборке затем можно исследовать поведение алгоритмов.

Некоторые эксперименты, выполненные в этом направлении, подтверждают сделанное выше замечание, что поведение алгоритма в среднем гораздо лучше, чем поведение в худшем случае. В работе Джонсона [262] алгоритмы «в первый подходящий» и в «первый подходящий в порядке убывания» для решения задачи об упаковке анализировались методом Монте-Карло при самых разных предположениях о распределении размеров предметов, причём в качестве оценки оптимума $OPT(I)$ использовалась сумма размеров предметов. В результате исследований было выяснено, что погрешность $R_{FF}(I)$ в среднем равна 1.07, в то время как в худшем случае она равна 1.70, а погрешность $R_{FFD}(I)$ в среднем равна 1.02, в то время как в худшем случае она равна 1.22. (Интересно отметить, что взаимное ранжирование этих алгоритмов «в среднем случае» почти такое же, как «в худшем случае».) В работе Сахни [466] было установлено, что одна конкретная вполне полиномиальная приближённая схема для решения задачи об упорядочении с директивными сроками работает с погрешностью всего 1,005 даже в том случае, когда ε выбрано так, чтобы гарантировать погрешность 1.10, а алгоритм в действительности почти всегда находит оптимальное решение. Однако подобный сравнительный анализ проведён лишь для очень небольшого числа алгоритмов, и в настоящее время, вероятно, неразумно делать решительные и поспешные выводы.

В некоторых недавних работах предложены более совершенные подходы, позволяющие доказывать *теоремы* об ожидаемом поведении приближённых алгоритмов при различных конкретных распределениях вероятностей. В работе [191], например, в предположении «равновероятности» в некотором смысле всех графов на n вершинах доказано, что некоторый простой алгоритм раскраски A имеет среднюю погрешность $R_A(G)$, равную 2, хотя в худшем случае погрешность $R_A(G)$ может быть произвольно велика. Аналогичные результаты были получены для задачи об упаковке, см. [484,541], для задачи о максимальном независимом множестве, задачи о гамильтоновом цикле и геометрических версий задач о коммивояжере и дереве Штейнера, см. [282, 283, 284, 18]. Несмотря на то что даже для простейших алгоритмов и распределений вероятностей такой анализ может оказаться весьма затруднительным, это направление представляется многообещающим и в ближайшем будущем можно ожидать появления новых результатов этого типа.

Конечно, анализ поведения алгоритмов в среднем, рассматриваемый как основа для оценки их поведения «на практике», имеет свои недостатки. Для того чтобы вычислить среднее, необходимо в первую очередь предположить, что индивидуальные задачи подчиняются некоторому распределению вероятностей, и довольно часто оказывается не ясно, какое распределение следует выбрать. (Интересный способ избежать этого путём применения алгоритмов, обладающих внутренней рандомизацией, обсуждается в работе [441].) Распределения вероятностей, для которых, пользуясь методами, имеющимися на сегодняшний день, можно анализировать алгоритмы, могут довольно сильно отличаться от истинных распределений, встречающихся на практике, где преобладают задачи со сложной структурой (что ведёт к отклонениям, с большим трудом поддающимся формализации). На практике также распределения вероятностей могут непредсказуемым образом меняться со временем. Кроме того, результаты о погрешности в среднем ничего не говорят о работе алгоритмов на конкретных задачах, в то время как оценка погрешности в худшем случае даёт по крайней мере оценку поведения алгоритмов. Поэтому, если нужно получить как можно больше информации о поведении приближённых алгоритмов на практике, вероятно, лучше всего подвергнуть их анализу всеми возможными способами как с точки зрения поведения в худшем случае, так и с точки зрения поведения в среднем.

Глава 7.

За пределами класса *NP*-полных задач

В настоящей главе рассматривается несколько теоретических вопросов, связанных с теорией *NP*-полных задач (и возникших благодаря ей). У читателя не предполагается никаких дополнительных знаний, даже начинающий вполне может освоить основные положения этой главы и их связь с остальной теорией. Читатели, заинтересовавшиеся результатами, приведёнными в настоящей главе, смогут найти более подробные сведения, включая доказательства этих результатов в статьях и книгах, указанных в списке литературы. Следуя принятому ранее стилю изложения, мы будем определять новые понятия в терминах языков и машин Тьюринга и обсуждать их неформально в терминах массовых задач и алгоритмов.

В разд. 7.1 с точки зрения, изложенной в гл. 2, рассматриваются известные результаты и гипотезы о структуре класса *NP* и обсуждается существование в классе *NP* труднорешаемых задач, не являющихся *NP*-полными. В разд. 7.2–7.4 обсуждаются вопросы «полноты» для классов задач, которые представляются более трудными, чем *NP*-полные задачи, однако труднорешаемость этих задач также ещё *предстоит доказать*. В разд. 7.2, в частности, рассматривается «полиномиальная иерархия», а в разд. 7.3 вводится понятие КР-полноты для задач перечисления. В разд. 7.4 на основе понятия полноты с полиномиальной памятью начинается обсуждение взаимосвязи ограничений по времени и по памяти. В разд. 7.5 это обсуждение продолжается, при этом вводится понятие «сводимость с логарифмической памятью» и показывается, как оно может быть использовано для исследования ограничений по памяти в задачах класса P. Наконец, в разд. 7.6 мы вновь возвращаемся к фундаментальной проблеме «верно ли, что $P = NP$?». Хотя мы считаем, что эти классы *различны*, но, по-видимому, имеются серьёзные препятствия для доказательства этого утверждения современными методами, и мы обсуждаем возможные объяснения этого факта.

7.1. Структура класса *NP*

В гл. 2 уже приводилась простая диаграмма класса *NP*, изображённая на рис. 7.1. Напомним, что класс *NP* в качестве подклассов содержит класс P и класс *NP*-полных задач (сокращённо *NPC*). В предположении $P \neq NP$ эти два класса не пересекаются.

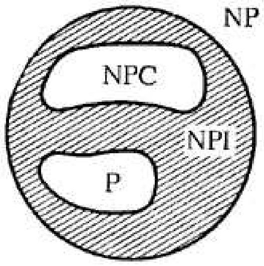


Рис. 7..1: *Уточнённая диаграмма класса NP (в предположении, что $P \neq NP$).*

Если $P \neq NP$, то *NP* не может совпадать также и с классом $P \cup NPC$. Обозначим через NPI класс $NP \setminus (P \cup NPC)$ задач, имеющих «промежуточную» сложность (между P и *NP*). То, что класс NPI не пуст (в предположении, что $P \neq NP$), есть следствие более общего результата, доказанного Ладнером [312]. Для того чтобы сформулировать этот результат, напомним, что ***рекурсивным*** языком называется любой язык, который можно распознать некоторой (не обязательно полиномиальной) ДМТ-программой, останавливающейся на всех входах.

Теорема 7.1. *Пусть B – такой рекурсивный язык, что $B \notin P$. Тогда существует распознаваемый за полиномиальное время язык $D \in P$, такой, что язык $A = D \cap B$ не принадлежит P, $A \propto B$, но не имеет места $B \propto A$.*

Эта теорема будет применяться следующим образом. Пусть B – произвольный *NP*-полный язык. Если $P \neq NP$, то $B \notin P$, поэтому условие теоремы выполняется. Более того, поскольку $D \in P$ и $B \in NP$, то получающийся язык A принадлежит *NP*. Из теоремы, следовательно, вытекает, что не имеет места $B \propto A$, так что A не может быть *NP*-полным, и поскольку $A \notin P$, то $A \in NPI$. Рассмотрим конкретный пример. Пусть B – это *NP*-полная задача ГАМИЛЬТОНОВ ЦИКЛ. Тогда теорема утверждает, что существует такой класс графов, распознаваемых за полиномиальное время, что задача ГАМИЛЬТОНОВ ЦИКЛ, будучи рассмотрена на классе этих графов, ие является *NP*-полной и не принадлежит классу P (в предположении, что $P \neq NP$).

Если $P \neq NP$, то теорема 7.1 утверждает не только непустоту класса NPI, но также даёт некоторую информацию о структуре этого класса промежуточных задач. В предположении $P \neq NP$ класс NPI должен состоять из бесконечного семейства различных классов эквивалентности языков. Например, если B – любая задача из класса NPI, то теорема 7.1 позволяет построить задачу, которая «проще», чем B, но всё ещё не принадлежит классу P. Ладнер [312] доказал также, что если $P \neq NP$, то класс NPI должен содержать такие пары языков C и D, что *ни* $C \propto D$ *ни* $D \propto C$. Таким образом, если $P \neq NP$, то в классе NP относительно частичного порядка, который определяется отношением $\propto$, должны быть несравнимые элементы.

После этих теоретических рассуждений можно задаться вопросом: существуют ли «естественные» задачи, являющиеся кандидатами на принадлежность классу NPI? В предположении $P \neq NP$ для указания представителей этого класса *может быть* использована теорема 7.1, но языки, которые при этом получаются, оказываются весьма неестественными из-за сложной техники «диагонализации», применяемой для их построения. Конечно, любая «открытая» задача из класса NP, т. е. задача, для которой не доказана ни принадлежность классу P, ни NP-полнота, может считаться кандидатом на принадлежность классу NPI. Однако некоторые открытые задачи принято считать более вероятными кандидатами, поскольку они выдержали проверку временем и им присущи некоторые свойства, видимо отличающие их от задач, NP-полнота которых уже доказана. В частности, три открытые задачи, приведённые в работе Карпа [280], остаются открытыми, и в настоящее время и их часто с различной степенью уверенности относят к задачам из класса NPI.

ИЗОМОРФИЗМ ГРАФОВ

УСЛОВИЕ. Заданы графы $G = (V, E)$ и $G' = (V, E')$.

ВОПРОС. Верно ли, что графы G и G' изоморфны, т. е. существует ли взаимно-однозначная функция $f : V \to V$, такая, что $\{u, v\} \in E$ тогда и только тогда, когда $\{f(u), f(v)\} \in E'$?

СОСТАВНЫЕ ЧИСЛА

УСЛОВИЕ. Задано положительное целое число K.

ВОПРОС. Существуют ли целые числа m и n; $m, n > 1$, такие, что $K = mn$?

ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ[1]

УСЛОВИЕ. Заданы целочисленные векторы $V_i = (v_i[1], ..., v_i[n])$, $1 \leqslant i \leqslant m$, $D = (d_1, ..., d_m)$, $C = (c_1, ..., c_m)$, целое число B.

ВОПРОС. Существует ли такой вектор $X = (x_1, ..., x_n)$ с рациональными координатами, что $V_i X \leqslant d_i$, $1 \leqslant i \leqslant m$ и $CX \geqslant B$?

Исследователи, предпринимавшие попытки доказать, что задача ИЗОМОРФИЗМ ГРАФОВ является NP-полной, отмечают, что её структура гораздо более ограничительна, чем структура типичной NP-полной задачи,

[1] Л. Г. Хачиян [33*] доказал, что эта задача лежит в классе P. – *Прим. ред.*

такой, например, как ИЗОМОРФИЗМ ПОДГРАФУ. Доказательства NP-полноты требуют определённой свободы действий в том смысле, что если искомая структура X (подмножество, перестановка, расписание и т. д.) существует, то она должна существовать и тогда, когда некоторые детали рассматриваемой индивидуальной задачи локально изменяются. Например, функция f будет изоморфизмом графа H и подграфа графа G, даже если в G добавить или исключить некоторые рёбра, не принадлежащие образу отображения f. Однако если f есть изоморфизм H и G, то любое изменение в G должно сопровождаться соответствующим изменением в H, в противном случае f уже не будет изоморфизмом. Другими словами, доказательства NP-полноты, видимо, требуют определённой избыточности задачи, отсутствующей в задаче ИЗОМОРФИЗМ ГРАФОВ. К сожалению, отсутствие избыточности мало помогает как при построении полиномиального алгоритма для решения задачи ИЗОМОРФИЗМ ГРАФОВ, так и при доказательстве того, что эта задача лежит в классе NPI.

Аргументы в пользу того, что задачи СОСТАВНЫЕ ЧИСЛА и ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ, видимо, не являются NP-полными, имеют другой источник. Напомним, что в гл. 2 было отмечено, что из принадлежности задачи Π классу NP, по-видимому, не следует, что в классе NP лежит дополнение Π^C этой задачи (т. е. задача с противоположным ответом: $Y_{\Pi^C} = D_\Pi \setminus Y_\Pi$). Определим

$$co{-}NP = \{\Pi^C : \Pi \in NP\}$$

или в терминах языков

$$co{-}NP = \{\Sigma^* \setminus L : L - \text{язык в алфавите } \Sigma \text{ и } L \in NP\}.$$

Основываясь на том, что многие задачи из класса $co{-}NP$, видимо, не принадлежат NP, вполне разумно предположить, что $NP \neq co{-}NP$. Отметим, что, хотя аргументы в пользу этого предположения близки к аргументам, приводившимся в пользу того, что $P \neq NP$, предположение $NP \neq co{-}NP$ «сильнее», чем $P \neq NP$. Класс P замкнут относительно операции взятия дополнения (т. е. $P = co - P$), поэтому из того, что $NP \neq co{-}NP$, следует, что $P \neq NP$, хотя неравенство $P \neq NP$ может иметь место и в том случае, когда $NP = co{-}NP$. Тем не менее следующая несложная теорема показывает, что имеется тесная связь между NP-полнотой задач СОСТАВНЫЕ ЧИСЛА и ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ и предположением, что $NP \neq co{-}NP$.

Теорема 7.2. *Если существует такая NP-полная задача П, что* $\Pi^C \in NP$, *то* $NP = co{-}NP$.

Доказательство этой теоремы аналогично доказательству леммы 2.1 и получается непосредственно с помощью операции композиции машин Тьюринга. В предположении, что $P \neq NP$ и $NP \neq co{-}NP$, теорема 7.2 даёт нам новую картину класса NP и его окружения, изображённую на рис. 7.2.

В качестве следствия из теоремы 7.2 получаем, что задача $\Pi \in NP$, для которой Π^C принадлежит классу NP, не может быть NP-полной, если, конечно, $NP \neq co-NP$. Поэтому было бы естественно ожидать, что такая задача Π не является NP-полной. Однако и задача СОСТАВНЫЕ ЧИСЛА, и задача ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ обладают этим свойством.

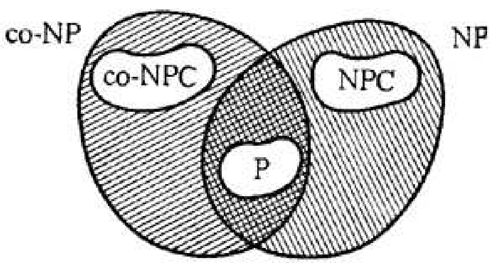


Рис. 7..2: *Картина класса NP и его непосредственных соседей в предположении, что $P \neq NP$ и $NP \neq co-NP$. Класс P может совпадать, а может и не совпадать с классом $NP \cap co-NP$.*

Очевидно, что задача СОСТАВНЫЕ ЧИСЛА лежит в классе NP. Дополнительной к ней задачей будет задача ПРОСТЫЕ ЧИСЛА: «Дано положительное целое число K. Верно ли, что K – простое?» Было [434] показано, что ПРОСТЫЕ ЧИСЛА принадлежит классу NP, а именно было доказано, что для любого простого числа K имеется доказательство простоты этого числа, достаточно короткое для того, чтобы его можно было указать и проверить за время, ограниченное полиномом от $\log K$. Аналогичным образом с помощью фундаментальной теоремы двойственности из теории линейного программирования можно доказать, что задача, дополнительная к задаче ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ, представляет собой вариант задачи линейного программирования и, значит, принадлежит классу NP.

Таким образом, если $NP \neq co-NP$, то ни задача СОСТАВНЫЕ ЧИСЛА, ни задача ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ не могут быть NP-полными. Этот факт можно считать довольно сильным аргументом в пользу того, что указанные задачи не являются NP-полными. Если ни одна из них не может быть решена за полиномиальное время, то они принадлежат классу NPI. Следующий аргумент, видимо, несколько слабее. Как уже отмечалось, симплекс-метод для решения задачи линейного программирования, хотя и имеет экспоненциальную временýю сложность, очень часто работает настолько быстро, что возникает ощущение, что можно построить другой алгоритм решения задачи линейного программирования, который уже является полиномиальным. В случае задачи СОСТАВНЫЕ ЧИСЛА мы в некотором смысле даже ближе к построению полиномиального алгоритма.

Алгоритм для выяснения вопроса, является заданное число простым или составным, предложен в работе Миллера [385]. В этой работе также доказано, что если верна «расширенная гипотеза Римана» из теории чисел, то алгоритм Миллера имеет полиномиальную временну́ю сложность. Таким образом, хотя ни для одной из указанных задач не известно полиномиального алгоритма[2], мы почти уверены, что никто не будет утверждать, что эти задачи NP-полны. (На самом деле доказательство того, что $\Pi \in NP$ и $\Pi^C \in NP$, может рассматриваться как указание на то, что задача Π *может* быть решена полиномиальным алгоритмом, хотя мы пока и не знаем, верно ли, что $P = NP \cap co{-}NP$.)

Наличие таких (более или менее вероятных) кандидатов на принадлежность классу NPI позволяет построить новые классы «трудных» задач в классе NP. В частности, отношение полиномиальной сводимости $\propto$ позволяет построить классы эквивалентности для каждой из задач ИЗОМОРФИЗМ ГРАФОВ, ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ (LP) и СОСТАВНЫЕ ЧИСЛА с очевидными следствиями. Например, если задача $\Pi \in NP$ обладает тем свойством, что $\Pi \propto LP$ и $LP \propto \Pi$, то Π разрешима за полиномиальное время (или NP-полна, или лежит в классе NPI) тогда и только тогда, когда LP разрешима за полиномиальное время (соответственно NP-полна или лежит в классе NPI).

Обычно нетрудно видеть, что формулировки задачи линейного программирования, которые встречаются в самых разных работах по линейному программированию, эквивалентны относительно отношения $\propto$ задаче LP. Рисс и Добкин [448] резюмировали основные результаты относительно этого класса задач и несколько расширили его, Итаи [251] показал, что некоторые задачи о рациональных потоках в сетях также эквивалентны задаче LP. Задачи, полиномиально эквивалентные задаче ИЗОМОРФИЗМ ГРАФОВ, были найдены Бусом [52], Бабаи [23], Миллером [386], Козеном [300, 304] и другими авторами. Однако большинство найденных задач этого класса либо являются ограниченными версиями задачи ИЗОМОРФИЗМ ГРАФОВ, либо задачами об изоморфизме объектов и структур другого типа, так что все эти задачи одного и того же «типа». Во многих работах по теории чисел формулируются утверждения справедливые тогда и только тогда, когда некоторые числа являются простыми, что позволяет построить класс задач, эквивалентных задаче СОСТАВНЫЕ ЧИСЛА. Некоторые из этих эквивалентностей приведены среди открытых задач в приложении.

Если имеется задача из класса NP, которую мы считаем трудной, но NP-полноту которой мы доказать не можем, то существуют, конечно, более сильные способы подтверждения её труднорешаемости, чем доказательство эквивалентности этой задачи одной из открытых задач. Один из методов, упомянутый в гл. 5, заключается в использовании сводимости по Тьюрингу. Если удастся доказать, что некоторая известная задача Π' сводима по Тьюрингу за полиномиальное время к рассматриваемой задаче Π, то, очевидно, Π, подобно Π', разрешима за полиномиальное время тогда и только тогда, когда $P = NP$, хотя мы и не можем доказать NP-полноту задачи Π, поскольку не известно, следует ли из сводимости $\Pi' \propto_T \Pi$ сводимость

[2] См. предыдущую сноску – *Прим. ред.*

$\Pi' \propto \Pi$. (В общем случае из полиномиальной сводимости по Тьюрингу *не* следует полиномиальная сводимость [316], хотя такие примеры в классе *NP* не известны.)

Потенциально более полезная идея предложена в работе [3]. Она состоит в использовании понятия «γ-сводимость». Это понятие позволяет доказывать труднорешаемость задачи не в предположении, что $P \neq NP$, а в предположении, что $NP \neq co{-}NP$. Таким образом, понятие γ-сводимости слабее понятий сводимости по Тьюрингу и полиномиальной сводимости, но оно позволяет дать существенное подтверждение предположению о том, что рассматриваемая задача трудна.

В отличие от других рассмотренных нами понятий сводимости понятие γ-сводимости имеет недетерминированный характер. Определим вначале понятие *отношения* R_M, вычисляемого НДМТ-программой:

$$R_M = \left\{ < x, y >: \begin{array}{r} \text{существует слово } z, \text{ такое, что на входе } x \\ \text{и догадке } z \text{ программа } M \text{ выдаёт } y \end{array} \right.$$

(здесь «выдаёт» означает то же самое, что при вычислении функций ДМТ-программами).

Будем говорить, что язык L_1 в алфавите Σ γ-*сводится* к языку L_2 в алфавите Σ_2 (обозначение: $L_1 \propto_\gamma L_2$), если существует такая полиномиальная НДМТ-программа M, что для каждого $x \in \Sigma^*$ существует некоторое слово $y \in \Sigma_2^*$, такое, что $< x, y > \in R_M$, и для всех $< x, y > \in R_M$ выполняется соотношение $x \in L_1$ тогда и только тогда, когда $y \in L_2$. Другими словами, на каждом входе x программы M имеется по крайней мере одно останавливающееся вычисление, и все останавливающиеся на x вычисления выдают элементы из L_2 тогда и только тогда, когда $x \in L_1$.

Нетрудно видеть, что отношение γ-сводимости $\propto_\gamma$, подобно отношению полиномиальной сводимости $\propto$, транзитивно, а также, что из $L_1 \propto L_2$ следует $L_1 \propto_\gamma L_2$ (хотя не известно, верно ли обратное). Язык $L \in NP$ называется γ-*полным*, если для всех $L' \in NP$, $L' \propto_\gamma L$. Таким образом, все *NP*-полные языки являются также и γ-полными, но в принципе язык может быть γ-полным, но не *NP*-полным. Примеры γ-полных задач, *NP*-полнота которых не известна, можно найти в работах [431,3]. Ниже приводится только один пример, взятый из последней работы.

ЛИНЕЙНАЯ ДЕЛИМОСТЬ

УСЛОВИЕ. Заданы положительные целые числа a и c.

ВОПРОС. Существует ли такое целое число x, что c делится на $ax + 1$?

(Интересно отметить, что эта задача очень легко решается за полиномиальное время, если $ax + 1$ заменить на ax, поскольку в последнем случае задача может быть переформулирована так: «Делится ли c на a?» Заметим также, что рассматриваемую задачу можно решить за псевдополиномиальное время, и, следовательно, её предполагаемая труднорешаемость существенно зависит от соглашения, что числа представляются словами, длина которых ограничена полиномом от логарифмов величин этих чисел.)

Доказательство γ-полноты задачи ЛИНЕЙНАЯ ДЕЛИМОСТЬ основано на γ-сведении к ней задачи 3-ВЫП. При этом используется недетерминированное угадывание большого числа простых чисел, доказательств простоты и разложений на простые множители. На самом деле все доказанные на сегодняшний день результаты о γ-полноте касаются теоретико-числовых задач и используют недетерминированность для угадывания теоретико-числовых структур. Поэтому возможность широкого применения понятия γ-сводимости ещё требует своего подтверждения. Применение понятия γ-полноты к вопросам труднорешаемости основано на следующей теореме [3].

Теорема 7.3. *Если язык L является γ-полным и $L \in NP \cap co{-}NP$, то $NP = co{-}NP$.*

В частности, поскольку $P \subseteq NP \cap co{-}NP$, то из этой теоремы следует, что если задача ЛИНЕЙНАЯ ДЕЛИМОСТЬ разрешима за полиномиальное время, то $NP = co{-}NP$. (Заметим также, что в силу теоремы 7.3 γ-полнота задач СОСТАВНЫЕ ЧИСЛА и ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ мало вероятна, так как они обе принадлежат классу $NP \cap co{-}NP$.) На рис. 7.3 в общую картину класса NP включены также γ-полные задачи.

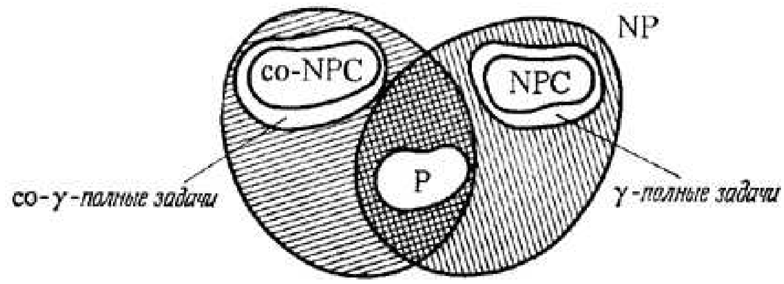


Рис. 7..3: *Ещё раз уточнённая картина класса NP (в предположении, что $P \neq NP$ и $NP \neq co{-}NP$).*

До сих пор довольно подробно обсуждалась структура класса NP и особенно класса промежуточных задач NPI. В заключение настоящего раздела рассмотрим структуру класса NPC (класса NP-полных задач). Несмотря на то что все эти задачи «эквивалентны» относительно полиномиальной сводимости, некоторые задачи из этого класса связаны между собой гораздо теснее, чем другие. Вспомним, например, указанную в гл. 3 прямую сводимость друг к другу задач ВЕРШИННОЕ ПОКРЫТИЕ, КЛИКА и НЕЗАВИСИМОЕ МНОЖЕСТВО. Эту тесную связь между задачами можно сформулировать с помощью понятия «изоморфизм с полиномиальным временем». Будем говорить, что два языка $L_1 \in \Sigma_1^*$ и $L_2 \in \Sigma_2^*$ *изоморфны с полиномиальным временем (или полиномиально изоморфны)*, если существует взаимно-однозначное отображение «на» $f : \Sigma_1^* \to \Sigma_2^*$, такое, что f полиномиально сводит L_1 к L_2, а f^{-1} полиномиально сводит L_2 к L_1.

Легко проверить, что задачи ВЕРШИННОЕ ПОКРЫТИЕ, КЛИКА и НЕЗАВИСИМОЕ МНОЖЕСТВО полиномиально изоморфны; можно также предположить, что с помощью отношения полиномиального изоморфизма можно построить и другие классы эквивалентности. Исследования в этом направлении были проведены в работах [39, 200]. При этом был установлен удивительный факт – *все* известные NP-полные задачи оказываются полиномиально изоморфными! В этих работах даны весьма общие методы преобразования обычной полиномиальной сводимости в полиномиальной изоморфизм, и в настоящее время не известно ни одной пары NP-полных задач, сводимость между которыми нельзя было бы преобразовать в изоморфизм, хотя, конечно, получаемые при этом «изоморфизмы» оказываются совсем не такими простыми, как изоморфизм между задачами ВЕРШИННОЕ ПОКРЫТИЕ, КЛИКА и НЕЗАВИСИМОЕ МНОЖЕСТВО. На основе этого Берман и Хартманис предположили, что все NP-полные задачи полиномиально изоморфны. Подобно многим другим, упоминавшимся выше предположениям, из этого предположения следует, что $P \neq NP$. Действительно, если бы P совпадало с NP, то все языки из P были бы NP-полными и, в силу предположения, полиномиально изоморфными. Однако P содержит как конечные, так и бесконечные языки, поэтому все языки в P не могут быть изоморфными.

7.2. Полиномиальная иерархия

В предыдущем разделе были рассмотрены NP-полные задачи и задачи, которые, возможно, «проще» их. В настоящем и следующих двух разделах будут обсуждаться NP-трудные задачи, которые по крайней мере не «проще» NP-полных задач. Стандартный пример NP-трудной задачи, которая, видимо, не является NP-лёгкой, даёт следующая задача из работ [384, 500].

МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ

УСЛОВИЕ. Задано правильно построенное булевское выражение E, содержащее литералы из множества переменных V, постоянные T (истина) и F (ложь), логические связки $\vee$(и), $\wedge$ (или), $\neg$ (не) и $\rightarrow$ (следует), а также неотрицательное целое число K.

ВОПРОС. Существует ли правильно построенное булевское выражение E', содержащее не более K вхождений литералов и эквивалентное E, т. е. такое, что для всех наборов значений истинности на множестве V выражения E и E' принимают одинаковые истинностные значения?

Эта задача NP-трудная, поскольку к ней сводится по Тьюрингу задача ВЫПОЛНИМОСТЬ (выполнимое выражение E, имеющее вид конъюнктивной нормальной формы, либо ложно при любом наборе значений истинности и, значит, эквивалентно выражению F, либо не эквивалентно никакому выражению, не содержащему литералов).

Однако, по-видимому, задача МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ не является NP-лёгкой. До сих пор никому не удалось показать, что эту задачу можно решить за полиномиальное время с использованием в качестве оракула задачу ВЫПОЛНИМОСТЬ (или любую другую задачу класса *NP*). Такой оракул может оказаться полезным для проверки эквивалентности выражений E' и E, но не для отыскания подходящего кандидата для E'.

Задачу МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ, однако, можно решить за полиномиальное время, если вместо детерминированной ОМТ из гл. 5 использовать *недетерминированную* оракульную машину Тьюринга (НОМТ). НОМТ представляет собой НДМТ, дополненную оракульной лентой, аналогично тому, как ОМТ есть ДМТ, дополненная оракульной лентой. Иначе говоря, НОМТ может с самого начала сделать догадку, а также в процессе вычисления консультироваться с оракулом. Неформально говоря, НОМТ с оракулом П соответствует недетерминированному алгоритму с подпрограммой для решения задачи П. Время работы НОМТ определяется так же, как для ОМТ и НДМТ.

Задачу МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ можно решить за полиномиальное время на НОМТ, если в качестве оракула для класса *NP* взять задачу ВЫПОЛНИМОСТЬ БУЛЕВСКОГО ВЫРАЖЕНИЯ (т.е. при заданном правильно построенном булевском выражении E спрашивается, существует ли такой набор значений истинности, при котором E принимает значение «истина»). Последняя задача, хотя и представляет собой несколько более общую версию стандартной задачи ВЫПОЛНИМОСТЬ, всё равно принадлежит классу *NP*. Для того чтобы узнать ответ для фиксированной индивидуальной задачи из задачи МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ, условие которой состоит из E и K, достаточно угадать выражение E', содержащее не более K вхождений литералов, и с помощью оракула определить выполнимость выражения $\neg\left((E' \to E)' \wedge (E \to E')\right)$. Если это выражение невыполнимо, то E' – искомое выражение.

Только что приведённую недетерминированную программу с оракулом можно назвать «(полиномиальной) недетерминированной сводимостью по Тьюрингу» задачи МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ к задаче ВЫПОЛНИМОСТЬ БУЛЕВСКОГО ВЫРАЖЕНИЯ. Эта сводимость наводит на мысль определить новый, более широкий класс задач, которые за полиномиальное время недетерминированно сводимы по Тьюрингу к задачам класса *NP*. Для этой цели введём соответствующие обозначения. Пусть Y – некоторый класс языков. Классы языков P^Y и NP^Y определяются следующим образом:

$$P^Y = \{L: \text{ существует такой язык } L' \in K, \text{ что } L \propto_T L'),$$

$$NP^Y = \left\{ L: \begin{array}{c} \text{существует такой язык } L' \in K, \text{ что сущес-} \\ \text{твует полиномиальная недетерминирован-} \\ \text{ная сводимость по Тьюрингу языка } L \text{ к } L' \end{array} \right\}.$$

Заметим, что P^{NP} есть просто класс всех NP-простых языков. Класс NP^{NP} содержит язык МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ. Не известно, отличаются или совпадают классы P^{NP} и NP^{NP}. Отношения включения между двумя построенными классами и классами P, NP, $co-NP$ изображены на рис. 7.4.

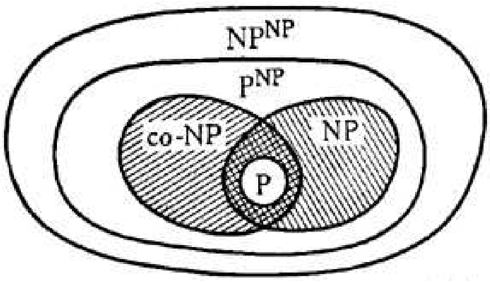


Рис. 7..4: *Отношения включения между классами языков, в том числе новыми классами P^{NP} и NP^{NP} (в предположении, что $P \neq NP$).*

На основе этой картины взаимоотношения классов языков Мейер и Стокмейер [384] заметили, что процесс определения новых классов языков исходя из старых может быть продолжен до бесконечности, при этом будут получаться классы, по-видимому, всё более сложных языков. В результате возникает так называемая *полиномиальная иерархия.* Классы языков этой иерархии обозначаются через Σ_K^p, Π_K^p и Δ_K^p (где верхний индекс p используется только для того, чтобы отличать эти классы от аналогичных классов в арифметической иерархии Клини (см. Роджерс [453]) и определяются следующим образом:

$$\Sigma_0^p = \Pi_0^p = \Delta_0^p = P,$$

для всех $k \geqslant 0$

$$\Delta_{K+1}^p = P^{\Sigma_k^p},$$

$$\Sigma_{K+1}^p = NP^{\Sigma_k^p},$$

$$\Pi_{K+1}^p = co - \Sigma_{k+1}^p,$$

(т. е. класс Π_{k+1}^p состоит из всех задач, дополнительных задачам класса Σ_{k+1}^p. На самом деле $\Pi_1^p = co-NP$, $\Sigma_1^p = NP$ и $\Delta_1^p = P$. Аналогично, $\Delta_2^p = P^{NP}$, $\Sigma_2^p = NP^{NP}$. На рис. 7.5 показаны взаимоотношения между классами языков этой иерархии.

Полиномиальная иерархия позволяет более детально классифицировать NP-трудные задачи распознавания свойств. Во-первых, можно спросить, содержится ли вообще рассматриваемая задача в иерархии (в разд. 7.4 будут указаны некоторые задачи, по-видимому не содержащиеся в этой иерархии, хотя до сих пор не известно даже, являются ли они труднорешаемыми). Для того чтобы выяснить, содержится ли рассматриваемая задача в

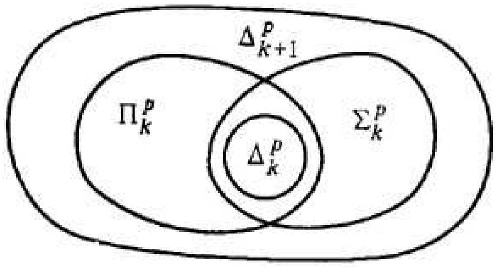


Рис. 7..5: *Взаимоотношения между классами полиномиальной иерархии.*

иерархии, полезно иметь прямой способ доказательства того, что эта задача принадлежит некоторому классу иерархии, без последовательного применения индуктивного определения. Для этого можно использовать понятие отношения таким же образом, как Карп [280] воспользовался понятием отношения для определения класса *NP*. Пусть Γ – некоторый алфавит; отношением размерности k над Γ^* называется множество R, состоящее из наборов $< z_1, z_2, \ldots, z_k >$ из k слов $z_i, 1 \leqslant i \leqslant k$, в алфавите Γ. Будем говорить, что отношение распознаваемо за полиномиальное время, если найдётся детерминированная программа с полиномиальным временем работы, распознающая язык, который состоит в точности из всех наборов по k слов, принадлежащих R. Следующая теорема доказана в работе [536].

`Теорема 7.4.` *Пусть L – язык в алфавите Γ^*, $|\Gamma| \geqslant 2$. Для любого $k \geqslant 1$ язык $L \in \Sigma^p_k$ тогда и только тогда, когда существуют такие полиномы $p_1, p_2, \ldots, p_k$ и отношение R над Γ^* размерности $k+1$, распознаваемое за полиномиальное время, что для всех $x \in \Gamma^*$ имеет место*

$$\begin{array}{ll} x \in L \Leftrightarrow & (\exists\ y_1 \in \Gamma^*,\ |y_1| \leqslant p_1(|x|) \\ & (\forall\ y_2 \in \Gamma^*,\ |y_2| \leqslant p_2(|x|) \\ & \ldots\ldots\ldots\ldots \\ & (Qy_k \in \Gamma^*,\ |y_k| \leqslant p_k(|x|) \\ & [< x, y_1, y_2, \ldots, y_k > \in R], \end{array}$$

где квантор Q перед переменной y_k есть $\exists$, если k нечётно, и $\forall$, если k чётно, а промежуточные кванторы чередуются.

Отметим, что задача МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ представляется в таком виде при $k = 2$, если отношение R определить так: $< (x, K), y_1, y_2 > \in R$ тогда и только тогда, когда x и y_1 – правильно построенные булевские выражения, причем y_1 содержит не более K вхождений литералов, а y_2 – набор значений истинности, выполняющий выражение $(x \to y_1) \wedge (y_1 \to x)$. Можно получить аналогичное описание класса Π^p_k, если в теореме 7.4 все кванторы $\exists$ и $\forall$ поменять местами (см. [536]).

Сформулированная теорема даёт способ определения верхней границы для наименьшего из таких k, что $L \in \Sigma_k^p$. Определение нижней границы (хотя бы даже условное) представляется более трудной задачей. Один результат в этом направлении был получен Легетом [333], он состоит в том, что если $NP \neq co{-}NP$, то некоторые задачи распознавания, связанные с оптимизационными задачами, не могут лежать ни в NP, ни в $co{-}NP$. Эти задачи распознавания отличаются от стандартных тем, что в них формулируются некоторые утверждения относительно оптимальности. Простым примером является следующая задача.

КЛИКА МАКСИМАЛЬНОГО РАЗМЕРА

УСЛОВИЕ. Заданы граф G и положительное целое число K.

ВОПРОС. Верно ли, что наибольший полный подграф графа G содержит ровно K вершин?

В отличие от вопроса: «Существует ли клика размера не менее K?», – который формулируется в задаче КЛИКА, в рассматриваемой задаче спрашивается: «Будет ли K размером максимальной клики?» Задача распознавания оптимальности, соответствующая задаче МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ, формулируется следующим образом.

РАЗМЕР МИНИМАЛЬНОГО ЭКВИВАЛЕНТНОГО ВЫРАЖЕНИЯ

УСЛОВИЕ. Заданы правильно построенное булевское выражение E и неотрицательное целое число K.

ВОПРОС. Верно ли, что не существует правильно построенного булевского выражения, эквивалентного E и содержащего не более $K-1$ литералов, но существует выражение, эквивалентное E и содержащее ровно K литералов?

В результатах Легета ключевым понятием является понятие «недетерминированная полиномиальная табличная сводимость», которая обозначается через $\propto_c^{NP}$ и определена в работе [316]. Смысл этого понятия заключается в том, что $L \propto_c^{NP} L'$, если некоторая недетерминированная оракульная программа, останавливающаяся в состоянии «нет», когда её оракул говорит «нет», может проверить принадлежность языку L, с использованием оракула для L'. Основная теорема Легета [333] в применении к классу NP формулируется следующим образом.

Теорема 7.5. *Если L_0, L_1 и L_2 – такие языки, что L_1 и L_2 NP- полны, $L_1^c \propto_c^{NP} L_0$ и $L_2^c \propto_c^{NP} L_0^c$ (где L^c обозначает язык дополнительный к L), то*

$$L_0 \in NP \cup co{-}NP \Rightarrow NP = co{-}NP.$$

Эта теорема применима к задачам КЛИКА МАКСИМАЛЬНОГО РАЗМЕРА и РАЗМЕР МИНИМАЛЬНОГО ЭКВИВАЛЕНТНОГО ВЫРАЖЕНИЯ, а также ко многим другим подобным задачам распознавания. Если эту теорему применять к задаче КЛИКА МАКСИМАЛЬНОГО РАЗМЕРА, то в качестве языков L_1 и L_2 можно взять язык, соответствующий задаче

КЛИКА. Нетрудно показать, что задача выяснения того, что граф не содержит полного подграфа на K вершинах, сводится как к задаче КЛИКА МАКСИМАЛЬНОГО РАЗМЕРА, так и к её дополнению.

Комбинируя теоремы 7.4 и 7.5, можно довольно точно указать место рассматриваемой задачи в полиномиальной иерархии, на нижних уровнях которой расположено большинство естественно возникающих задач. Например, на основе этих теорем можно заключить, что если $P \neq co{-}NP$, то задача КЛИКА МАКСИМАЛЬНОГО РАЗМЕРА лежит в классе $\Delta_2^p \setminus (\Sigma_1^p \cup \Pi_1^p)$, т. е. NP-проста, но не принадлежит ни NP, ни $co{-}NP$.

В основу попыток классификации задач, расположенных на более высоких уровнях иерархии, положено обобщение понятия NP-полноты задачи на соответствующем уровне иерархии. Если удастся выявить «самую трудную» задачу класса Σ_k^p, то эта задача не будет принадлежать классу Σ_{k-1}^p, если, конечно, эти классы не совпадают. Понятие «самой трудной» задачи нам уже знакомо. Язык L будем называть полным в классе Σ_k^p (относительно полиномиальной сводимости), если $L \in \Sigma_k^p$ и для всех $L' \in \Sigma_k^p$, $L' \propto L$. Аналогичные определения полноты можно дать для классов Π_k^p и Δ_k^p. Легко показать, что если язык L полон в классе Σ_k^p и $L \in \Sigma_{k-1}^p$, то $\Sigma_k^p = \Sigma_{k-1}^p$ (аналогичные утверждения имеют место и для классов Π_k^p и Δ_k^p). Таким образом, доказательство полноты задачи в некотором классе иерархии – это достаточно эффективный способ выяснения места этой задачи в иерархии.

Как и в случае NP-полноты, было бы гораздо проще доказывать полноту в классе иерархии, если известна некоторая «первая» задача, полная в этом классе, т.е. задача, которая играла бы такую же роль, какую ВЫПОЛНИМОСТЬ играет в классе NP. Задачи B_k, полные в классах Σ_k^p для всех $k \geqslant 1$, были впервые указаны в работе [384]. Индивидуальная задача из B_k представляет собой правильно построенное булевское выражение E с множеством переменных $X = \{x[i,j] : 1 \leqslant i \leqslant k,\ 1 \leqslant j \leqslant m_i\}$, где $m_1, m_2, ..., m_k \geqslant 0$ – некоторые целые числа. Вопрос заключается в истинности следующего выражения:

$$
\begin{aligned}
&(\exists\, x[1,1]) \dots (\exists\, x[1,m_1])\\
&\quad(\forall\, x[2,1] \dots (\forall\, x[2,m_2])\\
&\qquad \dots \quad \dots \quad \dots \quad \dots\\
&\qquad\quad (Qx[k,1]) \dots (Qx[k,m_k])E,
\end{aligned}
$$

где квантор Q есть квантор $\exists$ при k нечётном и $\forall$ при k чётном, а остальные кванторы чередуются согласно значению первого параметра переменной x.

Отметим, что по теореме 7.4 $B_k \in \Sigma_k^p$. Следующий результат доказан в работах [384] и [536].

Теорема 7.6. *Для всех $k \geqslant 1$ язык B_k полон относительно полиномиальной сводимости в классе Σ_k^p, а его дополнение – язык B_k^c полон относительно полиномиальной сводимости в классе Π_k^p.*

Таким образом, учитывая транзитивность отношения $\propto$ для доказательства полноты задачи П в классе Σ_k^p, достаточно показать, что $\Pi \in \Sigma_k^p$ и $B_k \propto \Pi$. Несмотря на это, до сих пор получено мало результатов подобного типа. Козен [301] показал существование аналогичной иерархии полных задач, формулируемых не в терминах булевских выражений, а в терминах конечно представленных алгебр, но результатов об индивидуальных задачах, представляющих самостоятельный интерес, мало. Один из них, относящийся к задаче НЕЭКВИВАЛЕНТНОСТЬ ЦЕЛОЧИСЛЕННЫХ ВЫРАЖЕНИЙ, получен в работе [500]. Если $n \in Z^+$, то двоичная запись числа n называется целочисленным выражением, представляющим множество $\{n\}$. Если e и f – целочисленные выражения, представляющие множества E и F, то $(e \cup f)$ называется целочисленным выражением, представляющим $E \cup F$, а $(e+f)$ называется целочисленным выражением, представляющим множество $\{m+n : m \in E, n \in F\}$. Индивидуальная задача из задачи НЕЭКВИВАЛЕНТНОСТЬ ЦЕЛОЧИСЛЕННЫХ ВЫРАЖЕНИЙ – это пара $< e, f >$ целочисленных выражений, а вопрос формулируется следующим образом: «Верно ли, что e и f представляют различные подмножества Z^+?». *Стокмейер* [500] доказал, что эта задача полна в классе Σ_2^p. Кроме этого результата, даже о классе Σ_2^p известно немного. Не известно, например, верно ли, что задача МИНИМАЛЬНОЕ ЭКВИВАЛЕНТНОЕ ВЫРАЖЕНИЕ полна в классе Σ_2^p?

Не следует, однако, слишком переживать из-за отсутствия результатов в этом направлении, так как различия задач, вытекающие из структуры полиномиальной иерархии, видимо, имеют больше теоретическое, нежели практическое значение. Это объясняется следующей теоремой Стокмейера [500].

Теорема 7.7. *Если для некоторого $k \geqslant 1$ имеет место равенство $\Sigma_k^p = \Pi_k^p$, то $\Sigma_j^p = \Pi_j^p = \Sigma_k^p$ для всех $j \geqslant k$.*

В частности, если $P = NP$, то $NP = \Sigma_1^p = \Pi_1^p$ и поэтому $\Sigma_j^p = P$ для всех $j \geqslant 0$. Пусть $PH = \bigcup_{j=1}^{\infty} \Sigma_j^p$ – множество всех языков полиномиальной иерархии. Тогда $P = NP$ тогда и только тогда, когда $PH = P$, т. е. тогда и только тогда, когда все языки иерархии можно распознать за полиномиальное время и вся иерархия сжимается в один класс P.

Таким образом, утверждение, что NP-трудная задача П лежит в классе PH, хотя и не эквивалентно NP-лёгкости задачи П, но имеет те же следствия: П можно решить за полиномиальное время тогда и только тогда, когда $P = NP$. Тонкие свойства задачи П, зависящие от её положения в иерархии, с вычислительной точки зрения, видимо, не имеют большого значения ещё и потому, что можно доказать, что любая задача класса PH может быть решена перебором за детерминированное время $O(2^{q(n)})$, где q – некоторый полином, n – длина записи входа (этот же результат для класса NP был доказан в теореме 2.1).

Тем не менее в предположении $P \neq NP$ полиномиальная иерархия представляет теоретический интерес, а теорема 7.7 прибавляет к списку фундаментальных открытых вопросов ещё один: «Верно ли, что PH – бесконечная иерархия, т. е. верно ли, что $\Sigma_{k+1}^p \setminus \Sigma_k^p$ не пусто для любого $k \geqslant 1$?» Заметим, вполне может оказаться, что $P \neq NP$, $NP \neq co{-}NP$ и в то же время вся иерархия сжимается в один класс Σ_2^p, хотя такой результат и был бы весьма удивителен.

7.3. Сложность задач перечисления

Задачи перечисления относятся к числу естественных кандидатов на роль труднорешаемых задач даже в том случае, если $P = NP$. Для того чтобы определить такое понятие, как задача перечисления, напомним понятие переборной задачи из гл. 5. В переборной задаче Π каждая индивидуальная задача $I \in D_\Pi$ имеет ассоциированное с ней множество решений $S_\Pi(I)$ и при заданном I требуется найти один элемент из множества $S_\Pi(I)$ (в соответствующей задаче распознавания спрашивается, пусто $S_\Pi(I)$ или нет). *Задача перечисления*, соответствующая переборной задаче Π, формулируется так: «Задано I, какова мощность множества $S_\Pi(I)$, т. е. *сколько* имеется решений?»

Со многими задачами распознавания, которые мы обсуждали, можно естественным образом связать задачи перечисления. Например, с задачей ГАМИЛЬТОНОВ ЦИКЛ можно связать следующую задачу перечисления: «Задан граф G, сколько различных гамильтоновых циклов имеется в G?» Задаче ВЫПОЛНИМОСТЬ соответствует задача: «Задано множество дизъюнкций C на множестве переменных V, сколько существует наборов значений истинности, выполняющих одновременно все дизъюнкции в C?»

Отметим, что в задаче перечисления не требуется предъявить все элементы множества $S_\Pi(I)$, необходимо только определить их количество. Таким образом, хотя число гамильтоновых циклов в графе G может быть экспоненциально велико (в зависимости от числа вершин) и для составления их списка потребуется экспоненциальное время, ответ задачи перечисления может быть записан полиномиальным числом двоичных знаков. Следовательно, «размер» ответа сам по себе не препятствует решению задачи за полиномиальное время.

В действительности некоторые нетривиальные задачи перечисления можно решить за полиномиальное время. В работе [198], например, показано, что задача перечисления: «Дан граф G, сколько в нём имеется различных остовных деревьев?» – может быть решена за полиномиальное время с помощью «матричной теоремы о деревьях» Кирхгофа и вычисления некоторых определителей. Подобным же образом задача: «Задан граф G, сколько в нём имеется эйлеровых путей?» – может быть решена за полиномиальное время.

Тем не менее многие задачи перечисления, по-видимому, очень сложны. Задачи перечисления, соответствующие NP-полным задачам, очевидным образом являются NP-трудными, поскольку если известна мощность множества $S_\Pi(I)$, то можно легко определить, пусто это множество или нет. Некоторые задачи перечисления, видимо, сложнее соответствующих задач существования. Даже в том случае, если $P = NP$ и за полиномиальное время можно определить, содержит ли произвольный граф гамильтонов цикл, совершенно не ясно, поможет ли это подсчитать за полиномиальное время число гамильтоновых циклов в графе G.

Исходя из этого наблюдения, Вэльянт [522] предложил рассмотреть новый класс полиномиально эквивалентных задач – класс «КР-полных задач»[3], включающий многие задачи перечисления и имеющий целью подчеркнуть дополнительную сложность задач перечисления. (Аналогичное предложение было сделано в работе [491].) Задача перечисления принадлежит классу КР, если найдётся недетерминированный алгоритм, такой, что для каждой индивидуальной задачи $I \in D_\Pi$ число различных «догадок», ведущих к принятию I, равно в точности $S_\Pi(I)$, а длина самого продолжительного принимающего вычисления ограничена полиномом от $Length[I]$. (Вэльянт [522] определил этот класс задач в терминах так называемых «считающих машин Тьюринга», которые аналогичны «пороговым машинам» из работ [490, 491] и вероятностным машинам Тьюринга из работы [171].)

Легко видеть, что рассмотренные выше задачи перечисления лежат в классе КР. В самом деле, если Π – произвольная переборная задача, для которой существует полином p, такой, что для всех $I \in D_\Pi$ и всех $\sigma \in S_\Pi(I)$ «длина» не превосходит $p(Length[I])$, а также для всех I и σ за полиномиальное время можно определить, верно ли утверждение $\sigma \in S_\Pi(I)$, то нетрудно видеть, что задача перечисления, соответствующая Π, лежит в классе КР. Таким образом, можно заключить, что класс КР содержит по крайней мере большую часть тех задач перечисления, которые было бы желательно рассмотреть, а также что среди них имеется много трудных задач. Понятие «полноты» в классе КР опять-таки используется для выявления «самых трудных» задач этого класса. Задача перечисления Π называется КР-*полной*, если $\Pi \in$ КР и для всех $\Pi' \in$ КР, $\Pi' \propto_T \Pi$.

Заметим, что в этом определении используется полиномиальная сводимость по Тьюрингу, а не просто полиномиальная сводимость. Это объясняется тем, что рассматриваемые в данном случае задачи являются не просто задачами распознавания, а имеют в качестве ответов числа. Можно, однако, рассмотреть следующий вариант понятия полиномиальной сводимости. Пусть имеются две переборные задачи Π и Π'. Будем говорить, что задача Π (полиномиально) *консервативно сводится* к задаче Π', если существует вычислимая за полиномиальное время функция $f : D_\Pi \to D_{\Pi'}$, такая, что для всех $I \in D_\Pi$, $|S_\Pi(I)| = |S_{\Pi'}(f(I))|$. Заметим, что консервативная сводимость задачи Π к задаче Π' в то же время является полиномиальной сводимостью одной задачи к другой. Более важно, однако, заметить, что консервативная сводимость задачи Π к задаче Π' автоматически даёт сводимость по Тьюрингу соответствующих задач перечисления. Таким образом, консервативная сводимость может служить ценным инструментом при доказательстве КР-полноты.

[3] Обозначение КР введено вместо используемого в оригинале символа $\#P$; K – первая буква от слова «количество». Здесь речь идёт о количестве элементов в перечисляемом множестве. – *Прим. ред.*

Этим инструментом впервые воспользовался *Саймон* [490]. Он заметил, что общая сводимость в доказательстве теоремы Кука может быть сделана консервативной, и таким образом получил следующий результат.

Теорема 7.8. *Задача подсчёта числа выполняющих наборов значений истинности для индивидуальной задачи ВЫПОЛНИМОСТЬ является КР-полной.*

Саймон [491] заметил также, что многие известные из литературы сводимости, использующиеся в доказательствах результатов об *NP*-полноте, являются консервативными, а в тех случаях, когда они не консервативны, можно найти консервативные сводимости (см. [521] и [135]). Отсюда можно заключить, что, например, задачи перечисления, соответствующие шести основным *NP*-полным задачам из гл. 3, являются КР-полными (хотя мы и не утверждаем, что сводимости, использовавшиеся в гл. 3, консервативны).

Может показаться удивительным тот факт, что некоторые задачи перечисления являются КР-полными, в то время как соответствующие переборные задачи *могут быть* разрешимы за полиномиальное время. Рассмотрим следующую задачу: «Задан двудольный граф G, спрашивается, сколько различных совершенных паросочетаний в нём содержится?» (Напомним, что двудольным графом $G = (V, E\ )$ называется граф, множество вершин которого разбито на два подмножества V_1 и V_2, так что любое ребро соединяет вершины только из разных подмножеств. Совершенным паросочетанием называется такое подмножество ребер E', $E' \subseteq E$, что любая вершина в V инцидентна ровно одному ребру из E'».) Эта задача перечисления особенно интересна потому, что она эквивалентна вычислению «перманента» матрицы 0-1 (перманент – аналог определителя, однако в соответствующую сумму все произведения элементов матрицы входят со знаком +). Указанная выше переборная задача хорошо известна ещё и потому, что это просто «задача о бракосочетании», упоминавшаяся в разд. 3.1.2, и, как там указано, разрешима за полиномиальное время. Тем не менее в работе [522] доказан следующий результат.

Теорема 7.9. *Задача о подсчёте числа различных совершенных паросочетаний в двудольном графе является КР-полной.*

Таким образом, в то время как для одних переборных задач, разрешимых за полиномиальное время, соответствующий задача перечисления не может быть решена за полиномиальное время при условии $P \neq NP$ (и даже в случае $P = NP$ это мало вероятно), для других задач, например задач об остовном дереве или об эйлеровом пути, упомянутых в начале настоящего раздела, соответствующие задачи перечисления могут быть решены за полиномиальное время. Было бы интересно продолжить эту классификацию. Частичные результаты в этом направлении содержатся в работе [260]. В ней, в частности, доказано, что некоторые задачи перечисления, основанные на переборных задачах, разрешимых за полиномиальное время (например, задаче K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО, обсуждавшейся в гл. 5), *NP*-трудны. Хотя в этой работе и не рассматриваются вопросы о КР-полноте, нетрудно показать, что обсуждаемые там задачи в действительности КР-полны. Другие КР-полные задачи можно найти также в работе [523].

7.4. Полнота с полиномиально ограниченной памятью

До сих пор нам встречался только один «ресурс», используемый для вычисления, – ***время***, необходимое для его выполнения. На практике в равной степени важен и другой ресурс – объём памяти ЭВМ, или число ячеек, необходимых для вычисления, который будет называться «ёмкостной» потребностью алгоритма. Время, требующееся при вычислении на машине Тьюринга, есть число шагов, выполняемых до момента перехода в заключительное состояние. Память, требующаяся при вычислении, есть число различных ячеек, которое при этом просматривает читающая/пишущая головка. Поскольку число просматриваемых ячеек не может превосходить числа шагов вычисления, то любая задача, разрешимая за полиномиальное время, разрешима также и с полиномиальной памятью. Более того, понятие полиномиальной памяти, как и понятие полиномиального времени, обладает свойством независимости от модели вычислительного устройства. Для всех реалистичных моделей ЭВМ класс задач, разрешимых на них с полиномиальной памятью, один и тот же. (В действительности в отношении памяти независимость от модели ЭВМ ещё более ярко выражена, а именно классы задач, разрешимых с памятью $O(n^k)$ для любого целого $k \geqslant 0$, одинаковы для всех стандартных моделей ЭВМ [215].) В настоящем разделе мы подробно рассмотрим взаимосвязь полиномиального времени и полиномиальной памяти.

Хотя все задачи, разрешимые за полиномиальное время, разрешимы также и с полиномиальной памятью, до сих пор не известно, существуют ли задачи, разрешимые с полиномиальной памятью, но *неразрешимые* за полиномиальное время. Гипотеза о существовании таких задач выглядит весьма правдоподобно, поскольку все задачи класса *NP*, все задачи полиномиальной иерархии и все задачи класса КР могут быть решены с полиномиальной памятью. Таким образом, если $P \neq NP$ или $P \neq$ КР, то вычисления с полиномиальной памятью не эквивалентны вычислениям с полиномиальным временем. Кроме того, имеется много задач, разрешимых с полиномиальной памятью, но, по-видимому, более сложных, чем задачи из перечисленных выше классов. Для того чтобы выделить «наиболее сложные» из этих задач, введём понятие «полноты» с полиномиальной памятью.

Давая формальные определения, мы вновь ограничимся языками (задачами распознавания). Обозначим через $P\text{-}SPACE$ класс всех языков, распознаваемых ДМТ-программой, использующей память, ограниченную полиномом и останавливающуюся на всех входах. Теперь понятие полноты определяется стандартно: язык L называется ***$P\text{-}SPACE$-полным*** (относительно полиномиальной сводимости), если $L \in P\text{-}SPACE$ и любой язык L'

из класса $P\text{-}SPACE$ сводится к $L, L' \propto L$. Из этого определения вытекает, что если язык L является $P\text{-}SPACE$-полным. то $L \in P$ тогда и только тогда, когда $P = P\text{-}SPACE$. Аналогичное утверждение будет верно, если заменить класс P на NP. Поскольку может оказаться, что $P = NP$, даже если $P \neq P\text{-}SPACE$, то $P\text{-}SPACE$-полнота задачи даёт более сильное свидетельство её труднорешаемости, чем NP-полнота. Факт $P\text{-}SPACE$-полноты задачи можно рассматривать так же, как указание о том, что задача не принадлежит ни классу NP, ни полиномиальной иерархии. В работе [536] показано, что если какая-нибудь $P\text{-}SPACE$-полная задача лежит в классе Σ_k^p для некоторого k, то вся иерархия сжимается до класса $P\text{-}SPACE$, т.е. $\Sigma_j^p = \Sigma_k^p = P\text{-}SPACE$ для всех $j \geqslant k$.

Фундаментальная $P\text{-}SPACE$-полная задача и её интересная связь с задачами B_k, полными в классе Σ_k^p, обсуждавшимися в разд. 7.2, были указаны в работе [503]. А именно, рассматривая задачи B_k как языки относительно одной общей схемы кодирования, можно получить $P\text{-}SPACE$-полный язык, полагая $B_w = \bigcup_{k=1}^{\infty} B_k$. Для большей ясности, однако, рассмотрим язык B_w как задачу распознавания, для чего дадим ему подходящее название и определение.

БУЛЕВСКИЕ ФОРМУЛЫ С КВАНТОРАМИ (БФК)

УСЛОВИЕ. Задана правильно построенная булевская формула

$$F = (Q_1x_1)(Q_2x_2)\,...\,(Q_nx_n)E,$$

где E – булевское выражение с переменными $x_1, x_2, ..., x_n$, а Q_i – один из кванторов «$\exists$ » или «$\forall$ ».

ВОПРОС. Верно ли, что F истинна?

Отметим, что существенное отличие задач БФК от B_k состоит в том, что в B_k допускаются индивидуальные задачи с заранее ограниченным числом чередования кванторов (либо за «$\forall$ » следует «$\exists$ », либо за «$\exists$ » следует «$\forall$ »). В БФК, напротив, число изменений кванторов не ограничено. Принадлежность БФК классу $P\text{-}SPACE$ следует из того, что истинность формулы F можно проверить, перебирая все возможные наборы значений истинности переменных $x_1, x_2, ..., x_n$ и вычисляя значения выражения E на этих наборах значений истинности. Запись проверяемого набора значений истинности, проверку истинности E и слежение за номером проверяемого набора значений истинности можно осуществить с использованием полиномиальной памяти, хотя для проверки всех $O(2^{p(n)})$ наборов значений истинности потребуется экспоненциальное время. То, что любой язык $L' \in P\text{-}SPACE$ может быть сведен к языку БФК, следует из аналога теоремы Кука, в котором вместо полиномиальных по времени вычислений моделируются вычисления, полиномиальные по памяти. В действительности число шагов вычисления может иметь порядок $O(2^{p(n)})$, где p – полином, n – длина входа (и в силу аналога теоремы 2.1, не может быть больше), эта неприятность преодолевается нетривиальным приёмом, предложенным в работах [503, 500]. Таким образом, можно получить следующий результат.

Теорема 7.10. *Задача БФК является P-SPACE-полной.*

На основе задачи БФК класс P-$SPACE$-полных задач был в значительной степени изучен. Хотя мы уже убедились в NP-трудности большинства элементов этого класса задач, следует принять во внимание дополнительное «подтверждение» труднорешаемости, которое даёт P-$SPACE$-полнота. Для P-$SPACE$-полной задачи мало вероятна принадлежность не только классу P, но и классу NP. Следовательно, свойства, проверка которых P-$SPACE$-полна, вероятно, не могут быть *проверены* за полиномиальное время даже с использованием «догадки» полиномиальной длины. Более того, с теоретической точки зрения даже приятно, что результаты о «полноте» в отличие от результатов о «трудности» позволяют довольно точно определить сложность задачи.

По мере расширения числа P-$SPACE$-полных задач класс P-$SPACE$-полных задач приобретает новые характеристики. Задачи, в нём содержащиеся, весьма отличаются от известных NP-полных задач. В заключение настоящего раздела приведем несколько P-$SPACE$-полных задач, которые, как мы надеемся, позволят читателю получить некоторое представление о том, как они выглядят.

В первую очередь заметим, что аналогично тому, как задача ВЫПОЛНИМОСТЬ сводится к задаче 3-ВЫП, задача БУЛЕВСКИЕ ФОРМУЛЫ С КВАНТОРАМИ может быть сведена к своему ограниченному варианту 3-ВЫП С КВАНТОРАМИ, в которой булевское выражение является индивидуальной задачей из 3-ВЫП (т. е. конъюнкцией 3 литеральных дизъюнкций). Так что и эта ограниченная задача P-$SPACE$-полна, см. [503]. В доказательстве используется приём из работы [31], который позволяет преобразовать произвольное булевское выражение, содержащее связки «$\rightarrow$» и «$\neg$ », в выражение нужного вида. Роль «основной» P-$SPACE$-полной задачи, аналогичную роли рассмотренных выше шести основных NP-полных задач, играет задача 3-ВЫП С КВАНТОРАМИ.

Особенно богатый источник P-$SPACE$-полных задач представляют собой комбинаторные игры. Игры, интересные с этой точки зрения, – это игры двух лиц. Их можно кратко описать, задав общую «позицию» в множестве всех возможных «позиций», критерий выявления игрока, имеющего «ход» в данной позиции, способ изменения позиции в результате «хода», указания заключительной позиции (и победителя), а также исходной позиции. Назовём игроков БЕЛЫЕ и ЧЁРНЫЕ, обозначим через P_0 исходную позицию, а также примем соглашение, что после достижения заключительной позиции все последующие ходы её не меняют. Утверждение, что БЕЛЫЕ, начиная игру, имеют форсированный выигрыш за n входов (n чётно), формализуется следующим образом:

Существует такой ход БЕЛЫХ из позиции P_0 в позицию P_1, что для всех ходов ЧЁРНЫХ из позиции P_1 в позицию P_2, существует такой ход БЕЛЫХ из позиции P_2 в позицию P_3, что для любого хода ЧЁРНЫХ из позиции P_3 в позицию P_4,

… … … …

существует такой ход БЕЛЫХ из позиции P_{n-2} в позицию P_{n-1} что для любого хода ЧЁРНЫХ из позиции P_{n-1} в позицию P_n, позиция P_n выигрышна для БЕЛЫХ.

Заметим, что в этом утверждении имеется последовательность «перемен кванторов», которая приводит к P-$SPACE$-полноте задачи БФК. Таким образом, не удивительно, что задача распознавания вида: «Задана начальная позиция некоторой игры, верно ли, что БЕЛЫЕ имеют форсированный выигрыш?» – оказывается P-$SPACE$-полной.

Первый результат такого типа был получен в работе [119]. Задача, которую обсуждали авторы, формулируется следующим образом.

ОБОБЩЕННЫЙ ГЕКС

УСЛОВИЕ. Задан неориентированный граф $G = (V, E)$ с выделенными вершинами $s, t \in V$.

ВОПРОС. Верно ли, что БЕЛЫЕ имеют форсированный выигрыш в следующей игре на G? ***Позицией*** является разбиение множества V на три таких подмножества $< V_1, V_2, V_3 >$, что $\{s, t\} \in V_3$. Исходная позиция имеет вид $< \varnothing, \varnothing, V >$, а все позиции вида $< V_1, V_2, \{s, t\} >$ заключительные. В позиции $< V_1, V_2, V_3 >$, где $V_3 \neq \{s, t\}$, если $|V_1| + |V_2|$ чётно, то ходят БЕЛЫЕ, в противном случае ходят ЧЁРНЫЕ. Ход БЕЛЫХ состоит в выборе вершины $u \in V_3 \backslash \{s, t\}$ и замены текущей позиции на позицию $< V_1 \cup \{u\}, V_2, V_3 \backslash \{u\} >$. ЧЁРНЫЕ ходят аналогично, удаляя вершину из V_3 и добавляя её к V_2. В заключительной позиции $< V_1, V_2, \{s, t\} >$ БЕЛЫЕ выигрывают тогда и только тогда, когда индуцированный множеством $V_1 \cup \{s, t\}$ подграф графа G содержит путь из s в t.

Основная идея этой игры состоит в том, что БЕЛЫЕ и ЧЁРНЫЕ по очереди выбирают вершины из множества $V \setminus \{s, t\}$. Цель БЕЛЫХ –построить путь из s в t, цель ЧЁРНЫХ – воспрепятствовать этому. В действительности игра ГЕКС имееr гораздо более специальную форму, чем игра ОБОБЩЕННЫЙ ГЕКС, и играется на графе регулярной структуры, похожей на шахматную доску. Рассматриваемая задача лежит в классе P-$SPACE$ потому, что в ней возможно только полиномиальное число шагов (если быть точным, то $|V| - 2$). Доказательство P-$SPACE$-полноты задачи получается с помощью сведения к ней задачи 3-ВЫП С КВАНТОРАМИ.

Известна также P-$SPACE$-полнота некоторых других игр. К их числу относятся: обобщение игры «география» – игра, в которой игроки по очереди приводят название стран, причем название новой страны начинается с той буквы, которой заканчивается название предыдущей [473], игра, получающаяся обобщением головоломки Братьев Паркер «Instant Insanity» [452], и обобщение игр в шахматы и Го на доски размера $n \times n$ (о двух последних играх, если не пользоваться специальными правилами, обеспечивающими полиномиальность общего числа ходов, установлена лишь P-$SPACE$-трудность) [127,345]. Другие примеры игр можно найти в списке приложения.

Кроме задач, связанных с играми, P-$SPACE$-полные задачи возникают в теории языков и автоматов, в программировании; при этом они часто оказываются ограниченными вариантами задач, труднорешаемость или даже неразрешимость которых известна. Одна из таких задач указана в работах [384, 9]. В этой задаче речь идёт о «регулярных выражениях». Она представляет интерес ещё и потому, что в ней нет такой явной перемены кванторов, как в рассматривавшихся до сих пор задачах. Определим вначале понятия «регулярное выражение» и «язык, им представляемый». Пусть Σ – конечный алфавит. *Регулярное выражение* над Σ и язык, представляемый им, определяются индуктивно:

(1) $\varnothing$ есть регулярное выражение, представляющее пустое множество слов;

(2) ε есть регулярное выражение, представляющее язык, состоящий из пустого слова ε;

(3) для всех $a \in \Sigma$ a есть регулярное выражение, представляющее язык $\{a\}$;

(4) если α и β – регулярные выражения, представляющие множества A и B, то

(i) $\alpha + \beta$ есть регулярное выражение, представляющее язык $A \cup B$,

(ii) $\alpha\beta$ есть регулярное выражение, представляющее язык $AB = \{xy : x \in A, y \in B\}$,

(iii) α^* есть регулярное выражение, представляющее язык $A^* = \bigcup_{i=0}^{\infty} A^i$, где $A^0 = \{\varepsilon\}$ и $A^{i+1} = A^i A$ при $i \geqslant 0$.

P-$SPACE$-полная задача относительно регулярных выражений формулируется следующим образом:

НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ

УСЛОВИЕ. Задано регулярное выражение a над конечным алфавитом Σ.

ВОПРОС. Верно ли, что множество, представляемое регулярным выражением α, отлично от Σ^*?

Эта задача остаётся P-$SPACE$-полной даже в том случае, если применять только фиксированный алфавит $\{0, 1\}$. Однако если расширить определение регулярного выражения, допуская использование вместо $\alpha \cdot \alpha$ сокращения $(\alpha)^2$ (и в частности, вместо $(((\alpha)^2)^2)^2$ сокращение $(\alpha)^8$), то возникающая в результате задача превратится в *труднорешаемую*, и для её решения, как доказано в [384], требуется экспоненциальная память. Доказательство P-$SPACE$-полноты этой задачи, приведённое в книге [9], представляет интерес потому, что в нём построена общая сводимость произвольного языка из класса P-$SPACE$, в которой регулярные выражения используются для кодирования *отвергающих* вычислений, а не принимающих. Таким образом, оно в некотором смысле двойственно к доказательству P-$SPACE$-полноты задачи БФК, а именно если некоторое слово не принадлежит множеству, представляемому построенным регулярным выражением, то это слово должно соответствовать принимающему вычислению.

В заключение рассмотрим задачу, из P-$SPACE$-полноты которой следует интересное утверждение о самом классе P-$SPACE$. Если задана ДМТ-программа, то её можно стандартным образом модифицировать так, что ни одно вычисление на входе длины n не потребует более $n+1$ ячеек ленты, а все вычисления, уже обладавшие этим свойством, останутся неизменными. Программа, возникающая в результате такой модификации, вообще говоря, может и не распознавать прежний язык, но для неё наверняка будет выполняться указанное ограничение по памяти. Если же программа уже удовлетворяла этому ограничению по памяти, то она *будет* распознавать тот же язык, что и раньше. Такие ДМТ-программы будут называться *линейно ограниченными.* Используя хорошо известное рассуждение об избыточном кодировании (см., например, [280]), можно доказать P-$SPACE$-полноту следующей задачи.

ПРИНЯТИЕ С ЛИНЕЙНОЙ ПАМЯТЬЮ

УСЛОВИЕ. Задана линейно ограниченная ДМТ-программа M и конечное слово x во входном алфавите соответствующей машины.

ВОПРОС. Верно ли, что M принимает x?

На первый взгляд этот результат может показаться удивительным. Он утверждает, что в некотором смысле задачи, разрешимые с полиномиальной памятью, не труднее задач, разрешимых с линейной памятью. Можно подумать, что он противоречит существованию языков, распознаваемых с полиномиальной памятью, но не распознаваемых никакой ДМТ-программой с линейно ограниченной памятью (в существовании последних мы убедимся в разд. 7.6). В действительности здесь нет противоречия, ибо из P-$SPACE$-полноты задачи ПРИНЯТИЕ С ЛИНЕЙНОЙ ПАМЯТЬЮ следует лишь, что линейная память содержится в P *тогда и только тогда, когда* P-$SPACE \subseteq P$, а не то, что все задачи класса P-$SPACE$-разрешимы с линейной памятью. На самом деле имеется более общий результат.

Теорема 7.11. *Если для некоторого $k \geqslant 1$ класс языков, распознаваемых ДМТ-программами, использующими память не более $O(n^k)$, содержится в P (соответственно в NP), то $P = P$-$SPACE$ (соответственно NP).*

(Аналогичный результат можно получить для вычислений с ограниченным временем: если для некоторого $k \geqslant 1$ класс языков, недетерминированно распознаваемых со временем $O(n^k)$, содержится в P, то $P = NP$.)

Определив понятие P-$SPACE$-полноты, естественно пойти и дальше, определив класс NP-$SPACE$, состоящий из языков, распознаваемых программами для *недетерминированных* машин Тьюринга, с полиномиальной оценкой по памяти. Однако, прежде чем мы сможем говорить об ограниченности недетерминированного вычисления по памяти, необходимо уточнить понятие памяти, требующейся для «угадывающего модуля НДМТ». Для многих вычислений в действительности нет необходимости помнить все символы слова-догадки после того, как они один раз уже просмотрены. В отличие от нашей модели НДМТ, у которой в память, необходимую для вычисления, включается память, необходимая для выписывания в начале вычисления всего слова-догадки, стандартные модели НДМТ, используемые для измерения памяти, обычно снабжаются дополнительным устройством, вообще не требующим памяти, из которого программа может в любой момент времени запрашивать «следующий» символ слова-догадки. Программа записывает символ на ленте и тем самым расходует «память» для этого символа только в том случае, если его нужно запомнить для некоторой операции, выполняемой в будущем.

Нетрудно показать, что, хотя память, требуемая изменённой моделью НДМТ, измеряется иначе, эта изменённая модель эквивалентна относительно полиномиальной временно́й сложности НДМТ из гл. 2, и поэтому при её использовании для определения класса $NP\text{-}SPACE$ сохранятся наши прежние результаты о недетерминированных вычислениях за полиномиальное время. Определив теперь класс $NP\text{-}SPACE$ как класс языков, распознаваемых программами для этой модифицированной модели НДМТ, требующими в своих принимающих вычислениях полиномиальной памяти, можно задаться вопросом о взаимосвязи классов $P\text{-}SPACE$ и $NP\text{-}SPACE$. Утверждение, что $P\text{-}SPACE = NP\text{-}SPACE$, может показаться неожиданным. Оно вытекает из следующего результата Сэвича [470].

Теорема 7.12. *Если язык L может быть распознан НДМТ-программой с памятью $T(n)$, где $T(n) \geqslant \log n$ для всех $n \geqslant 1$, то L можно распознать также ДМТ-программой с памятью $T^2(n)$.*

Из теоремы 7.12 следует, что $P\text{-}SPACE$-полнота – самый сильный из известных в настоящее время результатов о полноте и близок к результатам, из которых следует труднорешаемость. Конечно, ещё остаётся возможность доказывать, что задача $P\text{-}SPACE$-трудна (т. е. доказывать, что некоторая $P\text{-}SPACE$-полная задача сводима к ней по Тьюрингу, оставляя открытым вопрос о принадлежности этой задачи классу $P\text{-}SPACE$), но исключив эту возможность, мы попадаем в область труднорешаемых задач, которые будут обсуждаться в разд. 7.6.

Настоящий раздел мы закончим упоминанием об одной открытой проблеме, связанной с теоремой 7.12, которая до возникновения проблемы о взаимоотношении классов P и NP была, видимо, наиболее известной в теории сложности. Эта проблема известна как «LBA-проблема». Назовём ДМТ-программу, которая требует памяти, не превосходящей величины $n+1$, *детерминированным линейно ограниченным автоматом* (DLB-автоматом), а НДМТ-программу с тем же ограничением по памяти – *недетерминированным линейно ограниченным автоматом* (NLB-автоматом).

Множество языков, распознаваемых NLB-автоматами, совпадаёт с множеством «контекстно-зависимых языков» (класс языков, имеющих специальное грамматическое определение, см., например, [215]). По теореме 7.12 любой такой язык можно распознать ДМТ-программой с оценкой по памяти $(n+1)^2$. Вопрос заключается в следующем: «Можно ли все такие языки распознать DLB-автоматами?» (Или ещё более сильное утверждение: «Можно ли в теореме 7.12 отбросить показатель 2?») Этот вопрос относится к тому же предмету, что и вопрос о взаимоотношении классов P и NP, т. е. к соотношению между детерминированностью и недетерминированностью, правда, в более специальной форме. Он остаётся открытым и до настоящего времени, хотя был поставлен в середине 60-х годов. Непосредственное обсуждение этого вопроса, следствий и некоторых связанных с ним предположений (включая аналог вопроса о взаимоотношении классов *NP* и *co–NP*) можно найти в работе [202].

7.5. Логарифмическая память

В предыдущем разделе были рассмотрены взаимоотношения между классами временнóй сложности *P* и *NP*, с одной стороны, и классом $P\text{-}SPACE$-функций, вычислимых с полиномиально ограниченной памятью, – с другой. В настоящем разделе будет продолжено обсуждение связей между сложностью по времени и сложностью по памяти, но на этот раз мы обратимся к классу DLOG-SPACE, лежащему в *P* и *NP* и состоящему из языков, распознаваемых ДМТ-программами, которые требуют памяти, ограниченной логарифмом от длины входа.

На первый взгляд понятие логарифмической памяти может показаться бессмысленным, поскольку входное слово длины n само занимает n ячеек, так что любая ДМТ-программа, просматривающая слово от начала до конца, должна потребовать, самое меньшее, линейной памяти. Однако на практике часто оказывается полезно проводить различие между памятью, занимаемой входным словом, и дополнительной памятью, используемой для выполнения вычисления. Например, если при слиянии двух отсортированных списков, записанных на магнитных лентах, результат также пишется на магнитной ленте, то необходимая для слияния оперативная память в каждый момент времени включает только две ячейки, в которых запоминается по одному элементу из каждого списка. В действительности здесь чрезвычайно важно то обстоятельство, что нет необходимости в оперативной памяти, сравнимой по величине с длинами входных слов.

Чтобы изучить вопрос о том, как можно уменьшить потребность в памяти, так чтобы требовалась меньшая, чем линейная, память, изменим нашу стандартную модель ДМТ, как показано на рис. 7.6. Здесь добавлены входная лента с читающей головкой, которая может передвигаться в двух направлениях, и выходная лента с пишущей головкой. Входное слово, записанное на входной ленте, начинается в ячейке с номером 1, причем ячейка с номером 0 пуста, что позволяет распознавать левый конец слова. Само вычисление, как обычно, выполняется на бесконечной в обе стороны рабочей ленте, с использованием читающей/пишущей головки. Память, которая требуется при вычислении на этой модели, есть просто число ячеек, просмотренных читающей/пишущей головкой. Следует отметить, что классы *P*, *NP* и $P\text{-}SPACE$ при таком изменении модели вычислительного устройства не изменяются, зато появляется возможность затрачивать меньшее, чем линейное, количество памяти.

Исследования по использованию памяти, меньшей, чем линейная, сосредоточились на классе DLOG-SPACE языков, распознаваемых ДМТ-програ-

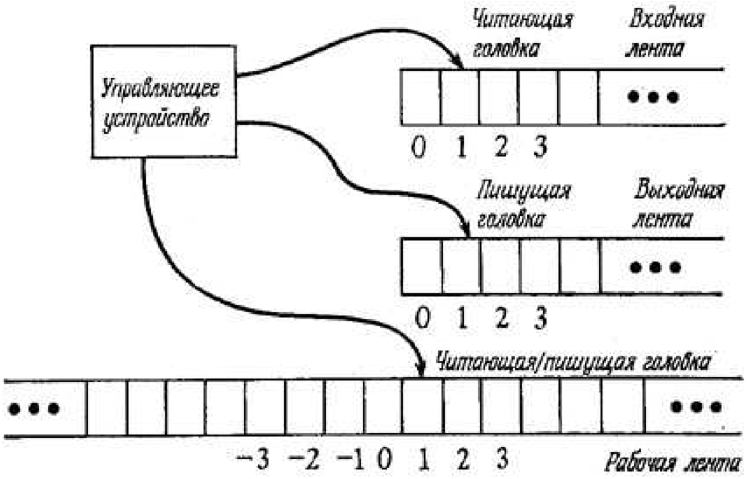


Рис. 7..6: *Схематическое изображение модели ДМТ, которая требует памяти меньшей, чем линейная.*

ммами, требующими памяти не более чем $\lceil \log_2(n+1) \rceil$, где n – длина входного слова. С помощью стандартной теоремы об «ускорении» (см., например, [215]) легко показать, что этот класс совпадает с классом языков, распознаваемых с памятью $C\lceil \log_2(n+1)\rceil$, для любой константы $C > 0$ и не так уж беден, как может показаться на первый взгляд. Более того, нетрудно видеть что DLOG-SPACE $\subseteq P$. Первый возникающий вопрос, конечно, состоит в следующем: «Верно ли, что DLOG-SPACE = P?»

Хотя имеется много нетривиальных задач, которые могут быть решены с логарифмической памятью (см., например, [355, 368]), по-видимому, DLOG-SPACE $\neq$ P. Многие задачи из класса P, кажется, требуют для своего решения значительно большей памяти, чем логарифмическая. Рассмотрим, например, задачу РАЗБИЕНИЕ. В гл. 4 было показано, что эта задача может быть решена за полиномиальное время, если размеры всех предметов не превосходят, скажем, квадрата их количества. Однако в этом случае приведённый там алгоритм потребует памяти, пропорциональной кубу числа предметов, что превосходит квадрат длины входа, и не видно способа существенно уменьшить количество необходимой памяти.

В работах [94, 95] приведены примеры задач класса P, для которых память, требуемая любым алгоритмом из весьма широкого класса, пропорциональна по меньшей мере $n^{1/2}$ или $n^{1/4}$. Более того, известно, что не может быть *одновременно* $P =$ DLOG-SPACE и $P = P\text{-}SPACE$ (это следует из теоремы 7.15, приведённой в следующем разделе). Таким образом, весьма вероятно, что некоторые задачи класса P требуют памяти, большей, чем логарифмическая, и мы вновь воспользуемся результатами о полноте, чтобы указать наиболее вероятных кандидатов.

Для нового понятия полноты нам нужен новый вид сводимости. Это объясняется тем, что полиномиальная сводимость не может различать задачи класса P. Пусть L_1 и L_2 – языки в алфавитах Σ_1 и Σ_2 соответственно. Будем говорить, что функция $f : \Sigma_1^* \to \Sigma_2^*$ осуществляет log-space-*сводимость* языка L_1 к языку L_2, если

(i) f может быть вычислена ДТМ-программой, использующей на входах длины n память, не превосходящую величины $\lceil \log_2(n+1) \rceil$, и

(ii) $x \in L_1$ тогда и только тогда, когда $f(x) \in L_2$.

Если имеется log-space-сводимость языка L_1 к языку L_2, то будем писать $L_1 \propto_{LOG} L_2$ и говорить, что при этом язык L_1 log-space сводится к языку L_2. Отметим, что так как все функции, вычислимые с логарифмически ограниченной памятью, вычислимы также и за полиномиальное время, то сводимость $L_1 \propto_{LOG} L_2$ влечет сводимость $L_1 \propto L_2$. По аналогии со сводимостью $\propto$ можно доказать следующие свойства сводимости $\propto_{LOG}$ (доказательства, однако, не тривиальны, см. [503]).

Теорема 7.13. *Предположим, что* $L_1 \propto_{LOG} L_2$ *и* $L_2 \propto_{LOG} L_3$. *Тогда*
(1) $L \propto_{LOG} L_3$;
(2) $L_2 \in DLOG\text{-}SPACE \Rightarrow L_1 \in DLOG\text{-}SPACE$.

Будем говорить, что язык $L \in P$ *log-space-полон* в классе P, если $L' \propto_{L0G} L$ для всех $L' \in P$, Таким образом, мы имеем желаемое свойство: если язык L log-space-полон в классе P, то $L \in DLOG\text{-}SPACE$ тогда и только тогда, когда $DLOG\text{-}SPACE = P$. Более общий результат, доказанный в работе [270], состоит в следующей теореме. (Здесь класс $DLOG^k\text{-}SPACE(k > 1)$ определяется аналогично классу DLOG-SPACE, с заменой границы $\lceil \log_2(n+1) \rceil$ границей $\lceil \log_2(n+1) \rceil^k$.)

Теорема 7.14. *Если язык* L *log-space-полон в классе* P *и если* $L \in DLOG\text{-}SPACE$, *то* $P \subseteq DLOG^k\text{-}SPACE$.

Первая log-space-полная задача была явно сформулирована Куком [94].

ДОСТИЖИМОСТЬ СИСТЕМОЙ ПУТЕЙ

УСЛОВИЕ. Заданы конечное множество «узлов» X, отношение $R \subseteq X \times X \times X$ и два подмножества $S, T \subseteq X$, S – множество «источников», T – множество «конечных узлов».

ВОПРОС. Существует ли «достижимый» конечный узел? (Узел $x \in X$ называется достижимым, если $x \in S$ или существуют достижимые узлы y, z, такие, что $< x, y, z > \in R$.)

Эта задача лежит в классе P, поскольку множество всех достижимых узлов тривиальным образом можно построить за полиномиальное время. Доказательство log-space-полноты этой задачи в классе P состоит, как обычно, в общем сведении с логарифмической памятью произвольной задачи класса P к рассматриваемой задаче. В работе [271] было указано аналогичное доказательство log-space-полноты в классе P следующей задачи.

ЕДИНИЧНАЯ РЕЗОЛЮЦИЯ

УСЛОВИЕ. Задано множество C дизъюнкций на множестве переменных $X = \{x_1, x_2, ..., x_n\}$ (т. е. условие такое же, как в задаче ВЫПОЛНИМОСТЬ).

ВОПРОС. Можно ли с помощью «единичной резолюции» вывести пустую дизъюнкцию (указывающую на противоречие) из множества C? Иными словами, существует ли последовательность дизъюнкций $c_1, c_2, ..., c_m$, такая, что c_m – пустая дизъюнкция, а c_i – либо дизъюнкция из множества C, либо существуют ранее выведенные дизъюнкции c_k и c_l $(k, l < i)$, такие, что $c_k = \{x_j\}$, $c_l = \{\bar{x}_j\} \cup c$; (или $c_k\{\bar{x}_j\}$, $c_l = x_j \cup c_i$) для некоторой переменной $x_j \in X$?

Другие log-space-полные задачи в классе P можно найти в работах [271, 313, 136, 527, 177, 300].

Использование сводимости с логарифмической памятью не ограничивается доказательством результатов о полноте в классе P. Например, задача ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ, как доказано в работе [103], является «log-space-трудной для класса P». Иными словами, хотя и не известно, что наличие полиномиального алгоритма решения этой задачи привело бы к равенству $P = NP$, однако существование log-space-алгоритма решения этой задачи привело бы к равенству DLOG-SPACE = P[4]. Более того, как было замечено в работах [269] и [384], большинство (если не все) сводимостей, используемых для доказательства результатов об NP-полноте, могут быть выполнены с логарифмической памятью (основная память, необходимая в таких сводимостях, затрачивается на вычисления значения $p(n)$ некоторого полинома P от длины записи n исходной информации, что может быть сделано с логарифмической памятью, если алфавит букв, которые пишутся на ленте, имеет достаточное количество символов). Таким образом, класс языков, «log-space-полных в классе NP», составляет по крайней мере большой подкласс класса NP-полных задач, и если хоть одна задача этого класса принадлежит классу DLOG^k-SPACE для некоторого $k \geqslant 1$, то все задачи класса NP принадлежат классу DLOG-SPACE.

Подобным же образом было замечено, что сводимости, используемые для доказательства результатов о P-$SPACE$-полноте, также являются log-space-сводимостями (в действительности в некоторых работах P-$SPACE$-полнота определяется непосредственно в терминах log-space-сводимости, а не в терминах полиномиальной сводимости). Однако это замечание для класса P-$SPACE$ не имеет таких следствий, как для класса NP, ибо нам уже известно, что для всех k $P\text{-}SPACE \neq DLOG^k\text{-}SPACE$, и даже более того,

$$P - SPACE \neq POLYLOG - SPACE = \bigcup_{k=1}^{\infty} DLOG - SPACE.$$

[4]Алгоритм Хачияна [33*] для решения задачи LP требует полиномиальной памяти, и, таким образом, вопрос о совпадении классов DLOG-SPACE и P остаётся открытым. – *Прим. перев.*

Тем не менее, если язык L log-space-полон в классе $P\text{-}SPACE$, то можно заключить, что существует константа $r > 0$, такая, что для ДТМ-программы, распознающей L, необходима память, пропорциональная n^r для бесконечного числа входов (см., например, [500]). Этот вывод сделать нельзя, если известна лишь $P\text{-}SPACE$-полнота языка L в смысле определения из разд. 7.4.

Наконец, понятием сводимости с логарифмической памятью можно воспользоваться для того, чтобы сформулировать ещё один вопрос о соотношении между детерминизмом и недетерминизмом, на этот раз уже на логарифмическом уровне. Пусть NLOG-SPACE – это класс всех языков, распознаваемых недетерминированными машинами Тьюринга с памятью, ограниченной $\lceil \log_2(n+1) \rceil$ (определёнными таким образом, что память, необходимая для записи входа и догадки, не учитывается). Вопрос состоит в следующем: «Верно ли, что DLOG-SPACE = NLOG-SPACE?» Гипотеза заключается в том, что эти классы не совпадают, а языки, log-space-полные в классе NLOG-SPACE, можно считать кандидатами на языки класса NLOG-SPACE, не принадлежащие классу DLOG-SPACE. Примеры и другие детали можно найти в работах [471, 270, 507, 273].

Мы закончим настоящий раздел кратким рассмотрением отношений включения между обсуждавшимися выше классами языков. Уже отмечалось, что DLOG-SPACE $\subseteq$ P. Менее очевидный результат [92] состоит в том, что NLOG-SPACE $\subseteq$ P. Однако совсем не ясно, верно ли включение POLYLOG-SPACE $\subseteq$ P или даже включение DLOG^2-SPACE $\subseteq$ NP. Хотя пока не доказано, что эти включения не имеют места, в работе [48] показано, что $P \neq$POLYLOG-SPACE и $NP \neq$POLYLOG-SPACE. Результаты, полученные в этой работе, особенно интересны потому, что из них не следует, что один класс содержится в другом или что два класса несравнимы относительно включения, хотя в них утверждается, что классы не могут совпадать. Другие результаты относительно взаимосвязей между различными классами сложности по времени и памяти обсуждаются в работах [46, 47, 384].

7.6. Доказательства труднорешаемости и взаимоотношения между P и NP

В качестве подтверждения гипотезы $P \neq NP$ мы упоминали тот факт, что было бесплодно потрачено много усилий на отыскание полиномиальных по времени алгоритмов решения NP-полных задач. Видимо, точнее было бы сказать, что это является аргументом в пользу трудности доказательства того, что $P = NP$. В настоящем разделе будут приведены аргументы в пользу того, что доказательство различия классов P и NP также не просто.

Начнем с рассмотрения некоторых частичных, но доказанных результатов. Вместо того чтобы браться за решение вопроса в полной общности, доказывая, что все алгоритмы решения некоторой NP-полной задачи работают в течение экспоненциального времени, некоторые исследователи с целью доказать отсутствие полиномиальных алгоритмов решения этой задачи ограничились рассмотрением определённых ограниченных классов алгоритмов. Например, Галил [137] (продолжая работу Цейтина [515]) показал, что алгоритмы решения задачи ВЫПОЛНИМОСТЬ, основанные на технике резолюций, должны иметь экспоненциальную времену́ю сложность. Для задачи НЕЗАВИСИМОЕ МНОЖЕСТВО Хватал [76] рассмотрел весьма широкий класс алгоритмов, включающий алгоритм Тарьяна и Трояновского [513], упомянутый в гл. 6, и доказал, что любой такой алгоритм работает экспоненциальное время (точнее, для каждого такого алгоритма существует константа $c > 1$, такая, что для «почти всех» графов на n вершинах время работы алгоритма больше c^n). Мак-Диармид [378] получил аналогичные результаты об алгоритмах для задачи РАСКРАЩИВАЕМОСТЬ ГРАФА В K ЦВЕТОВ.

Результаты этого типа имеют практическое значение потому, что они позволяют исключить некоторые соблазнительные на первый взгляд подходы к построению полиномиальных алгоритмов. Однако, если даже дальнейшие исследования позволят расширить класс непригодных подходов, кажется маловероятным, что, действуя таким «кусочным» методом, удастся исключить все возможные подходы к построению алгоритмов. Если мы хотим доказать, что $P \neq NP$, то нам необходимо получить нижнюю оценку временно́й сложности, которая выполнялась бы для всех алгоритмов решения задачи. К сожалению, даже самые сильные нижние оценки такого типа, доказанные к настоящему времени, оставляют желать много лучшего.

Один из опробованных подходов к получению нижних оценок заключается в том, чтобы оценить число элементов в логических схемах, вычисляющих конечные функции. Такая функция получается, если ограничиться индивидуальными задачами некоторой NP-полной задачи, имеющими фиксированный размер n. Если удастся показать, что число необходимых элементов зависит от n экспоненциально, то отсюда будет следовать труднорешаемость задачи. Самая сильная нижняя оценка, полученная в рамках этой модели, потребовавшая большой изобретательности и находчивости, составляет всего лишь $2{,}5n$ (см. [422]). В действительности доказательство нижних оценок для общей модели процесса вычислений, даже чуть-чуть более сильных, чем линейные (таких, например, как в работе [428]), в настоящее время оказывается очень трудным. Доказательство такими методами экспоненциальных нижних оценок представляется делом далекого будущего.

Тем не менее, как упомянуто в гл. 1, труднорешаемость некоторых задач, хотя и не принадлежащих классу NP, доказана. Можно ли применить эти методы для доказательства того, что $P \neq NP$? После того как мы ознакомимся с этими методами, будет видно, что это также представляется маловероятным.

Все доказательства труднорешаемости, полученные до настоящего времени, базируются, по существу, на двух основных результатах теории сложности. Будем называть функцию $F(n)$ *конструируемой по памяти*, если существует ДМТ-программа, которая, начав работать на слове, состоящем из n единиц, останавливается после того, как читающая/пишущая головка просмотрит ровно $F(n)$ ячеек ленты (предполагается, что у ДМТ есть отдельная входная лента). Функция $F(n)$ называется *конструируемой по времени*, если существует ДТМ-программа, которая, начав работать на слове, состоящем из n единиц, останавливается, выполнив ровно $F(n)$ шагов[5]. Наиболее часто встречающиеся функции, такие, как $n^k, 2^{cn}, k^{cn}, n!$ и т.д., при фиксированных целых положительных константах c и k конструируемы как по памяти, так и по времени. Функции $\lceil \log_2 n \rceil^k$ при всех положительных целых k конструируемы по памяти, но по очевидным причинам неконструируемы по времени. Два важных результата формулируются следующим образом:

Теорема 7.15. *Пусть $F_1(n)$ и $F_2(n)$ – функции, конструируемые по памяти, $F_2(n) \geqslant \log_2(n)$ для всех $n \geqslant 1$. Если*

$$\lim_{n\to\infty} inf \frac{F_1(n)}{F_2(n)} = 0,$$

то существует язык L, распознаваемый некоторой ДМТ-программой со сложностью по памяти, ограниченной $F_2(n)$, но не распознаваемый никакой ДМТ-программой, имеющей сложность по памяти, ограниченную $F_1(n)$.

Теорема 7.16. *Если $F_1(n)$ и $F_2(n)$ – функции, конструируемые по времени, и*

$$\lim_{n\to\infty} inf \frac{F_1(n)}{F_2(n)} = 0,$$

то существует язык L, распознаваемый ДМТ-программой, временная сложность которой ограничена функцией $F_2(n) \cdot \log_2 F_2(n)$, но не распознаваемый никакой ДМТ-программой с временно́й сложностью, ограниченной функцией $F_1(n)$.

Эти результаты были впервые получены в работах [203] и [204] (см. также работу [215]). Основные выводы, которые можно сделать на основе этих результатов, состоят в том, что некоторые классы языков должны содержать языки, распознавание которых – труднорешаемая задача. Например, класс EXP-TIME (состоящий из всех языков, распознаваемых с временно́й сложностью, ограниченной 2^{pn}, где p – полином, n – длина записи входа) для всех $k \geqslant 1$ должен содержать языки, сложность которых по крайней мере 2^{n^k}. Следует, однако, подчеркнуть, что такие языки строятся с помощью «диагонального» метода. Поэтому нельзя считать, что они соответствуют «естественным» задачам, и с ними нельзя работать непосредственно. По этой причине такие языки используются для доказательства труднорешаемости естественных задач косвенным путем на основе понятий «полноты»

[5] В машине Тьюринга с несколькими головками на каждом шаге работы обычно допускается одновременный сдвиг всех головок, – *Прим. перев.*

и «трудности» в классе задач. Напомним, что язык L называется полным в классе языков C (относительно полиномиальной сводимости), если $L \in C$ и $L' \propto L$ для всех $L' \in C$ (L называется C-трудным, если выполняется второе свойство, но L не обязательно лежит в C). Теперь можно сформулировать общий аналог леммы 2.1.

Лемма 7.1. *Если язык L полон в классе C (или C-труден) и C содержит труднорешаемую задачу, то L соответствует труднорешаемой задаче.*

Доказательства труднорешаемости естественных задач базируются на этой лемме при выборе в качестве C различных конкретных классов языков. В работе [384] доказано, что задача неэквивалентности, регулярных выражений с операцией «возведения в квадрат» полна в классе EXP-SPACE, в котором содержится класс EXP-TIME и, следовательно, содержатся труднорешаемые задачи. В работе [122] доказано, что задача распознавания истинных утверждений в арифметике Прессбургера C-трудна, где C = NEXP-TIME (недетерминированный аналог класса EXP-TIME). В работах [68, 502] доказана полнота в классе EXP-TIME различных комбинаторных игр. В работе [257] также доказана полнота в классе EXP-TIME «задачи циркулярности в атрибутивных грамматиках».

В действительности многие из этих доказательств дают более сильные нижние границы сложности, чем «простая» труднорешаемость, а в некоторых доказательствах получены нижние оценки сложности по памяти. Более того, если класс C содержит достаточно сложные задачи, можно доказать астрономические нижние оценки. Такие оценки получены в работе [381] для «слабой монадной теории следования второго порядка», а именно здесь доказано, что любая ДМТ-программа, распознающая истинные утверждения этой теории, должна иметь временýю сложность, большую, чем

$$2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

для любого числа «этажей» в формуле. Мы не будем вдаваться в детали методов доказательства, так как будет видно, что этим подходом, по-видимому, нельзя воспользоваться для доказательства того, что класс NP содержит труднорешаемые задачи.

Для того чтобы с помощью леммы 7.1 доказать утверждение $P \neq NP$, необходимо уметь доказывать, что некоторый язык $L \in NP$ полон в некотором классе C, содержащем труднорешаемые задачи. Однако легко понять, что подобный результат имел бы и другие, кроме доказательства того, что $P \neq NP$, следствия, которые можно получить на основе следующей простой леммы.

Лемма 7.2. *Если язык L принадлежит классу NP и L полон (относительно полиномиальной сводимости) в классе C, то C содержится в NP.*

Поэтому, если $L \in NP$, то не представляется возможным доказать, что L полон в классах EXP-SPACE или NEXP-SPACE. В самом деле, по теореме 7.15 невозможно включение EXP-SPACE$\subseteq NP$, ибо класс EXP-SPACE строго содержит класс P-$SPACE$, который в свою очередь включает класс NP. Из недетерминированного варианта теоремы 7.16, вытекающего из результатов Кука [93] (см. также [475]), следует, что невозможно включение NEXP-TIME $\subseteq NP$. Не следует также надеяться, что удастся доказать неравенство $P \neq NP$ с использованием понятия полноты в классе EXP-TIME, потому что если бы некоторая задача класса NP была бы полна в классе EXP-TIME, то имело бы место маловероятное (хотя пока и не опровергнутое) включение EXP-TIME $\subseteq NP$. Отметим также, что лемма 7.2 остаётся верной и в случае, если в её условии заменить класс NP классом P-$SPACE$, и аналогичные замечания можно сделать относительно попыток доказательства этим методом неравенства $P \neq P$-$SPACE$.

Таким образом, лемма 7.1 оказывается слишком неконкретной, чтобы её можно было использовать для доказательства труднорешаемости на «нижнем» уровне иерархии, соответствующем классу NP. Следовательно, вместо того чтобы пытаться доказать неравенство $P \neq NP$ посредством (хотя бы даже косвенного) сведения труднорешаемой задачи к задаче класса NP, можно было бы с помощью метода диагонализации, используемого в доказательствах теорем 7.15 и 7.16, попытаться построить труднорешаемую задачу в классе NP непосредственно. К сожалению, такого построения не известно, и стандартные методы построения подобных задач, видимо, неприменимы к доказательству утверждения $P \neq NP$. Причины этого можно пояснить с помощью понятия «релятивизация».

В работе [25] замечено, что обычные методы диагонализации (а также метод «моделирования», подобный методу доказательства теоремы 2.1) применимы и в случае «релятивизации» относительно любого языка L_0, т. е. и в том случае, если все программы для машин Тьюринга превращаются в программы для машин с оракулом L_0. Например, релятивизованная версия теоремы 7.16 формулируется так:

`Теорема 7.17.` *(релятивизованная). Если функции $F_1(n)$ и $F_2(n)$ конструируемы по времени и*

$$\lim_{n\to\infty} inf \frac{F_1(n)}{F_2(n)} = 0,$$

то существует язык L, распознаваемый некоторой OMT-программой с оракулом L_0 и временно́й сложностью, ограниченной $F_2(n) - \log_2 F_2(n)$, но не распознаваемый никакой OMT-программой с оракулом L_0 и временно́й сложностью, ограниченной $F_1(n)$.

Эта релятивизованная теорема верна для любого языка L_0. То же самое верно и для релятивизованной версии теоремы 7.15. Таким образом, если мы умеем доказывать, что $P \neq NP$ с помощью метода диагонализации, то следует ожидать, что то же самое доказательство верно и для утверждения

$P^L \neq NP^L$ (где P^L и NP^L – варианты классов P и NP, определяемые с помощью оракульных машин Тьюринга с оракулом L). С другой стороны, если бы методами моделирования удалось доказать, что $P = NP$, то из этого следовало бы, что для любого языка L верно утверждение $P^L = NP^L$, ибо методы моделирования легко релятивизовать. Поскольку все основные методы доказательства результатов подобного сорта относятся к двум упомянутым выше типам, то следующие теоремы [25] выглядят несколько обескураживающими.

Теорема 7.18. *Существуют рекурсивные языки A и B, такие, что*
(1) $P^A = NP^A$,
(2) $P^B \neq NP^B$.

Эта теорема даёт яркое подтверждение тому, что современные методы доказательства не достаточны ни для доказательства того, что $P \neq NP$, ни того, что $P = NP$. Результаты, взятые из той же работы, приводимые ниже, указывают также на то, что эти методы бесполезны и для решения других открытых вопросов.

Теорема 7.19. *Существуют рекурсивные языки D, E, F и G, такие, что:*
(1) $NP^D \neq co{-}NP^D$,
(2) $NP^E = co{-}NP^E$, но $P^E \neq NP^E$,
(3) $P^F \neq NP^F$, но $P^F = NP^F \cap co{-}NP^F$,
(4) $P^G \neq NP^G \cap co{-}NP^G$, но $NP^G \neq co{-}NP^G$.

Более того, пользуясь этими результатами а также результатами работы [26]), можно сформулировать аналогичные утверждения относительно возможности доказательства современными средствами родственных утверждений, касающихся полиномиальной иерархии и вопроса о взаимоотношении классов P и $P\text{-}SPACE$. (В вопросе о памяти имеется, однако, неясность, связанная с выбором соответствующей модели для получения результатов о релятивизации, см. [315] и [369].)

Для решения вопросов о взаимоотношениях между классами P и NP, а также других, упомянутых выше классов языков, вероятно, требуются совершенно иные методы доказательства. В том случае, конечно, если вообще подобные вопросы можно решить. Хартманис и Хопкрофт [201] применили методы релятивизации, близкие к рассмотренным выше, чтобы пояснить возможность того, что, по крайней мере в принципе, вопросы такого типа могут быть независимыми от теории множеств и, следовательно, неразрешимыми при использовании стандартных логических средств. Хотя авторы этой книги, как и многие другие специалисты в данной области, не так пессимистически смотрят на разрешимость вопроса о взаимоотношении классов P и NP, никто не надеется, что этот вопрос будет решен в скором времени.

Приложение А

Список *NP*-полных задач

В настоящее время при изучении задачи с вычислительной точки зрения стало более или менее общепринятым формулировать вопрос об её *NP*-полноте (или *NP*-трудности). Вследствие этого в научной литературе появилось много результатов об *NP*-полноте, а ещё большее число полученных результатов не опубликовано. В настоящем приложении собрано много подобных результатов. Они представлены в виде списка задач, *NP*-полнота или *NP*-трудностъ которых установлена.

В списке содержится более 300 основных элементов, а комментарии к ним в несколько раз увеличивают общее число формулируемых результатов. Каждый элемент списка состоит из четырех частей: (1) названия задачи, (2) её описания, (3) указания источника, из которого взят соответствующий результат, 4) дополнительных комментариев.

Название задачи служит удобным сокращением, которое используется для ссылок в тексте книги. Из соображений согласованности сокращений в некоторых случаях названия задач несколько отличаются от традиционных. Звездочка в скобках (*), которая следует за названием некоторых задач, указывает на тот факт, что принадлежность задачи классу *NP* не установлена, поэтому формулируемый результат следует понимать как *NP*-трудность задачи, а не как её *NP*-полноту.

Описание задач дано в стандартной форме, которой мы придерживались на протяжении всей книги, при этом формулируются *условие* и *вопрос* задачи. (В качестве основных элементов списка мы взяли лишь задачи распознавания свойств, хотя в комментариях иногда упоминаются и другие типы задач.) В большинстве случаев нам удалось сформулировать задачу на языке теории множеств и тем самым дать полное её описание. Однако стремление сделать описание сжатым часто приводит к тому, что задачи становятся интуитивно менее ясными, и в этих случаях читатель может обратиться к литературным источникам. Некоторые общеизвестные термины не определяются, а громоздкие определения, общие для целой группы задач, приводятся только один раз, в первой задаче, для формулировки которой они потребовались. Иногда полная формулировка задачи весьма сложна или требует обширного подготовительного материала. Поэтому мы даем только набросок формулировки, пользуясь отдельными «ключевыми» словами. При этом явно указаны литературные источники, в которых можно найти более точную формулировку.

В части, относящейся к *источнику* задачи, указывается как библиографический источник, откуда взят основной результат, так и соответствующая «известная NP-полная (или NP-трудная) задача», которой можно воспользоваться для доказательства результата. При этом мы выбирали задачи так, чтобы доказательство сводимости было, с нашей точки зрения, наиболее естественным, и часто исходили не из доказательства, приводимого в литературном источнике, а из доказательств, упомянутых в комментариях, или из наших собственных, неопубликованных доказательств. Некоторые результаты представляются настолько простыми, что некому за них воздать почести и некого упрекнуть.

В разделе *комментарий* приводятся различные варианты основного результата. В комментариях обычно цитируются различные результаты, касающиеся разрешимости за полиномиальное время и NP-трудности некоторых частных случаев и вариантов основной задачи. Таким образом, список может служить полезным источником информации о том, какие задачи можно, а какие нельзя (при условии $P \neq NP$) решить за полиномиальное время. Однако мы не пытались цитировать «наилучший» из известных полиномиальных алгоритмов решения задачи, а лишь принципиально обосновывали утверждение о разрешимости задачи за полиномиальное время. В тех случаях когда это необходимо, упоминаются также результаты о труднорешаемости и неразрешимости различных обобщений и вариантов основной задачи. Для задач, принадлежность которых классу NP не известна, приводится дополнительная информация о сложности задачи, в частности сведения о $P\text{-}SPACE$-полноте задачи или КР-полноте соответствующей задачи перечисления.

Задачи в списке разбиты на тематические группы. Они подразделяются на двенадцать категорий:

А1 Теория графов
А2 Построение сетей
А3 Множества и разбиения
А4 Хранение и поиск данных
А5 Теория расписаний
А6 Математическое программирование
А7 Алгебра и теория чисел
А8 Игры и головоломки
А9 Логика
А10 Автоматы и языки
А11 Оптимизация программ
А12 Разное

Задачи каждой категории нумеруются индивидуально, причем перед номером задачи пишутся первые буквы из названия соответствующего раздела (например, ТГ 7 соответствует седьмой задаче из раздела теории графов). В тринадцатой «категории» перечислено небольшое число открытых задач, выбранных на основе нашего представления об их важности и сложности. Другие открытые задачи рассеяны в комментариях к задачам основного списка, а очевидные пробелы в списке могут подсказать ещё большее число таких задач.

Многие задачи из списка вполне можно было бы отнести к нескольким различным категориям, так что мы советуем читателю внимательно просмотреть список, чтобы получить лучшее представление о том, в каком разделе следует искать ту или иную задачу. Следует также помнить, что комментарии к задачам следует просматривать особенно тщательно, потому что многие результаты упоминаются только в них. В указателе мы попытались привести полезную «навигационную» информацию.

Наконец, несколько слов о стандартах и методике, которой мы придерживались при составлении списка. Как и все списки подобного рода, список, приводимый ниже, в какой-то степени отражает вкусы авторов, но мы пытались в разумных пределах сделать его полным. При составлении списка мы не ограничились изучением публикаций, а пытались расширить нашу коллекцию сведений об *NP*-полноте из личных бесед, давали объявления в соответствующей периодике и на тематических конференциях с просьбой присылать нам неопубликованные результаты. При этом, однако, в список не вошел ни один результат, для которого не было бы приведено соответствующего подтверждения. Неопубликованные результаты включены в список или на основе рукописных заметок автора, или на основе наброска доказательства, который проверили либо мы сами, либо наши «доверенные эксперты». В отдельных случаях мы проверяли результат, пытаясь получить для него новое, собственное доказательство (зачастую это оказывается проще, чем проверять чужое доказательство).

Некоторые результаты мы не включили в список (хотя нас все-таки можно упрекнуть в том, что мы включили в список некоторые весьма специфические задачи). В большинстве случаев мы избегали задач со слишком сложной формулировкой, ограничиваясь в таких случаях лишь ссылкой на литературу, это не относится, однако, к задачам, которые, как нам казалось, окажутся интересными для специалистов. Мы также не включали в список задачи, *NP*-полнота которых непосредственно вытекает из *NP*-полноты других задач списка, опять-таки делая исключения для задач, представляющих значительный независимый интерес, и для задач, подзадачи которых нам хотелось бы прокомментировать. Таким образом, список не следует считать энциклопедически полным, содержащим все известные результаты об *NP*-полноте. Правильнее его считать, с одной стороны, аннотированной библиографией, служащей отправной точкой для изучения литературы по теории *NP*-полноты, а с другой стороны, базой данных, включающей обширный набор *NP*-полных задач, которыми можно воспользоваться для доказательства *NP*-полноты других задач.

А1. ТЕОРИЯ ГРАФОВ

А1.1. Покрытие и разбиение

[ТГ 1] ВЕРШИННОЕ ПОКРЫТИЕ

УСЛОВИЕ. Заданы граф $G(V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли в графе G вершинное покрытие мощности, не превосходящей K, т. е. существует ли подмножество $V' \in V$, такое что $|V'| \leqslant K$ (и для любого рёбра $\{u, v\} \in E$ по крайней мере одна из вершин u или v принадлежит V' ?

Источник [280]. К рассматриваемой задаче сводится задача 3-ВЫП (см. гл. 3).

Комментарий. При некоторых ограничениях на граф G эта задача эквивалентна по сложности задаче НЕЗАВИСИМОЕ МНОЖЕСТВО. Если потребовать, чтобы подграф, индуцированный множеством вершин V', был связен, то задача остаётся *NP*-полной даже для планарных графов, у которых степени всех вершин не превосходят 4, см. [149]. Задача легко решается за полиномиальное время, если потребовать, чтобы множество V' было одновременно и вершинным покрытием и независимым множеством в графе G. Близкая по постановке задача РЁБЕРНОЕ ПОКРЫТИЕ, в которой ищется наименьшее подмножество $E' \subseteq E$, такое, что каждая вершина $v \in V$ принадлежит по крайней мере одному ребру $e \in E'$, может быть решена за полиномиальное время, если искать паросочетания в графе (см., например, [324]).

[ТГ 2] ДОМИНИРУЮЩЕЕ МНОЖЕСТВО

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли в графе G доминирующее множество, состоящее не более чем из K элементов, т. е. подмножество $V' \subseteq V$, такое, что $|V'| \leqslant K$ и для всех вершин $u \in V \setminus V'$ существует такая вершина $v \in V$, что $\{u, v\} \in E$?

Источник. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной, если ограничиться графами со степенью вершин не более 3 или планарными регулярными графами степени 4 [152]. Модификация этой задачи, в которой требуется, чтобы индуцированный на множестве V' граф был связен, также является *NP*-полной, если ограничиться планарными регулярными графами степени 4 [152]. Эта задача будет *NP*-полной и в том случае, если потребовать, чтобы множество V' было одновременно доминирующим и независимым. На деревьях задача разрешима за полиномиальное время [81]. Близкая задача ДОМИНИРУЮЩЕЕ МНОЖЕСТВО РЁБЕР, в которой ищется подмножество $E' \subseteq E$, состоящее не более чем из K рёбер, таких, что любое ребро из E имеет общую вершину по крайней мере с одним ребром из множества E', является *NP*-полной даже для планарных и двудольных графов с максимальной степенью вершин, равной 3, однако на деревьях может быть решена за полиномиальное время [540], [390].

[ТГ 3] **ЧИСЛО ДОМИНИРУЮЩИХ ПОДМНОЖЕСТВ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Верно ли, что в графе G имеется не менее K различных доминирующих множеств, т. е. можно ли V разбить на $k \geqslant K$ непересекающихся подмножеств $V_1, V_2, \ldots, V_k$, таких, что V_i есть доминирующее множество в графе G?

Источник [159]. К этой задаче сводится задача 3-ВЫП. Рассматриваемая задача обсуждается в работе [82].

Комментарий. Задача остаётся *NP*-полной для любого фиксированного $K > 3$. (Если граф G не содержит изолированной вершины, то число доминирующих подмножеств равно по крайней мере 2).

[ТГ 4] **РАСКРАШИВАЕМОСТЬ ГРАФА (ХРОМАТИЧЕСКОЕ ЧИСЛО)**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Верно ли, что граф G K-раскрашиваем? (Граф называется K-раскрашиваемым, если существует такая функция $f : V \to \{1, 2, \ldots, K\}$, что если $\{u, v\} \in E$, то $f(u) \neq f(v)$.)

Источник [280]. К этой задаче сводится задача 3-ВЫП.

Комментарий. При $K = 2$ задача разрешима за полиномиальное время, но при всех фиксированных $K \geqslant 3$, а также для $K = 3$ и плоских графов, степени всех вершин которых не превосходят 4, задача *NP*-полна. Она остаётся также *NP*-полной при $K = 3$, когда G – это граф пересечений отрезков прямых линий на плоскости [112]. Задача *NP*-полна также для произвольного K и графов пересечения окружностей и дуг окружностей (даже если графы представлены в виде семейства дуг). Для графов пересечения дуг окружностей, представленных в виде семейства дуг, при любом фиксированном K задача разрешима за полиномиальное время [153]. В общем случае задача может быть решена за полиномиальное время для графов сравнимости [117], для триангулированных графов [163], для (3,1)-графов [532], а также для графов, не имеющих вершин степени выше 3 [56].

[ТГ 5] **АХРОМАТИЧЕСКОЕ ЧИСЛО**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Верно ли, что ахроматическое число графа G больше K? Иными словами, существует ли разбиение множества V на непересекающиеся подмножества $V_1, V_2, \ldots, V_k$, $k \geqslant K$, такие, что все V_i являются независимыми множествами графа G (т. е. никакие две вершины из V_i не соединены ребром из E) и для любой пары (V_i, V_j) различных подмножеств множество $V_i \cup V_j$ не есть независимое подмножество в G?

Источник [540]. К этой задаче сводится задача МИНИМАЛЬНОЕ ПО МОЩНОСТИ МАКСИМАЛЬНОЕ ПАРОСОЧЕТАНИЕ.

Комментарий. Задача остаётся *NP*-полной и в том случае, если G есть дополнение к двудольному графу, и, следовательно, любое независимое подмножество содержит не более двух вершин.

[ТГ 6] **МОНОХРОМАТИЧЕСКИЙ ТРЕУГОЛЬНИК**

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Существует ли разбиение множества E на два непересекающихся подмножества E_1 и E_2, такие, что ни один из графов $G_1 = (V, E_1)$ или $G_2 = (V, E_2)$ не содержит треугольника?

Источник [63]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Варианты этой задачи, в которых «треугольник» заменяется любым фиксированным полным графом с большим числом вершин, также *NP*-полны [63]. Варианты задачи, в которых «треугольник» заменяется на «k-звезду» (т. е. на граф, у которого центральная вершина степени k соединена с k другими вершинами степени 1), разрешимы за полиномиальное время [64].

[ТГ 7] **МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли подмножество $V' \subseteq V$, такое, что $|V'| \leqslant K$ и V содержит по крайней мере одну вершину любого ориентированного цикла в G?

Источник [280]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной, если ограничиться графами, у которых все вершины имеют степень захода (или выхода) не больше 2, или плоскими графами со степенью захода (или выхода) не более 3 [152], а также если ограничиться рёберными ориентированными графами [168]. Для редуцируемых графов задача может быть решена за полиномиальное время [481]. Соответствующая задача для неориентированных графов также *NP*-полна.

[ТГ 8] **МНОЖЕСТВО ДУГ, РАЗРЕЗАЮЩИХ КОНТУРЫ**

УСЛОВИЕ. Заданы ориентированный граф $G(V, A)$ и положительное целое число $K \leqslant |A|$.

ВОПРОС. Существует ли подмножество $A' \subseteq A$, такое, что $|A'| \leqslant A$ и A' содержит по крайней мере одну дугу из каждого ориентированного цикла в G?

Источник [280]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной, если ограничиться ориентированными графами, у которых все вершины имеют степень захода и выхода не более 3, или рёберными ориентированными графами [168]. Для планарных ориентированных графов задача разрешима за полиномиальное время [361]. Соответствующая задача для неориентированных графов тривиальным образом решается за полиномиальное время.

[ТГ 9] **МНОЖЕСТВО ДУГ, ЧАСТИЧНО РАЗРЕЗАЮЩИХ ЦИКЛЫ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительные целые числа K и L, $K \leqslant |E|$, $L \leqslant |V|$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \leqslant K$ и E' содержит по крайней мере одну дугу из каждого ориентированного цикла в G, имеющего длину не более L?

Источник [538]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся NP-полной для любого фиксированного $L \geqslant 3$, а также для двудольных графов (с фиксированным $L \geqslant 4$). Однако если $L = |V|$, т. е. если отыскивается множество E', содержащее ребро из каждого цикла в G, то задача легко решается за полиномиальное время.

[ТГ 10] **МИНИМАЛЬНОЕ ПО МОЩНОСТИ МАКСИМАЛЬНОЕ ПАРОСОЧЕТАНИЕ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \leqslant K$ и E' – максимальное паросочетание, т. е. никакие два ребра из E' не имеют общего конца, а любое ребро из множества $E \setminus E'$ имеет общий конец с одним из рёбер множества E'?

Источник [540]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ для кубических графов.

Комментарий. Задача остаётся NP-полной, если ограничиться планарными графами или двудольными графами, не имеющими вершин, степени выше 3. Задача отыскания максимального из «максимальных паросочетаний» аналогична обычной задаче о паросочетании и разрешима за полиномиальное время (см., например, [324]).

[ТГ 11] **РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ**

УСЛОВИЕ. Задан граф $G = (V, E)$, такой, что для некоторого целого числа $q : |V| = 3q$.

ВОПРОС. Можно ли разбить вершины графа G на q непересекающихся множеств $V_1, V_2, \ldots, V_q$, таких, что каждое из них содержит ровно 3 вершины и для всех $V_i = \{u_i, v_i, w_i\}$, $1 \leqslant i \leqslant q$, ребра $\{u_i, v_i\}$, $\{u_i, w_i\}$ и $\{v_i, w_i\}$ принадлежат множеству E?

Источник [472]. К этой задаче сводится задача 3-С (см. гл. 3).

Комментарий. Эта задача – частный случай задачи [ТГ 12].

[ТГ 12] **РАЗБИЕНИЕ НА ИЗОМОРФНЫЕ ПОДГРАФЫ**

УСЛОВИЕ. Заданы графы $G(V, E)$ и $H = (V', E')$, такие, что для некоторого целого числа $q \in Z^+ : |V| = q|V'|$.

ВОПРОС. Можно ли разбить вершины графа G на q непересекающихся множеств $V_1, V_2, \ldots, V_q$, таких, что при $1 \leqslant i \leqslant q$ подграфы графа G, индуцированные множествами V_i, изоморфны графу H?

Источник [288]. К этой задаче сводится задача 3-С.

Комментарий. Задача остаётся *NP*-полной для любого фиксированного графа H, содержащего по крайней мере 3 вершины. Если H – любой фиксированный граф, содержащий связную компоненту, имеющую не менее 3 вершин, то аналогичная задача, в которой требуется, чтобы число вершин в подграфах, индуцированных множествами V_i, было равно числу вершин графа H, а подграфы, индуцированные множествами V_i, содержали подграф, изоморфный H, также *NP*-полна. Для любого графа H, не удовлетворяющего сформулированному выше требованию, обе последние задачи могут быть решены за полиномиальное время (с помощью паросочетаний).

[ТГ 13] **РАЗБИЕНИЕ НА ГАМИЛЬТОНОВЫ ПОДГРАФЫ**

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$.

ВОПРОС. Можно ли разбить вершины графа G на непересекающиеся подмножества $V_1, V_2, \ldots, V_k$, такие, что каждое V_i содержит по крайней мере 3 вершины, а все индуцируемые ими подграфы графа G содержали бы гамильтоновы циклы?

Источник [522]. К этой задаче сводится 3-ВЫП. (См. также [208].)

Комментарий. С помощью метода паросочетаний задача разрешается за полиномиальное время, если потребовать, чтобы V_i содержало не менее двух вершин [108]. Аналогичная задача для неориентированных графов также может быть решена за полиномиальное время, даже если требовать, чтобы $|V_i| \geqslant 3$. Однако задача остаётся *NP*-полной, если требуется, чтобы $|V_i| \geqslant 6$ [414] или в условии задачи имеется верхняя граница для величины k.

[ТГ 14] **РАЗБИЕНИЕ НА ЛЕСА**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Можно ли разбить вершины графа G на $k \leqslant K$ непересекающихся подмножеств $V_1, V_2, \ldots, V_k$, таких, что для всех i, $1 \leqslant i \leqslant k$, подграф, индуцированный множеством V_i, не содержит циклов?

Источник [152]. К этой задаче сводится задача 3-РАСКРАШИВАЕМОСТЬ ГРАФА.

[ТГ 15] **РАЗБИЕНИЕ НА КЛИКИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Можно ли разбить вершины графа G на $k \leqslant K$ непересекающихся множеств $V_1, V_2, \ldots, V_k$, таких что для всех $i (1 \leqslant i \leqslant k)$ подграф, индуцированный множеством V_i, был бы полным?

Источник [280]. (В этой работе задача названа ПОКРЫТИЕ КЛИКАМИ.) К данной задаче сводится задача K-РАСКРАШИВАЕМОСТЬ ГРАФА.

Комментарий. Задача остаётся *NP*-полной для рёберных графов [22], для графов, не содержащих полных подграфов на 4 вершинах (см. конструкцию в задаче РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ в гл. 3), а также для всех фиксированных $K \geqslant 3$. Для $K \leqslant 2$ задача решается с помощью паросочетания за полиномиальное время, если граф не содержит полных подграфов на 3 вершинах. Она разрешима за полиномиальное время для графов пересечения дуг окружностей (если задано их представление в виде семейства дуг) [165], для триангулированных графов [163] и для графов сравнимости [178].

[ТГ 16] **РАЗБИЕНИЕ НА СОВЕРШЕННЫЕ ПАРОСОЧЕТАНИЯ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Можно ли вершины графа G разбить на $k(k \leqslant K)$ непересекающихся множеств $V_1, V_2, \dots, V_k$, таких, что для всех $i(1 \leqslant i \leqslant k)$ подграф, индуцированный множеством V_i, является совершенным паросочетанием (или состоит только из вершин степени 1)?

Источник [474]. К этой задаче сводится задача 3-ВЫП С РАЗНОЗНАЧНЫМИ ЛИТЕРАЛАМИ.

Комментарий. Для $K = 2$ и для планарных кубических графов задача остаётся *NP*-полной.

[ТГ 17] **ПОКРЫТИЕ КЛИКАМИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существуют ли в множестве V $k(k \leqslant K)$ подмножеств $V_1, V_2, \dots, V_k$, таких, что каждое подмножество индуцирует полный подграф графа G и для каждого ребра $\{u, v\} \in E$ существует подмножество V_i, содержащее как u, так и v?

Источник [298], [405]. К этой задаче сводится задача РАЗБИЕНИЕ НА КЛИКИ.

[ТГ 18] **ПОКРЫТИЕ ПОЛНЫМИ ДВУДОЛЬНЫМИ ПОДГРАФАМИ**

УСЛОВИЕ. Заданы двудольный граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существуют ли $k(k \leqslant K)$ подмножеств $V_1, \dots, V_k$ множества V, таких, что каждое подмножество V_i индуцирует полный двудольный подграф графа G и для каждого ребра $\{u, v\} \in E$ существует некоторое V_i, содержащее как u, так и v?

Источник [405]. К этой задаче сводится задача РАЗБИЕНИЕ НА КЛИКИ.

A1.2. Подграфы и надграфы

[ТГ 19] **КЛИКА**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Верно ли, что G содержит клику размера не менее K? Иными словами, существует ли подмножество $V' \in V$, такое, что $|V'| \geqslant K$ и любые две вершины в V' соединены ребром из E?

Источник [280]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ (см. гл. 3).

Комментарий. Задача разрешима за полиномиальное время для следующих графов: ограниченной степени d, планарных, рёберных, триангулированных [163], графов сравнимостей [117], графов пересечения окружностей [163], графов пересечения дуг окружностей (если задано их представление в виде семейства дуг) [165]. Вариант этой задачи, в котором при заданном r $(0 < r < 1)$ спрашивается, верно ли, что G содержит клику не менее чем на $r|V|$ вершинах, также NP-полон.

[ТГ 20] **НЕЗАВИСИМОЕ МНОЖЕСТВО**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Верно ли, что G содержит независимое множество мощности не менее K? Иными словами, верно ли, что существует подмножество $V' \subseteq V$, такое, что $|V'| \geqslant K$ и никакие две вершины из V не соединены ребром из E?

Источник. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ (см. гл. 3).

Комментарий. Задача остаётся NP-полной для кубических планарных графов [157, 149, 372], рёберных графов, ориентированных графов [168], тотальных графов, двудольных графов [540] и для графов, не содержащих треугольников [433]. Задача решается за полиномиальное время для двудольных графов (с помощью паросочетаний, см., например, [196]), рёберных графов (с помощью паросочетаний), графов, не имеющих вершин степени выше 2, триангулированных графов [163], графов пересечения окружностей [164], графов пересечения дуг окружностей (если задано представление в виде семейства дуг) [165], графов сравнимости [178] и для графов без «3-звезд» [389].

[ТГ 21] **ИНДУЦИРОВАННЫЙ ПОДГРАФ СО СВОЙСТВОМ П***

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли подмножество $V \leqslant V$, такое, что $|V'| \geqslant K$ и подграф графа G, индуцированный множеством V', обладает свойством Π (см. комментарий, где указаны возможные варианты выбора Π)?

Источник [537, 538, 341]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача NP-трудна для любого свойства Π, выполняющегося для произвольно больших, но не всех графов и обладающего «наследственностью» (т. е. свойство выполняется для всех подграфов графа, если оно выполняется для самого графа). Если в дополнение к этому для заданного графа за полиномиальное время можно проверить выполнение свойства П, то задача является *NP*-полной. Ниже приведены примеры таких свойств П: «G есть клика», «G есть независимое множество», «G – планарный граф», «G – двудольный граф», «G – внешнепланарный граф», «G – рёберный граф», «G– триангулированный граф», «G– граф сравнимости» и «G – лес». Те же результаты остаются верными, если G ограничено лишь семейством планарных (соответственно ациклических ориентированных) графов, а ограничения, наложенные на свойство П, выполняются для планарных (соответственно ациклических ориентированных) графов. Более слабые результаты имеют место для двудольных графов [538].

[ТГ 22] **ИНДУЦИРОВАННЫЙ СВЯЗНЫЙ ПОДГРАФ СО СВОЙСТВОМ П(*)**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли подмножество $V' \subseteq V$, такое, что $|V'| \geqslant K$ и подграф графа G, индуцированный подмножеством V', связен и обладает свойством П (возможные варианты выбора свойства П указаны в комментарии)?

Источник [538]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача *NP*-трудна для любого свойства с условием наследственности и выполняющегося для произвольно больших связных, но не для всех связных графов. Если в дополнение к этому для заданного графа за полиномиальное время можно проверить выполнение свойства П, то задача является *NP*-полной. В качестве П можно взять любое из свойств, перечисленных в предыдущей задаче, кроме «G есть независимое множество». Близкий по формулировке вопрос: «Верно ли, что максимальный индуцированный подграф графа G, обладающий свойством П, также является связным?» – не лежит ни в *NP*, ни в $co - NP$, если $NP \neq co - NP$ [538].

[ТГ 23] **ИНДУЦИРОВАННЫЙ ПУТЬ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли подмножество $V' \subseteq V$, такое, что $|V'| \geqslant K$ и подграф, индуцированный множеством V, является простым путем на $|V'|$ вершинах?

Источник [539]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Заметим, что в данном случае свойство не обладает наследственностью, поэтому настоящий результат не следует из двух предыдущих. Задача остаётся *NP*-полной, если G – двудольный граф. Этот же результат имеет место для варианта задачи, в котором вместо «простого пути» отыскивается «простой цикл». Задачи об отыскании самого длинного простого пути или самого длинного простого цикла (не обязательно индуцированных) также являются *NP*-полными.

[ТГ 24]**СБАЛАНСИРОВАННЫЙ ПОЛНЫЙ ДВУДОЛЬНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы двудольный граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существуют ли два непересекающихся подмножества $V_1, V_2 \subseteq V$, такие, что $|V_1| + |V_2| = K$, и из того, что $u \in V_1$, $v \in V_2$, следует, что $\{u, e\} \in E$?

Источник [152]. К этой задаче сводится задача КЛИКА.

Комментарий. Близкая задача, в которой требование «$|V_1| = |V_2| = K$» заменено требованием «$|V_1| + |V_2| = K$» для двудольных графов, разрешима за полиномиальное время (в силу имеющейся в таких графах связи между паросочетаниями и независимыми множествами, см., например, [196]), а для произвольных графов *NP*-полна [538].

[ТГ 25] **ДВУДОЛЬНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \geqslant K$ и $G' = (V, E')$ – двудольный подграф?

Источник [157]. К данной задаче сводится задача МАКСИМАЛЬНАЯ 2-ВЫПОЛНИМОСТЬ.

Комментарий. Задача остаётся *NP*-полной для графов, не имеющих вершин степени более 3 и не содержащих треугольников, и (или) если потребовать совязности искомого подграфа [538]. Задача разрешима за полиномиальное время, если G – пленарный граф [195, 406] или если $K = |E|$.

[ТГ 26] **СВЯЗНЫЙ ПОДГРАФ ОГРАНИЧЕННОЙ СТЕПЕНИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, неотрицательное целое число $d \leqslant |V|$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \geqslant K$ и подграф $G' = (V, E')$ связен и не имеет вершин степени более d?

Источник [538]. К этой задаче сводится задача ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Задача остаётся *NP*-полной для любого фиксированного $d \geqslant 2$. Разрешима (с помощью техники паросочетаний, см. [108]) за полиномиальное время, если не требовать связности G'. Соответствующая задача об индуцированном подграфе, в которой отыскивается подмножество $V' \subseteq V$, такое, что $|V| \geqslant K$ и подграф, индуцированный подмножеством V, не имеет вершин степени выше d, является *NP*-полной для любого фиксированного $d \geqslant 0$ [342a], а если потребовать связности графа G', то *NP*-полна для любого фиксированного $d \geqslant 2$ [538].

[ТГ 27] **ПЛАНАРНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \geqslant K$ и граф $G' = (V, E')$ планарен?

Источник [358]. К данной задаче сводится ограниченная на двудольные графы задача ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Соответствующая задача, в которой граф G' есть подграф, индуцированный подмножеством V, содержащим не менее K вершин, также *NP*-полна [306, 538]. Если $K = |E|$, то за полиномиальное время может быть решена первая задача, а если $K = |V|$ – то вторая (это объясняется тем, что проверку планарности графа можно осуществить за полиномиальное время; см., например, [213]). Близкая задача, в которой спрашивается: «Содержит ли G связный «внешнепланарный» подграф с K или более вершинами», также *NP*-полна [538].

[ТГ 28] **РЁБЕРНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \geqslant K$ и подграф $G' = (V, E')$ есть рёберный граф? (Граф $G' = (V, E')$ называется рёберным, если существует граф $H = (U, F)$, такой, что G' изоморфен графу с множеством вершин F и множеством рёбер, состоящим из всех пар рёбер $\{e, f\}$, таких, что e и f имеют общий конец из множества U.)

Источник [538]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если в G нет вершины, степень которой больше 4. Если потребовать связности подграфа, то *NP*-полнота задачи сохранится и при снижении ограничения на степень до 3. Реберные графы можно распознать за полиномиальное время (см., например, [196], где используется термин «рёберные графы»).

[ТГ 29] **ТРАНЗИТИВНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K \leqslant |A|$.

ВОПРОС. Существует ли подмножество $A' \subseteq A$, такое, что $|A'| \geqslant K$ и граф $G' = (V, A')$ транзитивен? (Граф $G' = (V, A')$ называется транзитивным, если для любой пары вершин u и v существует третья вершина $w \in V$, такая, что $(u, w), (w, v) \in A'$, то $(u, v) \in A'$.)

Источник [538]. К этой задаче сводится задача ДВУДОЛЬНЫЙ ПОДГРАФ (в которой рассматриваются только подграфы, не содержащие треугольников).

Комментарий. *NP*-полным является также вариант этой задачи, в котором G –неориентированный граф, и спрашивается, существует ли «подграф сравнимости», т. е. такой подграф, который с помощью выбора подходящей ориентации рёбер можно сделать транзитивным ориентированным графом. Этот вариант остаётся *NP*-полным даже тогда, когда в G нет вершин степени больше 3. В обоих случаях задачи остаются *NP*-полными, если потребовать связности искомого подграфа.

[ТГ 30] **ОДНОСТОРОННЕ СВЯЗНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K \leqslant |A|$.

ВОПРОС. Существует ли подмножество $A' \subseteq A$, такое, что $|A| \geqslant K$ и граф $G' = (V, A')$ имеет по крайней мере один ориентированный путь между любой парой вершин?

Источник [370]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной и для ациклических ориентированных графов.

[ТГ 31] **МИНИМАЛЬНЫЙ K. СВЯЗНЫЙ ПОДГРАФ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительные целые числа K и $B(K \leqslant |V|, B \leqslant |E|)$.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \leqslant B$ и граф $G' = (V, E')$ является K-связным (граф называется K-связным, если он остаётся связным после удаления любых $K - 1$ вершин (и инцидентных им рёбер)).

Источник [72]. К этой задаче сводится задача ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Соответствующая задача о рёберно-связном графе также *NP*-полна. Обе задачи (о вершинной и рёберной связности) *NP*-полны для любого фиксированного $K \geqslant 2$, а для $K = 1$ тривиально решаются за полиномиальное время.

[ТГ 32] **КУБИЧЕСКИЙ ПОДГРАФ**

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Существует ли непустое подмножество $E' \subseteq E$, такое, что в графе $G' = (V, E')$ любая вершина имеет степень, равную 3 или 0?

Источник [75]. К этой задаче сводится задача 3-РАСКРАШИВАЕМОСТЬ ГРАФА.

[ТГ 33] **МИНИМАЛЬНЫЙ ЭКВИВАЛЕНТНЫЙ ОРИЕНТИРОВАННЫЙ ГРАФ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K \leqslant |A|$.

ВОПРОС. Существует ли подмножество $A' \subseteq A$, такое, что $|A'| \leqslant K$ и для любой упорядоченной пары вершин $u, v \in V$ в графе $G' = (V, A')$ тогда и только тогда имеется ориентированный путь из u к v, если такой путь имеется в G.

[ТГ 34] **ГАМИЛЬТОНОВО ПОПОЛНЕНИЕ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и неотрицательное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли множество E', содержащее множество E и такое, что $|E' \setminus E| \leqslant K$, а граф $G' = (V, E')$ имеет гамильтонов цикл?

Источник. К этой задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся *NP*-полной для любого фиксированного $K \geqslant 0$. Аналогичным образом построенные задачи о пополнении для задач ГАМИЛЬТОНОВ ПУТЬ, ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ПУТЬ и ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ также *NP*-полны. Задачи ГАМИЛЬТОНОВО ПОПОЛНЕНИЕ и ГАМИЛЬТОНОВ ПУТЬ В ПОПОЛНЕНИИ могут быть решены за полиномиальное время, если G – дерево (см. [45]).

[ТГ 35] **ПОПОЛНЕНИЕ ДО ГРАФА ИНТЕРВАЛОВ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и неотрицательное целое число K.

ВОПРОС. Существует ли множество E', содержащее E, такое, что $|E' \setminus E| \leqslant K$ и граф $G' = (V, E')$ является графом интервалов?

Источник [140]. К данной задаче сводится задача ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ.

Комментарий. Задача остаётся *NP*-полной, если ограничиться только рёберными графами. Для $K = 0$ задача разрешима за полиномиальное время, см. [132, 54].

[ТГ 36] **ПОПОЛНЕНИЕ ДО ГРАФА ПУТЕЙ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и неотрицательное целое число K.

ВОПРОС. Существует ли множество E', содержащее множество E, такое, что $|E' \setminus E| \leqslant K$ и граф $G' = (V, E')$ есть граф пересечения семейства путей в неориентированном дереве?

Источник [169]. К этой задаче сводится ПОПОЛНЕНИЕ ДО ГРАФА ИНТЕРВАЛОВ.

Комментарий. Соответствующая задача, в которой требуется, чтобы G был графом пересечения ориентированных путей в ориентированном дереве (т. е. корневом дереве с направлением дуг от корня), также является *NP*-полной.

A1.3. Упорядочение вершин

[ТГ 37] **ГАМИЛЬТОНОВ ЦИКЛ**

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Имеется ли в G гамильтонов цикл?

Источник [280]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ (см. гл. 3).

Комментарий. Задача остаётся *NP*-полной в следующих случаях: (1) G – планарный, кубический, 3-связный и не имеет граней, ограниченных меньше чем пятью рёбрами [158], (2) G – двудольный [305], (3) G – квадрат некоторого графа [75], (4) в условии задачи задан некоторый гамильтонов путь в графе G [417], (5) G – рёберный граф. Задача разрешима за полиномиальное время, если в G нет вершин степени больше 2 или если G – рёберный граф (см., например, [357]). Куб нетривиального связного графа всегда содержит гамильтонов цикл, см. [277].

[ТГ 38] **ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ**

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$.

ВОПРОС. Имеется ли в G ориентированный гамильтонов цикл?

Источник [280]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ (см. гл. 3).

Комментарий. Задача остаётся *NP*-полной, если G – планарный граф, не имеющий вершин, инцидентных более чем трем дугам [432]. Задача разрешима за полиномиальное время, если степени захода (или выхода) всех вершин не превосходят 1, а также если G – граф турнира, см. [393], или рёберный граф (см., например, [357]).

[ТГ 39] **ГАМИЛЬТОНОВ ПУТЬ**

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Верно ли, что в G имеется гамильтонов путь?

Источник. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ (см. гл. 3).

Комментарий. Задача остаётся *NP*-полной в случаях (1) и (2) из задачи ГАМИЛЬТОНОВ ЦИКЛ и разрешима за полиномиальное время при тех же предположениях, при которых разрешима и задача ГЦ. Соответствующая задача ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ПУТЬ является *NP*-полной, к ней также применимы комментарии из задачи ОРИЕНТИРОВАННЫЙ ГЦ. Задача остаётся *NP*-полной, если в условии указана начальная или конечная или обе точки пути. Задача ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ПУТЬ может быть решена за полиномиальное время для ациклических ориентированных графов (см., например, [324]).

[ТГ 40] **ШИРИНА**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли линейное упорядочение множества V, такое, что его ширина не более K? Иными словами: «Существует ли взаимно-однозначная функция $f : V \to \{1, 2, \ldots, |V|\}$, такая, что для всех $\{u, v\} \in E$ выполнено соотношение $|f(u) - f(v)| \leqslant K$?

Источник [407]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Остаётся *NP*-полной для деревьев со степенью вершин, не превышающей 3 [143]. Задача соответствует минимизации «ширины» симметрической матрицы путем одновременных перестановок столбцов и строк.

[ТГ 41] **ШИРИНА ОРИЕНТИРОВАННОГО ГРАФА**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли такая взаимно-однозначная функция $f : V \to \{1, 2, \ldots, |V|\}$, что для всех $(u, v) \in A$ $f(u) < f(v)$ и $(f(v) - f(u)) \leqslant K$?

Источник [143]. К этой задаче сводится задача 3-РАЗБИЕНИЕ.

Комментарий. Задача остаётся *NP*-полной для корневых ориентированных деревьев с максимальной степенью захода, равной 1, и максимальной степенью выхода 2. Последняя задача соответствует минимизации «ширины» верхнетреугольной матрицы с помощью одновременных перестановок строк и столбцов.

[ТГ 42] **ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число K.

ВОПРОС. Существует ли взаимно-однозначная функция $f : V \to \{1, 2, \ldots, |V|\}$, такая, что $\sum_{\{u,v\} \in E} |f(u) - f(v)| \leqslant K$?

Источник [157]. К задаче сводится задача ПРОСТОЙ МАКСИМАЛЬНЫЙ РАЗРЕЗ.

Комментарий. Задача остаётся *NP*-полной для двудольных графов [118] и разрешима за полиномиальное время, если G – дерево [486, 176].

[ТГ 43] **ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ ОРИЕНТИРОВАННОГО ГРАФА**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и целое число K.

ВОПРОС. Существует ли такая взаимно-однозначная функция $f : V \to \{1, 2, \dots, |V|\}$, что $f(u) < f(v)$ для всех $(u, v) \in A$ и $\sum_{u,v \in A} |f(v) - f(u)| \leqslant K$?

Источник [118]. К этой задаче сводится задача ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ.

Комментарий. Задача разрешима за полиномиальное время, если G – дерево, даже в том случае, если рёбрам графа приписаны целые веса, а целевая функция является взвешенной суммой разностей [5].

[ТГ 44] **ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ С МИНИМАЛЬНЫМ РАЗРЕЗАНИЕМ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число K.

ВОПРОС. Существует ли такая взаимно-однозначная функция $f : V \to \{1, 2, \dots, |V|\}$, что для всех i, $1 < i < |V|$,

$$|\{\{u, v\} \in E : f(u) \leqslant i < f(v)\} \leqslant K?$$

Источник [498, 168]. К этой задаче сводится задача ПРОСТОЙ МАКСИМАЛЬНЫЙ РАЗРЕЗ.

[ТГ 45] **ПРЕДСТАВЛЕНИЕ В ВИДЕ КОРНЕВОГО ДЕРЕВА**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число K.

ВОПРОС. Существует ли корневое дерево $T = (U, F)$, где $|U| = |V|$ и взаимно-однозначная функция $f : V \to U$, обладающие следующими свойствами: для любого ребра $\{u, v\} \in E$ существует простой путь, начинающийся в корне и проходящий через $f(u)$ и $f(v)$, такой, что $\sum_{\{u,v\} \in E} d(f(u), f(v)) \leqslant K$ (здесь $d(x, y)$ – число рёбер на пути от x до y в дереве T)?

Источник [168]. К этой задаче сводится задача ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ.

[ТГ 46] **УПОРЯДОЧЕННОЕ ИСКЛЮЧЕНИЕ ДЛЯ ОРИЕНТИРОВАННОГО ГРАФА**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$, неотрицательное целое число K.

ВОПРОС. Существует ли для графа упорядочение исключения, заполненность которого не более K? Другими словами: «Существует ли взаимно-однозначная функция $f : V \to \{1, 2, \dots, |V|\}$, такая, что имеется самое большее K пар вершин $(u, v) \in (V \times V) \setminus A$, обладающие свойством: G содержит ориентированный путь из u в v, проходящий только через такие вершины w, что $f(w) < min\{f(u), f(v)\}$?

Источник [454]. К этой задаче сводится 3-ВЫП.

Комментарий. Эта задача возникает при применении исключения по Гауссу к разреженным матрицам. Для $K = 0$ задача решается за полиномиальное время. Аналогичная задача для неориентированных графов (соответствующих симметричным матрицам) эквивалентна задаче ПОПОЛНЕНИЕ ДО ТРИАНГУЛИРОВАННОГО ГРАФА, сложность которой пока остаётся не выясненной.

[ТГ 47] **ПОСЛЕДОВАТЕЛЬНОЕ ИСКЛЮЧЕНИЕ СТЕПЕНЕЙ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и последовательность $< d_1, d_2, ..., d_{|V|} >$ неотрицательных целых чисел, не превышающих $|V| - 1$.

ВОПРОС. Существует ли взаимно-однозначная функция $f : V \to \{1, 2, \dots, |V|\}$, со следующим свойством для всех i, $1 \leqslant i \leqslant |V|$: если $f(v) = i$, то существует в точности d_i вершин u, таких, что $f(u) > i$ и $\{u, v\} \in E$?

Источник [154]. К этой задаче сводится ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ.

Комментарий. Вариант этой задачи, в котором отыскивается такая функция f, что для всех i, $1 \leqslant i \leqslant |V|$, из равенства $f(v) = i$ следует, что существует ровно d_i вершин u, таких, что $\{u, v\} \in E$ тривиально решается за полиномиальное время.

А1.4. Морфизмы и изоморфизмы

[ТГ 48] **ИЗОМОРФИЗМ ПОДГРАФУ**

УСЛОВИЕ. Заданы два графа $G = (V_1, E_1)$ и $H = (V_2, E_2)$.

ВОПРОС. Содержит ли граф G подграф, изоморфный графу H? Иначе говоря, существуют ли подмножества V и E, $V \subseteq V_1$, $E \subseteq E_1$, такие, что $|V| = |V_2|$, $|E| = |E_2|$, и такая взаимно-однозначная функция $f := V_2 \to V$, что $\{u, v\} \in E$ тогда и только тогда, когда $\{f(u), f(v)\} \in E$?

Источник [91]. К этой задаче сводится задача КЛИКА.

Комментарий. Частные случаи этой задачи: КЛИКА, ПОЛНЫЙ ДВУДОЛЬНЫЙ ПОДГРАФ, ГАМИЛЬТОНОВ ЦИКЛ и т. д. Задача может быть решена за полиномиальное время, если G – лес, а H – дерево, см. [111, 449]. Остаётся *NP*-полной, если G – дерево, а H – лес (см. гл. 4) или если G – произвольный граф, а H – дерево (ГАМИЛЬТОНОВ ПУТЬ). Варианты этой задачи для ориентированных графов также *NP*-полны даже в том случае, если G – ациклический граф, а H– дерево [13]. Однако если G – ориентированный лес, а H – ориентированное дерево, то задачу можно решить за полиномиальное время. Если $|V_1| = |V_2|$ и $|E_1| = |E_2|$, то эта задача превращается в задачу ИЗОМОРФИЗМ ГРАФОВ, сложность которой остаётся неизвестной для ориентированных и неориентированных графов.

[ТГ 49] **НАИБОЛЬШИЙ ОБЩИЙ ПОДГРАФ**

УСЛОВИЕ. Заданы два графа $G = (V_1, E_1)$, $H = (V_2, E_2)$ и положительное число K.

ВОПРОС. Существуют ли подмножества $E_1' \subseteq E_1$ и $E_2' \subseteq E_2$, такие, что $|E_1'| = |E_2'| \geqslant K$ и подграфы $G' = (V, E_1')$ и $H' = (V_2, E_2')$ изоморфны?

Источник. К этой задаче сводится задача КЛИКА.

Комментарий. Задачу можно решить за полиномиальное время, если G и H –деревья, см. [111].

[ТГ 50] **МАКСИМАЛЬНОЕ ПАРОСОЧЕТАНИЕ ПОДГРАФОВ**

УСЛОВИЕ. Заданы два ориентированных графа $G = (V_1, A_1)$, $H = (V_2, A_2)$ и положительное целое число K.

ВОПРОС. Существует ли такое подмножество $R \subseteq V_1 \times V_2$, что $|R| \geqslant K$ и для всех $< u, u' >, < v, v' > \in R$ дуга (u, v) принадлежит множеству A_1 тогда и только тогда, когда $(u', v') \in A_2$?

Источник [152]. К этой задаче сводится задача КЛИКА. Она обсуждается также в работе [29].

[ТГ 51] **СТЯГИВАЕМОСТЬ ГРАФА**

УСЛОВИЕ. Заданы два графа $G = (V_1, E_1)$ и $H = (V_1, E_2)$.

ВОПРОС. Можно ли последовательностью стягиваний рёбер графа G получить граф, изоморфный H? (Последовательность стягиваний рёбер – это последовательность шагов, на каждом из которых две соседние вершины u и v заменяются одной вершиной w, соединённой рёбрами с теми и только теми вершинами, с которыми были соединены u или v.)

Источник [494]. К данной задаче сводится задача 3-ВЫП.

Комментарий. Задача может быть решена за полиномиальное время, если H – треугольник.

[ТГ 52] **ГОМОМОРФИЗМ ГРАФОВ**

УСЛОВИЕ. Заданы два графа $G = (V_1, E_1)$ и $H = (V_2, E_2)$.

ВОПРОС. Можно ли из графа G последовательностью отождествления несоседних вершин получить граф H? (Последовательность отождествления несоседних вершин – это последовательность шагов, заменяющих две несоседние вершины u и v одной вершиной w, соединенной в точности с теми вершинами, с которыми были соединены вершины u или v.)

Источник [340]. К этой задаче сводится задача K-РАСКРАШИВАЕМОСТЬ ГРАФА.

Комментарий. Задача остаётся *NP*-полной, если H – треугольник, но если H – одно ребро, то решается за полиномиальное время.

[ТГ 53] **D-МОРФИЗМ ОРИЕНТИРОВАННЫХ ГРАФОВ**

УСЛОВИЕ. Заданы два ориентированных графа $G = (V_1, A_1)$ и $H = (V_2, A_2)$.

ВОПРОС. Существует ли D-морфизм графа G в граф H? (D-морфизмом называется такая функция $f : V_1 \to V_2$, что (1) для каждой пары $(u, v) \in A_1$ либо $(f(u), f(v)) \in A_2$, либо $(f(v), f(u)) \in A_2$. (2) для всех $u \in V_1$ и $v' \in V_2$, если $(f(u), v') \in A_2$, то существует такая вершина $v, v \in f^{-1}(v')$, для которой $(u, v) \in A_1$).

Источник [129]. К данной задаче сводится задача НУМЕРАЦИЯ ГРАФА ПО ГРУНДИ.

A1.5. Разное

[ТГ 54] **ПУТЬ С ЗАПРЕЩЁННЫМИ ПАРАМИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$, набор $C = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ пар вершин из V и две выделенные вершины $s, t \in V$.

ВОПРОС. Существует ли в G ориентированный путь из s в t, содержащий не более одной вершины из каждой пары набора C?

Источник [134]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, если ограничиться ациклическими графами, не имеющими вершин со степенями захода или выхода более 2. Вариант этой задачи, в котором запрещены не вершины, а дуги, также *NP*-полон (при аналогичных предположениях). Обе задачи остаются *NP*-полными, если вершины любой пары из заданного набора не являются соседними.

[ТГ 55] **ПАРОСОЧЕТАНИЕ СО МНОГИМИ ВАРИАНТАМИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, разбиение множества E на непересекающиеся множества $E_1, E_2, \ldots, E_j$ и положительное целое число K.

ВОПРОС. Существует ли подмножество $E' \subseteq E$, такое, что $|E'| \geqslant K$, никакие два ребра из E' не имеют общей вершины и для всех $i (1 \leqslant i \leqslant J)$ $E' \cap E_i \neq 0$.

Источник [524, 253, 255]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной и в том случае, если G – двудольный граф, каждое из множеств E_i содержит, самое большее, 2 ребра и $K = |V|/2$. Если каждое из множеств E_i состоит из одного ребра, то задача превращается в обычную задачу о паросочетании в графе, разрешимую за полиномиальное время.

[ТГ 56] **НУМЕРАЦИЯ ГРАФА ПО ГРУНДИ**

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$.

ВОПРОС. Существует ли такая функция $f : V \to Z^+$, что для всех вершин $v \in V f(v)$ есть наименьшее неотрицательное целое число, не содержащееся в множестве $\{f(u) : u \in V, (v, u) \in A\}$?

Источник [526]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, если ограничиться только планарными графами, у которых все вершины имеют степень захода или выхода не более 5 (см. [129]).

[ТГ 57] **ЯДРО**

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$.

ВОПРОС. Верно ли, что в G есть ядро? (Ядром ориентированного графа $G = (V, A)$ называется такое подмножество вершин V', что никакие две вершины из V' не соединены дугой из A и для каждой вершины $v \in V \setminus V$ найдётся такая вершина $u \in V \setminus V'$, что $(u, v) \in A$.)

Источник [73]. К этой задаче сводится задача 3-ВЫП.

[ТГ 58] **K-ЗАМЫКАНИЕ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ и положительное целое число $K \leqslant V$.

ВОПРОС. Существует ли такое подмножество $V' \subseteq V$, что $|V'| \leqslant K$ и для любой дуги $(u, v) \in A$ либо $u \in V'$, либо $v \notin V'$?

Источник [439]. К данной задаче сводится задача КЛИКА.

[ТГ 59] **БАЗИС ГРАФА ПЕРЕСЕЧЕНИЙ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Верно ли, что G – граф пересечений семейства множеств, объединение которых имеет мощность не более K? Иными словами: «Существует ли K-элементное множество S и подмножества $S[v] \subseteq S$, $v \in V$, такие что $\{u, v\} \in E$ тогда и только тогда, когда $S[u] \cap S[w] \neq 0$?»

Источник [298]. К этой задаче сводится ПОКРЫТИЕ КЛИКАМИ.

[ТГ 60] **МНОЖЕСТВО ДУГ, РАЗЛИЧАЮЩИЕ ПУТИ**

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$, в котором выделены две вершины s, t и положительное целое число $K \leqslant |A|$.

ВОПРОС. Существует ли такое подмножество $A' \subseteq A$, что $|A'| \leqslant K$ и для каждой пары путей p_1 и p_2 из s в t найдётся дуга из A', принадлежащая ровно одному из них?

Источник [370]. К этой задаче сводится задача ВЕРШИННОЕ ПОКРЫТИЕ.

[ТГ 61] **МЕТРИЧЕСКАЯ РАЗМЕРНОСТЬ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли базис метрик на графе G мощности не более K? (Подмножество $V' \subseteq V$ называется базисом метрик графа G, если для каждой пары вершин $u, v \in V$ существует такая вершина $w \in V'$, что длина кратчайшего пути от u к w отлична от длины кратчайшего пути между v и w.)

Источник [152]. К данной задаче сводится задача 3-С. Понятие метрической размерности определяется в работе [197].

[ТГ 62] **РАЗМЕРНОСТЬ НЕШЕТРИЛ - РЁДЛ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли такая взаимно-однозначная функция

$$f : V \to \{(a_1, a_2, \dots, a_K) : 1 \leqslant a_i \leqslant |V| \text{ при } 1 \leqslant i \leqslant K,$$

что пара вершин $u, v \in V$ соединена ребром $\{u, v\} \in E$ тогда и только тогда, когда $f(u)$ и $f(v)$ различны по всем K координатам?

Источник [400]. К этой задаче сводится 3-РАСКРАШИВАЕ-МОСТЬ ГРАФА. Понятие размерности определяется в работе [401].

[ТГ 63] **ПОРОГОВОЕ ЧИСЛО**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |E|$.

ВОПРОС. Существует ли разбиение множества E на такие непересекающиеся подмножества $E_1, E_2, \dots, E_K$, что каждый из графов $G_i = (V, E_i)$, $1 \leqslant i \leqslant K$, является «пороговым»?

Источник [78]. К этой задаче сводится задача НЕЗАВИСИМОЕ МНОЖЕСТВО (ограниченная на графы, не имеющие треугольников) .

Комментарий. При $K = 1$ задача решается за полиномиальное время.

[ТГ 64] **ОРИЕНТИРОВАННЫЙ ДИАМЕТР**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Можно ли так ориентировать ребра графа G, чтобы возникший в результате ориентированный граф был сильно связен и имел диаметр не более K?

Источник [79]. К этой задаче сводится задача РАСЩЕПЛЕНИЕ МНОЖЕСТВА.

Комментарий. Вариант задачи, в котором «диаметр» заменен на «радиус», также NP-полон. Обе задачи остаются NP-полными при $K = 2$.

[ТГ 65] **ВЗВЕШЕННЫЙ ДИАМЕТР**

УСЛОВИЕ. Заданы граф $G = (V, E)$, и набор C, состоящий из $|E|$ неотрицательных целых чисел (не обязательно различных), и положительное целое число K.

ВОПРОС. Существует ли такая взаимно-однозначная функция $f : E \to C$, что при интерпретации $f(e)$ как длины ребра e диаметр графа G не превосходит K? (Т. е. между любыми двумя вершинами $u, v \in V$ имеется путь длины не более K.)

Источник [425]. К этой задаче сводится задача 3-РАЗБИЕНИЕ.

Комментарий. Задача NP-полна в сильном смысле даже в том случае, когда G – дерево. Вариант задачи, в котором «диаметр» заменяется «радиусом», имеет такую же сложность. Если C состоит только из нулей и единиц, то оба варианта (с диаметром и радиусом) для деревьев разрешимы за полиномиальное время, а для общих графов остаются NP-полными, даже если K фиксировать равным 2 (диаметр) или 1 (радиус). Для общих графов вариант задачи, в котором ищется функция, дающая диаметр не менее K, NP-полон в сильном смысле, для деревьев в случае диаметра задача разрешима за полиномиальное время, а в случае радиуса NP-полна.

А2. ПОСТРОЕНИЕ СЕТЕЙ

А2.1. Остовные деревья

[1]

---

[1] Как синонимы часто употребляются термины: дерево-остов, остов графа, или просто остов. – *Прим. ред.*

[ПС 1] **ОСТОВ ОГРАНИЧЕННОЙ СТЕПЕНИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли в графе G остов, у которого степени всех вершин не превосходят K?

Источник. К этой задаче сводится задача ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Для любого фиксированного $K \geqslant 2$ задача остаётся *NP*-полной.

[ПС 2] **ОСТОВ С МАКСИМАЛЬНЫМ ЧИСЛОМ ВИСЯЧИХ ВЕРШИН**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли в графе G остов, у которого не менее K вершин степени 1?

Источник [152]. К указанной задаче сводится задача ДОМИНИРУЮЩЕЕ МНОЖЕСТВО.

Комментарий. Задача остаётся *NP*-полной, если G – регулярный граф степени 4 или если G – планарный граф, степени вершин которого не превосходят 4.

[ПС 3] **ОСТОВ С КРАТЧАЙШИМ ПОЛНЫМ ПУТЕМ**

УСЛОВИЕ. Заданы граф G = (V,E), целое число $B \in Z^+$.

ВОПРОС. Существует ли такой остов T графа G, что сумма (по всем парам вершин $u, v \in G$) длин путей от u до v в T не превосходит K?

Источник [266]. К этой задаче сводится задача ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ.

[ПС 4] **ОСТОВ ОГРАНИЧЕННОГО ДИАМЕТРА**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(e) \in Z^+$ каждой вершины $e \in E$, а также положительные числа B и $D \leqslant |V|$.

ВОПРОС. Существует ли в графе G такой остов T, что сумма весов рёбер из T не превосходит B и в T нет простого пути, число рёбер которого не превосходит D?

Источник [125]. К этой задаче сводится задача ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ.

Комментарий. Задача остаётся *NP*-полной при любом фиксированном $D \geqslant 4$ даже в том случае, если веса рёбер принимают два значения: 1 или 2. Легко решается за полиномиальное время, если $D \leqslant 3$ или если веса всех рёбер равны.

[ПС 5] **ОСТОВ С ОГРАНИЧЕННОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с выделенной вершиной $v_0 \in V$, пропускные способности $c(e) \in Z_0^+$ и длина $l(e) \in Z_0^+$ каждого ребра $e \in E$, потребность $r(v) \in Z_0^+$ для каждой вершины $v \in V \setminus \{v_0\}$ и граница $B \in Z_0^+$.

ВОПРОС. Существует ли в графе G такой остов T, что сумма длин его рёбер не превосходит B и для каждого ребра e в T выполнено соотношение

$\sum_{u \in U(e)} r(u) \leqslant c(e)$ (где $U(e)$ –множество вершин, для которых путь в T, соединяющий эту вершину с вершиной v_0, содержит ребро e)?

Источник [409]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача *NP*-полна в сильном смысле даже в том случае, если все потребности равны 1, а все пропускные способности равны 3. Если же все потребности равны 1, а пропускные способности равны 2, то задача решается за полиномиальное время с помощью метода взвешенных паросочетаний. Если все потребности равны 0 или 2, а все пропускные способности равны 1, то задача также может быть решена за полиномиальное время (с помощью алгоритмов построения потоков минимальной стоимости, см., например, [ПО]), но если все пропускные способности равны 2, а потребности и длины рёбер равны либо 0, либо 1, то задача остаётся *NP*-полной [115].

[ПС 6] **ГЕОМЕТРИЧЕСКИЙ ОСТОВ С ОГРАНИЧЕННОЙ ПРОПУСКНОЙ СПОСОБНОСТЬЮ**

УСЛОВИЕ. Заданы множество $P \subseteq Z \times Z$ точек плоскости, выделенная точка $p_0 \in P$, потребность $r(p) \in Z_0^+$ для каждой $p \in P \setminus p_0$, пропускная способность $c \in Z^+$ и граница $B \in Z^+$.

ВОПРОС. Существует ли в полном графе $G = (P, E)$ такой остов $T = (P, E')$, что $\sum_{e \in E'} d(e) \leqslant B$ (где $d((x_1, y_1), (x_2, y_2) = \lceil r((x_1 + x_2)^2 + (y_1 - y_2)^2)^{1/2} \rceil$

– дискретное евклидово расстояние), причём $sum_{u \in U(e)} r(u) \leqslant c$ для каждого ребра $e \in E'$ (где $U(e)$ – множество таких вершин v, что путь в дереве T, соединяющий v с p_0, содержит ребро e)?

Источник [409]. К этой задаче сводится задача ТП-3.

Комментарий. Задача остаётся *NP*-полной и в том случае, если все потребности равны между собой.

[ПС 7] **ОПТИМАЛЬНЫЙ КОММУНИКАЦИОННЫЙ ОСТОВ**

УСЛОВИЕ. Заданы полный граф $G = (V, E)$, вес $w(e) \in Z_0^+$ каждого ребра $e \in E$, потребность $r(\{u, v\}) \in Z_0^+$ для каждой пары $\{u, v\}$ вершин из V, граница $B \in Z_0^+$.

ВОПРОС. Существует ли в G такой остов T, что

$$\sum_{u,v \in V} W(\{u, v\}) \cdot r(\{u, v\}) \leqslant B,$$

где $W(\{u, v\})$ обозначает сумму весов рёбер пути в T, соединяющего u и v?

Источник [266]. К этой задаче сводится задача ТП-3.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда все потребности равны между собой. Если все веса равны между собой, то задача разрешима за полиномиальное время [227].

[ПС 8] **ИЗОМОРФНЫЙ ОСТОВ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, дерево $T = (V_T, E_T)$.

ВОПРОС. Верно ли, что G содержит остовное дерево, изоморфное T?

Источник. К этой задаче сводится задача ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Задача остаётся *NP*-полной в следующих случаях: (а) T – путь, (б) T – полное бинарное дерево, см. [418], (в) T –3-звезда (т. е. $V_T = \{v_0\} \cup \{u_i, v_i, w_{:} 1 \leqslant i \leqslant n\}$, $E_T = \{\{v_0, u_i\}, \{u_i, v_i\}, \{v_i, w_i\} : 1 \leqslant i \leqslant n\}$), см. [152]. Если G – граф паросочетания или 2-звезда, то задача разрешима за полиномиальное время. Классификацию сложности этой задачи для других типов деревьев см. в [418].

[ПС 9] K **НАИЛУЧШИХ ОСТОВОВ (*)**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(e) \in Z_0^+$ каждого ребра $e \in E$, положительные целые числа K и B.

ВОПРОС. Существует ли в G K различных остовов, общий вес которых не превосходит B?

Источник [260]. К этой задаче сводится по Тьюрингу задача ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Принадлежность этой задачи классу *NP* не установлена. Задача может быть решена за псевдополиномиальное время (время, ограниченное полиномом от $|V|, K, \log B, \max\{\log w(e) : e \in E\}$), см. [322], и, следовательно, при любом фиксированном K может быть решена за полиномиальное время. Соответствующая задача перечисления является KP-полной. Однако вариант задачи перечисления, в котором отсутствуют веса, решается за полиномиальное время (см., например, [198]).

[ПС 10] **ЛЕС С ОГРАНИЧЕННЫМИ КОМПОНЕНТАМИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(v) \in Z_0^+$ каждой вершины $v \in V$, положительные целые числа K и $B(K \leqslant |V|)$.

ВОПРОС. Можно ли вершины из V так разбить на непересекающиеся множества $V_1, V_2, \ldots, V_k$ $(k \leqslant K)$, что подграф графа G, индуцированный множеством V_i, является деревом, а сумма весов вершин из V_i не превосходит B?

Источник [194]. К этой задаче сводится задача РАЗБИЕНИЕ НА ПУТИ ДЛИНЫ 2.

Комментарий. Задача остаётся *NP*-полной и тогда, когда все веса равны 1, а B – произвольное фиксированное целое число, большее 2 [152]. Задача может быть решена за полиномиальное время, если G – дерево или если все веса равны 1 и $B = 2$ [194].

[ПС 11] **ВЕТВЛЕНИЕ СО МНОГИМИ ВАРИАНТАМИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$, вес $w(a) \in Z^+$ каждой дуги $a \in A$ разбиение множества A на непересекающиеся множества $A_1, A_2, \ldots, A_m$ и положительное целое число K.

ВОПРОС. Существует ли такое подмножество $A' \subseteq A$, что $\sum a \in A' w(a) \geqslant K$, никакие две дуги из A' не оканчиваются в одной вершине, A' не содержит циклов и A' содержит не более одной дуги из каждого множества $A_i, 1 \leqslant i \leqslant n$?

Источник [152]. К этой задаче сводится задача 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда G сильно связан и если все веса равны между собой. Если $|A_i] = 1$ для всех i, то задача превращается просто в задачу отыскания «ветвления максимального веса», т. е. задачу отыскания пересечения двух матроидов, разрешимую за полиномиальное время (см., например, [512]). (В сильно связном графе ветвление максимального веса можно рассматривать как ориентированный остов максимального веса). Аналогичным образом если граф симметричен, то задача становится эквивалентной задаче об остове со многими вариантами (другой задаче о пересечении двух матроидов) и оказывается разрешимой за полиномиальное время [509].

[ПС 12] **ДЕРЕВО ШТЕЙНЕРА НА ГРАФЕ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(e) \in Z_0^+$ каждого ребра $e \in E$, подмножество $R \in V$ и положительная целая граница B.

ВОПРОС. Существует ли в графе G такое поддерево, содержащее все вершины из R, что сумма весов рёбер этого поддерева не превосходит B?

Источник [280]. К этой задаче сводится задача ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ.

Комментарий. Задача остаётся *NP*-полной и в том случае, если G – двудольный граф, не имеющий рёбер, соединяющих вершины множества R или $V \setminus R$ (см. [37]), или если G – планарный граф [149].

[ПС 13] **ГЕОМЕТРИЧЕСКОЕ ДЕРЕВО ШТЕЙНЕРА**

УСЛОВИЕ. Заданы множество $P \subseteq Z \times Z$ точек на плоскости и положительное число K.

ВОПРОС. Существует ли такое конечное подмножество $Q \in Z \times Z$, что для множества вершин $P \cup Q$ существует остов с весом не более K (где вес ребра $\{(x_1, y_1), (x_2, y_2)\}$ равен $\lceil((x_1 - -x_2)^2 + (Y_1 - -y_2)^2)^{1/2}\rceil$ – дискретному евклидову расстоянию между (x_1, y_1) и (x_2, y_2))?

Источник [142]. К этой задаче сводится задача ТП-3.

Комментарий. Задача *NP*-полна в сильном смысле. Остаётся *NP*-полной, если дискретное расстояние заменить «метрикой типа L_1»: $|x_1 - x_2| + |y_1 - y_2|$ (см. [149]) «или метрикой типа L_∞»: $\max\{|x_1 - x_2|, |y_1 - y_2|\}$ (последняя метрика при повороте на 45° эквивалентна метрике L_1). Задача остаётся *NP*-трудной в сильном смысле, если принять (недискретизированное) евклидово расстояние $((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2}$, но в последнем случае принадлежность задачи классу *NP* не установлена [142]. Некоторые полиномиальные алгоритмы для специальных случаев метрики типа L_1 предложены в работе [6].

A2.2. Разрезы и связность

[ПС 14] **РАЗБИЕНИЕ ГРАФА**

УСЛОВИЕ. Заданы граф $G = (V, A)$, вес $w(v) \in Z^+$ каждой вершины $v \in V$, длина $l(e) \in Z^+$ каждого ребра $e \in E$, а также положительные целые числа K и J.

ВОПРОС. Существует ли такое разбиение множества V на непересекающиеся подмножества $V_1, V_2, \ldots, V_m$, что

$$\sum_{v \in V_i} w(v) \leqslant K (\text{для всех } i, 1 \leqslant i \leqslant K),$$

и если $E' \subseteq E$ – множество рёбер, имеющих концы в разных множествах V_i, то

$$\sum_{e \in E'} l(e) \leqslant J?$$

Источник [243]. К этой задаче сводится задача РАЗБИЕНИЕ НА ТРЕУГОЛЬНИКИ.

Комментарий. Задача остаётся *NP*-полной для фиксированного $K \geqslant 3$ даже в том случае, если все веса равны 1. При $K = 2$ задача может быть решена за полиномиальное время методом паросочетаний.

[ПС 15] **АЦИКЛИЧЕСКОЕ РАЗБИЕНИЕ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$, вес $w(v) \in Z^+$ каждой вершины $v \in V$, стоимость $c(a) \in Z^+$ каждой дуги $a \in A$, положительные целые числа B и K.

ВОПРОС. Существует ли такое разбиение множества V на непересекающиеся множества $V_1, V_2, \ldots, V_m$, что ориентированный граф $G' = (V', A')$ (где $V' = \{V_1, V_2, \ldots, V_m\}$ и $(V_i, V_j) \in A'$ тогда и только тогда, когда $(v_i, v_j) \in A$ для некоторых $v_i \in V_i$ и $v_j \in V_j$) является ациклическим, сумма весов вершин каждого из множеств V_i не превосходит B и сумма стоимостей всех дуг, имеющих концы в разных множествах, не превосходит K?

Источник [152]. К этой задаче сводится задача ТП-3.

Комментарий. Задача остаётся *NP*-полной и в том случае, если $w(v) = 1$ для всех $v \in V$ и $c(a) = 1$ для всех $a \in A$. Задача может быть решена за полиномиальное время, если G содержит некоторый гамильтонов путь. Последнее свойство для ациклических графов может быть проверено за полиномиальное время (см. [287]). Если G – дерево, то общая задача *NP*-полна в обычном смысле, но может быть решена за псевдополиномиальное время (см. [365]). Все три задачи могут быть решены за полиномиальное время, если веса всех рёбер равны между собой (см. [194]) или если веса всех вершин равны между собой (см. [152]).

[ПС 16] **МАКСИМАЛЬНЫЙ РАЗРЕЗ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(e) \in Z^+$ каждого ребра $e \in E$, положительное целое число K.

ВОПРОС. Существует ли разбиение множества V на два таких непересекающихся множества V_1 и V_i, что сумма весов рёбер из E, соединяющих вершины из множеств V_1 и V_2, не менее K?

Источник [280]. К этой задаче сводится задача МАКСИМАЛЬНАЯ 2-ВЫПОЛНИМОСТЬ.

Комментарий. Задача остаётся *NP*-полной, если $w(e) = 1$ для всех $e \in E$ (она называется ПРОСТОЙ МАКСИМАЛЬНЫЙ РАЗРЕЗ), см. [157], и если, кроме того, в графе нет вершин степени более 3 см. [538]. Если G – пленарный граф, то задача может быть решена за полиномиальное время [195,406].

[ПС 17] **МИНИМАЛЬНЫЙ РАЗРЕЗ С ОГРАНИЧЕНИЯМИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с двумя выделенными вершинами $s, t \in V$, вес $w(e) \in Z^+$ каждого ребра $e \in E$, а также положительные целые числа B и K ($B \leqslant |V|$).

ВОПРОС. Существует ли разбиение множества V на два таких подмножества V_1 и V_2, что $s \in V_1$, $t \in V_2$, $|V_1| \leqslant B$, $|V_2| \leqslant B$ и сумма весов рёбер из E, соединяющих вершины из V_1 и V_2, не превосходит K?

Источник [157]. К этой задаче сводится задача ПРОСТОЙ МАКСИМАЛЬНЫЙ РАЗРЕЗ.

Комментарий. Задача остаётся NP-полной для $S = |V|/2$ и если $w(e) = 1$ для всех $e \in E$. Если $B = |V|$, то она может быть решена за полиномиальное время стандартными методами теории потоков в сетях.

[ПС 18] **ДВУСВЯЗНОЕ ПОПОЛНЕНИЕ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(\{u, v\}) \in Z^+$ каждой неупорядоченной пары $\{u, v\}$ вершин из V и положительное целое число B.

ВОПРОС. Существует ли такое множество E', состоящее из неупорядоченных пар вершин V, что $\sum_{e \in E'} w(e) \leqslant B$ и граф $G' = (V, E \cup E')$ – двусвязен (граф называется двусвязным, если после удаления любой отдельной вершины он остаётся связным)?

Источник [113]. К данной задаче сводится задача ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Близкая задача, в которой граф G' рёберно связан (т. е. не распадается после удаления одного ребра), также NP-полна. Обе задачи остаются NP-полными, если все веса равны либо 1, либо 2 и E пусто. Обе задачи могут быть решены за полиномиальное время, если все веса равны между собой.

[ПС 19] **СИЛЬНО СВЯЗНОЕ ПОПОЛНЕНИЕ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$, вес $w(u, v) \in Z^+$ каждой упорядоченной пары вершин $(u, v) \in V \times V$ и положительное целое число B.

ВОПРОС. Существует ли такое множество A' упорядоченных пар вершин из V, что $\sum_{a \in A'} w(a) \leqslant B$ и граф $G' = (V, A \cup A')$ сильно связен?

Источник [113]. К этой задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся NP-полной, если все веса равны 1 или 2, а A пусто. Может быть решена за полиномиальное время, если все веса равны между собой.

[ПС 20] **НАДЁЖНОСТЬ СЕТИ (*)**

УСЛОВИЕ. Заданы граф $G = (V, E)$, подмножество $V' \subseteq V$, для каждого $e \in E$ рациональное число $p(e)$, $0 \leqslant p(e) \leqslant 1$,– вероятность неисправности, положительное рациональное число $q \leqslant 1$.

ВОПРОС. Верно ли, что с вероятностью, не меньшей q, любые две вершины из V соединены по крайней мере одним путем, не содержащим неисправных рёбер. (Предполагается, что неисправности в ребрах возникают независимо.)

Источник [458]. К этой задаче сводится ДЕРЕВО ШТЕИНЕРА НА ГРАФЕ.

Комментарий. Принадлежность этой задачи классу *NP* не установлена. Задача остаётся *NP*-полной даже в том случае, если $|V'| \geqslant 2$ [523]. Аналогичная задача, в которой требуется, чтобы каждую пару вершин из V соединяли по крайней мере два непересекающихся пути, *NP*-трудна даже тогда, когда $V' = V$, [28]. Если G – ориентированный граф и требуется выяснить, существует ли ориентированный путь, соединяющий любую пару упорядоченных вершин в V, то задача *NP*-трудна как при $|V'| = 2$ [523], так и при $V' = V$ [27]. Многие задачи перечисления подграфов, удовлетворяющих указанным выше свойствам, KP-полны (см. [523]).

[ПС 21] **ВЫЖИВАЕМОСТЬ СЕТИ (*)**

УСЛОВИЕ. Заданы граф $G = (V, E)$, для каждого $x \in V \cup E$ рациональное число $p(x)$, $0 \leqslant p(x) \leqslant 1$,– «вероятность неисправности», положительное рациональное число $q \leqslant 1$.

ВОПРОС. Верно ли, что вероятность того, что для всех $u, v \in E$ по крайней мере один элемент u, v или $\{u, v\}$ неисправен, не меньше q? (В предположении, что неисправности рёбер и вершин независимы.)

Источник [458]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Принадлежность этой задачи классу *NP* не установлена.

А2.3. Задачи о маршрутах

[ПС 22] **КОММИВОЯЖЕР**

УСЛОВИЕ. Заданы множество C, включающее m городов, расстояние $d(c_i, c_j) \in Z^+$ между каждой парой городов $c_i, c_j \in C$ и положительное целое число B.

ВОПРОС. Существует ли маршрут длины не более B, проходящий через все города из C? Иными словами, существует ли такая перестановка городов $< c_{\pi(1)}, c_{\pi(2)}, \dots, c_{\pi(m)} >$, что

$$\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(m)}) \leqslant B?$$

Источник. К задаче сводится задача ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся *NP*-полной и в том случае, когда $d(c_i, c_j) \in \{1, 2\}$ для всех $c_i, c_j \in C$. Особые, разрешимые за полиномиальное время случаи этой задачи обсуждаются в работах [172, 160, 510]. Вариант этой задачи, в котором ищется маршрут со средним временем прибытия, не превосходящим B, также *NP*-полон (см. [469]).

[ПС 23] **ГЕОМЕТРИЧЕСКАЯ ЗАДАЧА О КОММИВОЯЖЕРЕ**

УСЛОВИЕ. Заданы множество $P \subseteq Z \times Z$ точек на плоскости и положительное целое число B.

ВОПРОС. Существует ли маршрут длины, не большей B, для индивидуальной задачи КОММИВОЯЖЕР, в которой $C = P$ и $d((x_1, y_1), (x_2, y_2)) = \lceil((x_1 - x_2)^2 + (y_1 - y_2)^2)^{1/2}\rceil$ –евклидово дискретное расстояние?

Источник [410, 141]. К этой задаче сводится задача ТП-3.

Комментарий. Задача *NP*-полна в сильном смысле. Она остаётся *NP*-полной в сильном смысле даже в том случае, если расстояние заменить метрикой типа L_1 [141] (или метрикой L_∞, которая после поворота на 45° становится эквивалентной метрике L_1). Задача остаётся *NP*-трудной в сильном смысле, если принять обычную (не дискретную) евклидову метрику, но в этом случае принадлежность этой задачи классу *NP* не установлена [141].

[ПС 24] **ЗАДАЧА КОММИВОЯЖЕРА С ОГРАНИЧЕННЫМИ РАССТОЯНИЯМИ**

УСЛОВИЕ. Заданы множество C, состоящее из m городов, расстояние $d(c_i, c_j) \in Z^+$ между каждой парой городов $c_i, c_j \in C$ и положительное целое число B.

ВОПРОС. Существует ли маршрут, проходящий через все города и не содержащий рёбер длины, большей B? Т.е. существует ли такая перестановка городов $< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$, что $d(c_{\pi(i)}, c_{\pi(i+1)}) \leqslant B$ при $1 \leqslant i \leqslant m$ и $d(c_{\pi(m)}, c_{\pi(1)}) \leqslant B$?

Источник. К данной задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся *NP*-полной, даже если $d(c_i, c_j) \in \{1, 2\}$ для всех $c_i, c_j \in C$. Важный частный случай задачи, разрешимый за полиномиальное время, описан в работе [172].

[ПС 25] **КИТАЙСКИЙ ПОЧТАЛЬОН ДЛЯ СМЕШАННЫХ ГРАФОВ**

УСЛОВИЕ. Заданы смешанный граф $G = (V, A, E)$, где A – множество ориентированных рёбер, E – множество неориентированных рёбер в V; длина $l(e) \in Z_0^+$ каждого ребра $e \in A \cup E$, граница $B \in Z^+$.

ВОПРОС. Существует ли в графе G цикл длины не более B, включающий по крайней мере один раз каждое ориентированное и каждое неориентированное ребро, причём ориентированные ребра должны входить в цикл только с правильной ориентацией?

Источник [480]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если длины всех рёбер равны между собой, граф G планарен и максимум степеней вершин в нём равен 3. Если либо A, либо E пусто (т. е. G – либо ориентированный, либо неориентированный граф), то задача может быть решена за полиномиальное время [109].

[ПС 26] **ЗАДАЧА О ПОДЪЕМНОМ КРАНЕ**

УСЛОВИЕ. Заданы смешанный граф $G = (V, A, E)$, длина $l(a) \in Z_0^+$ каждого элемента $e \in A \cup E$, граница $B \in Z^+$.

ВОПРОС. Существует ли в графе G цикл длины не более B, включающий каждое ориентированное ребро из A по крайней мере один раз и проходящий их только в правильном направлении?

Источник [130]. К этой задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если длины всех рёбер равны 1. Аналогичная задача о пути (с заранее выбранными концами или без них) также *NP*-полна.

[ПС 27] **СЕЛЬСКИЙ ПОЧТАЛЬОН**

УСЛОВИЕ. Заданы граф $G = (V, E)$, длина $l(e) \in Z_0^+$ каждого ребра $e \in E$, подмножество $E' \in E$ и граница $B \in Z_0^+$.

ВОПРОС. Существует ли в G цикл, включающий каждое ребро из E' и имеющий длину не более B?

Источник [335]. К данной задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $l(e) = 1$ для всех $e \in E$. То же самое верно для ориентированных графов.

[ПС 28] **САМЫЙ ДЛИННЫЙ ЦИКЛ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, длина $l(e) \in Z^+$ каждого ребра $e \in E$, положительное целое число K.

ВОПРОС. Существует ли в G цикл длины не менее K (т. е. такой цикл, сумма длин рёбер которого не менее K)?

Источник. К этой задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Задача остаётся *NP*-полной и в том случае, если $l(e) = 1$ для всех $e \in E$, То же самое верно и для ориентированных циклов в ориентированных графах. Если G – ориентированный граф турнира и $l(e) = 1$ для всех $e \in E$, то задача может быть решена за полиномиальное время [393]. Аналогичные задачи для ориентированных и неориентированных графов, в которых ищется цикл длины, не превосходящей K, могут быть решены за полиномиальное время (см., например, [254]), но если допускаются ребра отрицательной длины, то задача *NP*-полна.

[ПС 29] **САМЫЙ ДЛИННЫЙ ПУТЬ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, длина $l(e) \in Z^+$ каждого ребра $e \in E$, положительное целое число K, выделенные вершины $s, t \in V$.

ВОПРОС. Существует ли в графе G простой путь длины, не меньшей K (т. е. путь, у которого сумма длин рёбер не меньше K)?

Источник. К этой задаче сводится ГАМИЛЬТОНОВ ПУТЬ МЕЖДУ ДВУМЯ ВЕРШИНАМИ.

Комментарий. Задача остаётся *NP*-полной и тогда, когда $l(e) = 1$ для всех $e \in E$. То же самое верно для аналогичной задачи отыскания ориентированного пути в ориентированных графах. Для ациклических графов задача в общем виде может быть решена за полиномиальное время [324]. Аналогичные задачи об ориентированном и неориентированном «кратчайшем пути» для произвольных графов могут быть решены за полиномиальное время (см., например, [324]), но становятся *NP*-полными, если допускаются ребра отрицательной длины.

[ПС 30] **КРАТЧАЙШИЙ ПУТЬ С ОГРАНИЧЕНИЕМ ПО ВЕСУ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с двумя выделенными вершинами $s, t \in V$, длина $l(e) \in Z^+$, вес $w(e) \in Z^+$ каждого ребра $e \in E$ и положительные целые числа K и W.

ВОПРОС. Существует ли в G простой путь из s в t, веса не более W и длины не более K?

Источник [380]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Для ориентированных графов задача также *NP*-полна. Обе задачи разрешимы за полиномиальное время, если все веса или все длины равны между собой.

[ПС 31] K **КРАТЧАЙШИХ ПУТЕЙ (*)**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с двумя выделенными вершинами $s, t \in V$, длина $l(e) \in Z^+$ каждого ребра $e \in E$ и положительные целые числа B и K.

ВОПРОС. Существует ли в G K различных путей от s до t, веса которых не превосходят B?

Источник [260]. К этой задаче сводится по Тьюрингу ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Принадлежность этой задачи классу *NP* не установлена. Аналогичная задача о кратчайших циклах также *NP*-трудна. Обе задачи остаются *NP*-трудными и тогда, когда $l(e) = 1$ для всех $e \in E$. То же самое верно для аналогичных задач в ориентированных графах. Однако все варианты этой задачи могут быть решены за псевдополиномиальное время (т. е. время, ограниченное полиномом от $|V|$, K и $\log B$) и, следовательно, за полиномиальное время при любом фиксированном значении числа K. Соответствующие перечислительные задачи KP-полны.

А2.4. Потоковые задачи

[ПС 32] **ПОТОК МИНИМАЛЬНОЙ РЁБЕРНОЙ СТОИМОСТИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ с двумя выделенными вершинами s и t, пропускная способность $c(a) \in Z^+$ и стоимость $p(a) \in Z_0^+$ каждой дуги $a \in A$, потребность $R \in Z^+$, граница $B \in Z^+$.

ВОПРОС. Существует ли потоковая функция $f : A \to Z_0^+$, обладающая следующими свойствами:

(1) $f(a) \leqslant c(a)$ для всех $a \in A$;

(2) $\sum_{(u,v)\in A} f((u, v)) = \sum_{(u,v)\in A} f((v, u))$ для всех $v \in V \setminus \{s, t\}$ (т. е. во всех вершинах $v \in V \setminus \{s, t\}$ поток сохраняется);

(3) $\sum_{u,t\in A} f((u, t)) - \sum_{t,u\in A} f((t, u)) \geqslant R$ (т. е. в вершину t втекает поток величины не меньшеR);

(4) $\sum_{a\in A'} p(a) \leqslant B$, где $A' = \{a \in A : f(a) \neq 0\}$?

Источник [115]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной и тогда, когда $c(a) = 2$ и $p(a) \in \{0, 1\}$ для всех $a \in A$. Она разрешима за полиномиальное время, если $c(a) = 1$ для всех $a \in A$ [115] или если условие (4) заменить условием $\sum_{a\in A} p(a) \cdot f(a) \leqslant B$ (см., например, [324]). Однако задача остаётся *NP*-полной, если условие (4) заменить условием $\sum_{a\in A}(p_1(a)f(a)^2 + p_2(a)) \leqslant B$ [208].

[ПС 33] **ЦЕЛОЧИСЛЕННЫЙ ПОТОК В СЕТИ С УМНОЖИТЕЛЯМИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$, выделенные две вершины s и t, заданы «умножитель» $h(v) \in Z^+$ для каждой вершины $v \in V \setminus \{s,t\}$, пропускная способность $c(a) \in Z^+$ каждой дуги $a \in A$ и потребность $R \in Z^+$.

ВОПРОС. Существует ли потоковая функция $f : A \to Z_0$, такая что выполнены условия:

(1) $f(a) \leqslant c(a)$ для всех $a \in A$;

(2) для вершин $v \in V \setminus \{s,t\}$ выполнено соотношение

$$\sum_{(u,v)\in A} h(v) \cdot f((u,v)) = \sum_{(u,v)\in A} f((u,v));$$

(3) в вершину t притекает поток, не меньший R?

Источник [463]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Если $h(v) = 1$ для всех $v \in V \setminus \{s,t\}$, то она может быть решена за полиномиальное время стандартными методами теории потоков в сетях. Аналогичная задача, в которой допускаются нецелочисленные потоки, может быть решена методом линейного программирования.

[ПС 34] **ПОТОК В СЕТИ С ОГРАНИЧЕНИЯМИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ с двумя выделенными вершинами s и t, пропускная способность $c(a) \in Z^+$ каждой дуги $a \in A$, набор P ориентированных путей в G и потребность $R \in Z^+$.

ВОПРОС. Существует ли такая функция $g : P \to Z_0^+$, что если $f : A \to Z_0^+$ – потоковая функция, задаваемая соотношением $f(a) = \sum_{p\in P(a)} g(p)$ (где $P(a) \subseteq P$ – множество всех путей из P, содержащих дугу a), то f удовлетворяет условиям:

(1) $f(a) \leqslant c(a)$ для всех $a \in A$;

(2) в каждой вершине $v \in V \setminus \{s,t\}$ поток сохраняется;

(3) в вершину t втекает поток, не меньший R?

Источник [436]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, если все $c(a)$ равны 1. Аналогичная задача для нецелочисленных потоков эквивалентна задаче ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ, но вопрос о не совпадении наилучшего рационального и наилучшего целочисленного потоков является *NP*-полным.

[ПС 35] **ЦЕЛОЧИСЛЕННЫЙ ПОТОК В СЕТИ С ГОМОЛОГИЧНЫМИ ДУГАМИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ с въделенными двумя вершинами s и t, пропускная способность $c(a) \in Z^+$ каждой дуги

$a \in A'$, потребность $R \in Z^+$ и множество $H \in A \times A$ «гомологичных» пар дуг.

ВОПРОС. Существует ли такая потоковая функция $f : A \to Z_0^+$, что:

(1) $f(a) \leqslant c(a)$ для всех $a \in A$;

(2) во всех вершинах $v \in V \setminus \{s, t\}$ поток сохраняется;

(3) $f(a) = f(a')$ для всех пар гомологичных дуг $< a, a' > \in H$;

(4) в вершину t втекает поток, не меньший R?

Источник [463]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной и в том случае, если $c(a) = 1$ для всех $a \in A$. (Это можно доказать с помощью модификации конструкции из работы [114].) Аналогичная задача с нецелочисленными потоками полиномиально эквивалентна задаче ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ [251].

[ПС 36] **ЦЕЛОЧИСЛЕННЫЙ ПОТОК В СЕТИ С РАССЛОЕНИЯМИ**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ с выделенными вершинами s и t и такие «расслоения» $I_1, I_2, \ldots, I_k \subseteq A$, что $\bigcup_{1 \leqslant j \leqslant k} I_j = A$, пропускные способности расслоений $c_1, c_2, \ldots, c_k \in Z^+$, потребность $R \in Z^+$.

ВОПРОС. Существует ли такая потоковая функция $f : A \to Z_0^+$, что выполнены следующие условия:

(1) $\sum_{a \in I_j} f(a) \leqslant c_j$, для всех j, $1 \leqslant j \leqslant k$;

(2) в каждой вершине $v \in V \setminus \{s, t\}$ поток сохраняется;

(3) в вершину t втекает поток, не меньший R?

Источник [463]. К этой задаче сводится НЕЗАВИСИМОЕ МНОЖЕСТВО.

Комментарий. Если все пропускные способности равны 1 и каждое расслоение включает только две дуги, то задача остаётся *NP*-полной. Аналогичная задача для нецелочисленных потоков может быть решена с помощью линейного программирования.

[ПС 37] **НЕОРИЕНТИРОВАННЫЙ ПОТОК, ОГРАНИЧЕННЫЙ СНИЗУ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с выделенными вершинами s, t, пропускная способность $c(e) \in Z^+$ и нижняя граница $l(e) \in Z_0^+$ для каждого ребра $einE$, потребность $R \in Z^+$.

ВОПРОС. Существует ли такая потоковая функция

$$f : \{(u, v), (v, u) : \{u, v\} \in E\} \to Z_0^+,$$

что выполняются следующие условия:

(1) для любого $\{u, v\} \in E$ либо $f((u, v)) = 0$, либо $f((v, u)) = 0$

(2) $l(e) \leqslant \max\{f((u, v)), f((v, u))\} \leqslant c(e)$ для всех $e = \{u, v\} \in E$;

(3) во всех вершинах $vinV \setminus \{s, t\}$ поток сохраняется;

(4) в вершину t втекает поток, не меньший R?

Источник [251]. К этой задаче сводится ВЫПОЛНИМОСТЬ.

Комментарий. Задача NP-полна в сильном смысле даже в случае нецелочисленных потоков. Аналогичная задача для ориентированных графов может быть решена за полиномиальное время даже в том случае, если вместо потока, не меньшего R, искать поток, не больший R [125] (см. также [324]). Аналогичная задача ОРИЕНТИРОВАННЫЙ M-ПРОДУКТОВЫЙ ПОТОК, ОГРАНИЧЕННЫЙ СНИЗУ при всех $M \geqslant 2$ и нецелочисленных потоках, полиномиально эквивалентна задаче ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ [251].

[ПС 38] **ОРИЕНТИРОВАННЫЙ ДВУХПРОДУКТОВЫЙ ЦЕЛОЧИСЛЕННЫЙ ПОТОК**

УСЛОВИЕ. Заданы ориентированный граф $G = (V, A)$ с выделенными вершинами s_1, s_2, t_1 и t_2, пропускная способность $c(a) \in Z^+$ каждой дуги $a \in A$, потребности $R_1, R_2 \in Z^+$.

ВОПРОС. Существуют ли такие две потоковые функции $f_1, f_2 : AtoZ_0^+$, что

(1) $f_1(a) + f_2(a) \leqslant c(a)$ для всех $a \in A$;

(2) поток $f_i(i \in \{1, 2\})$ сохраняется во всех вершинах $v \in V \setminus \{s_i, t_i\}$;

(3) для $i \in \{1, 2\}$ поток f_i, втекающий в вершину t_i, не меньше R_i?

Источник [114]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся NP-полной даже в том случае, если $c(a) = 1$ для всех $a \in A$ и $R_1 = 1$. Вариант задачи, в котором $s_1 = s_2$, $t_1 = t_2$ и по дугам может передаваться только один конкретный продукт (для каждой дуги свой), также является NP-полным (это следует из работы [114]). Аналогичная M-продуктовая задача с нецелочисленными потоками при всех $M \geqslant 2$ полиномиально эквивалентна задаче ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ [251].

[ПС 39] **НЕОРИЕНТИРОВАННЫЙ ДВУХПРОДУКТОВЫЙ ЦЕЛОЧИСЛЕННЫЙ ПОТОК**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с выделенными вершинами s_1, s_2, t_1 и t_2, пропускная способность $c(e) \in Z^+$ каждой дуги $e \in E$, потребности $R_1, R_2 \in Z^+$.

ВОПРОС. Существуют ли такие две потоковые функции

$f_1, f_2 : \{(u, v), (v, u) : \{u, v\}\} \to Z_0^+$, что выполняются следующие условия:

(1) для любого $\{u, v\} \in E$ и $i \in \{1, 2\}$ либо $f_i((u, v)) = 0$, либо $f_i((v, u)) = 0$;

(2) для всех $\{u, v\} \in E$ выполнено соотношение $\max\{f_1((u, v)), f_1(v, u))\} + \max\{f_2((u, v)), f2((v, u))\} \leqslant c(\{u, v\})$;

(3) поток $f_i(i \in \{1, 2\})$ сохраняется во всех вершинах $v \in V \setminus \{s_i, t_i\}$;

(4) для $i \in \{1, 2\}$ поток f_i, втекающий в вершину t_i, не меньше R_i.

Источник [114]. К этой задаче сводится ОРИЕНТИРОВАННЫЙ ДВУХПРОДУКТОВЫЙ ЦЕЛОЧИСЛЕННЫЙ ПОТОК.

Комментарий. Задача остаётся NP-полной даже в том случае, если $c(e) = 1$ для всех $e \in E$. Она разрешима за полиномиальное время, если $c(e)$ чётно для всех $e \in E$. Аналогичная задача для нецелочисленных потоков может быть решена за полиномиальное время.

[ПС 40] **НЕПЕРЕСЕКАЮЩИЕСЯ СОЕДИНЯЮЩИЕ ПУТИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и набор непересекающихся пар вершин $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$.

ВОПРОС. Верно ли, что в G содержится k путей, попарно не имеющих общих вершин и соединяющих s_i, и t_i (для всех i, $1 \leqslant i \leqslant k$)?

Источник [269, 281, 366]. К этой задаче сводится 3-ВЫП.

Комментарий. Для планарных графов задача остаётся *NP*-полной [366]. Для любого фиксированного $k \geqslant 2$ сложность задачи остаётся открытой, но для планарных или транзитивных графов при $k = 2$ задача разрешима за полиномиальное время [424]. (Для общей задачи при $k = 2$ полиномиальный алгоритм был анонсирован в работе [487].) Аналогичная задача для планарных или ациклических ориентированных графов G при $k = 2$ разрешима за полиномиальное время, а для общих ориентированных графов *NP*-полна [424].

[ПС 41] **МАКСИМАЛЬНОЕ ЧИСЛО ОГРАНИЧЕННЫХ НЕПЕРЕСЕКАЮЩИХСЯ ПУТЕЙ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ с выделенными вершинами s и t, а также положительные целые числа J и $K (J, K \leqslant |V|)$.

ВОПРОС. Верно ли, что в графе G содержится не менее J путей из s в t, попарно не имеющих общих вершин и включающих не более K рёбер?

Источник [252]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной для любого фиксированного $K \geqslant 5$. При $K \leqslant 4$ она разрешима за полиномиальное время. Задача, в которой ищутся пути, не пересекающиеся только по рёбрам, *NP*-полна при всех фиксированных $K \geqslant 5$, она разрешима за полиномиальное время для $K \leqslant 3$ и открыта для $K = 4$. Те же самые результаты имеют место и в том случае, если граф и искомые пути ориентированы. Задача отыскания максимального числа непересекающихся путей из s в t без ограничения на их длину стандартными потоковыми методами разрешается за полиномиальное время (как в случае путей, не пересекающихся по вершинам, так и в случае путей, не пересекающихся по рёбрам).

[ПС 42] **МАКСИМАЛЬНОЕ ЧИСЛО НЕПЕРЕСЕКАЮЩИХСЯ ПУТЕЙ ФИКСИРОВАННОЙ ДЛИНЫ**

УСЛОВИЕ. Задан граф $G = (V, E)$ с выделенными вершинами s и t и заданы положительные целые числа J и $K (J, K \leqslant |V|)$.

ВОПРОС. Верно ли, что в G содержится не менее J путей из s в t, попарно не пересекающихся по вершинам и включающих ровно K рёбер?

Источник [252]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной для всех $K \geqslant 4$. При $K \leqslant 3$ она разрешима за полиномиальное время. Аналогичная задача для путей, не пересекающихся по рёбрам, *NP*-полна для любого фиксированного $K \geqslant 4$, задача разрешима за полиномиальное время при $K \leqslant 2$ и открыта при

$K = 3$. Эти же результаты имеют место для ориентированных графов и путей, с той лишь разницей, что вариант задачи, в котором ищутся пути, не пересекающиеся по дугам, разрешается за полиномиальное время при $K = 3$ и открыт при $K = 4$.

А2.5. Разное

[ПС 43] **КВАДРАТИЧНАЯ ЗАДАЧА О НАЗНАЧЕНИИ**

УСЛОВИЕ. Заданы неотрицательные целочисленные стоимости $c_{ij}, 1 \leqslant i, j \leqslant n$, расстояния $d_{kl}, 1 \leqslant k, l \leqslant m$, и граница $B \in Z^+$.

ВОПРОС. Существует ли такое однозначное отображение $f : \{1, 2, \dots, n\} \to \{1, 2, \dots, m\}$, что

$$\sum_{i=1}^{n} \sum j = 1, j \neq i^n c_{ij} d_{f(j)} \leqslant B?$$

Источник [469]. К этой задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

Комментарий. Частный случай этой задачи, когда $d_{kl} = k - l$, $c_{ij} = c_{ji}$ и $c_{ij} \in \{0, 1\}$, называется задачей ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ и является *NP*-полной. Общая задача обсуждается, например, в книге [161].

[ПС 44] **МИНИМИЗАЦИЯ ЧИСЛА ФИКТИВНЫХ РАБОТ В СЕТИ ПЕРТ**

УСЛОВИЕ. Заданы ациклический граф $G = (V, A)$ (в котором вершины представляют задания, а дуги – ограничения предшествования) и положительное целое число $K \leqslant |V|$.

ВОПРОС. Существует ли сеть типа ПЕРТ, соответствующая графу G и имеющая не более K фиктивных дуг? Иными словами, существует ли такой ориентированный ациклический граф $G' = (V, A')$ (где $V' = \{v_i^-, v_i^+ : v \in V\}$ и $\{(v_i^-, v_i^+ : v_i \in V\} \subseteq A'\})$, что $|A'| \leqslant |V| + K$ и в G' существует путь от $v_i^=$ к v_i^- тогда и только тогда, когда в G существует путь от v_i к v_j?

Источник [307]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

[ПС 45] **ТРИАНГУЛЯЦИЯ С ОГРАНИЧЕНИЕМ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, координаты $x(v), y((v) \in Z$ каждой вершины $v \in V$.

ВОПРОС. Существует ли такое подмножество $E' \in E$, что множество прямолинейных отрезков $\{[(x(u), y(u)), (x(v), y(v))]\} : \{u, v\} \in E'$ на плоскости представляет собой триангуляцию множества точек $\{(x(v), y(v)) : v \in V\}$?

Источник [359].

Комментарий. Задача является *NP*-полной в сильном смысле.

[ПС 46] **ГРАФ ПЕРЕСЕЧЕНИЯ ОТРЕЗКОВ РЕШЁТКИ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительные целые числа M, N.

ВОПРОС. Верно ли, что G представляет собой граф пересечения множества прямолинейных отрезков решетки $M \times N$? Иначе говоря, существует ли такое взаимно-однозначное отображение f множества вершин V в множество прямолинейных отрезков плоскости с концами (x, y) и (z, w) (где $1 \leqslant x \leqslant z \leqslant M$, $1 \leqslant y \leqslant w \leqslant N$ и либо $x = z$, либо $y = w$), что $\{u, v\} \in E$ тогда и только тогда, когда отрезки $f(u)$ и $f(v)$ пересекаются?

Источник [168]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Аналогичная задача, в которой спрашивается: «Верно ли, что G представляет собой граф пересечения прямоугольников решетки $M \times N$» также является *NP*-полной [168],

[ПС 47] **ВЛОЖЕНИЕ РЁБЕР В РЕШЁТКУ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительные целые числа M и N.

ВОПРОС. Существует ли такое взаимно-однозначное отображение $f : V \to \{1, 2, \dots, M\} \times \{1, 2, \dots, N\}$, что если $\{u, v\} \in E$, $f(u) = (x_1, y_1)$ и $f(v) = (x_2, y_2)$, то либо $x_1 = x_2$, либо $y_1 = y_2$ (т. е. $f(u)$ и $f(v)$ расположены на общей для них «линии» решетки) ?

Источник [168]. К этой задаче сводится 3-РАЗБИЕНИЕ.

[ПС 48] **ГЕОМЕТРИЧЕСКИ СВЯЗАННОЕ ДОМИНИРУЮЩЕЕ МНОЖЕСТВО**

УСЛОВИЕ. Заданы множество точек на плоскости $P = Z \times Z$, положительные целые числа B и K.

ВОПРОС. Существует ли такое подмножество $P' \subseteq P$, что $|P'| \leqslant K$, каждая точка множества $P \setminus P'$ находится на евклидовом расстоянии, не превосходящем B, от некоторой точки множества P' и граф $G = (P', E)$ связен (в графе G две вершины соединены ребром тогда и только тогда, когда евклидово расстояние между ними не превосходит B)?

Источник [344]. К этой задаче сводится ПЛАНАРНАЯ 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной и в том случае, если евклидова метрика заменяется метрикой типа L_1 или L_∞ [152].

[ПС 49] **МИНИМАЛЬНОЕ ВРЕМЯ ПЕРЕДАЧИ СООБЩЕНИЯ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, подмножество $V_0 \subseteq V$ и положительное целое число K.

ВОПРОС. Можно ли за время, не превосходящее K, «передать» сообщение от базисного множества V_0 ко всем другим вершинам? Иначе говоря, существует ли такая последовательность $V_0, E_1, V_1, E_2, \dots, E_k, V_k$, что $V_i \subseteq V, E_i \subseteq E, V_K = V$ и для всех $i, 1 \leqslant i \leqslant K$, выполнены условия: (1) ровно один конец каждого ребра из множества E_i принадлежит множеству V_{i-1} (2) никакие два ребра из множества E_i не имеют общего конца и (3) $V_i = V_{i-1} \cup \{v : \{u, v\} \in E_i\}$?

Источник [152]. К этой задаче сводится 3-С. Дальнейшие сведения об этой задаче можно найти в работе [121].

Комментарий. При любом фиксированном $K \geqslant 4$ задача *NP*-полна, а при $K = 1$ решается за полиномиальное время с помощью метода паросочетаний. В случае когда $|V_0| = 1$, задача остаётся *NP*-полной, но если G – дерево, то она разрешима за полиномиальное время [83].

[ПС 50] **МИНИМАКСНОЕ МНОЖЕСТВО ЦЕНТРОВ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(v) \in Z_0^+$ каждой вершины $v \in V$, длина $l(e) \in Z_0^+$ каждого ребра $e \in E$, положительное целое число $K \leqslant |V|$ и положительное рациональное число B.

ВОПРОС. Существует ли такое множество P мощности K, «состоящее из точек на G» (точкой на G называется либо вершина графа G, либо точка на ребре $e \in E$, где e рассматривается как прямолинейный отрезок длины $l(e)$), что если $d(v)$ – длина кратчайшего пути от вершины v до ближайшей к ней точке из P, то $\max\{d(v) \cdot w(v) : v \in V\} \leqslant B$?

Источник [278]. К этой задаче сводится ДОМИНИРУЮЩЕЕ МНОЖЕСТВО.

Комментарий. Эта задача известна также под названием задачи о «p-центрах». Задача остаётся *NP*-полной и тогда, когда $w(v) = 1$ для всех $v \in V$ и $l(e) = 1$ для всех $e \in E$. Для любого фиксированного K и для произвольного K при условии, что G – дерево, задача разрешима за полиномиальное время [278]. Вариант этой задачи, в котором P – подмножество вершин, также *NP*-полон, но если K фиксировано или G – дерево, то разрешим за полиномиальное время [493].

[ПС 51] **МНОЖЕСТВО ЦЕНТРОВ С МИНИМАЛЬНОЙ СУММОЙ**

УСЛОВИЕ. Заданы граф $G = (V, E)$, вес $w(v) \in Z_0^+$ каждой вершины $v \in V$, длина $l(e) \in Z_0^+$ каждого ребра $e \in E$, положительное целое число $K \leqslant |V|$ и положительное рациональное число B.

ВОПРОС. Существует ли такое множество P, состоящее из K «точек на G», что если $d(v)$ – длина кратчайшего пути от v до ближайшей к ней точки из P, то $\sum_{v \in V} d(v) \cdot w(v) \leqslant B$?

Источник [279]. К этой задаче сводится ДОМИНИРУЮЩЕЕ МНОЖЕСТВО.

Комментарий. Эта задача известна также под названием задачи о «p-медиане». Можно показать, что условие $P \subseteq V$ не ограничивает общности задачи. Задача остаётся *NP*-полной, если $w(v) = 1$ для всех $v \in V$ и $l(e) = 1$ для всех $einE$. Если K фиксировано или если K произвольно, а G – дерево, то задача разрешима за полиномиальное время.

А3. МНОЖЕСТВА И РАЗБИЕНИЯ

А3.1. Покрытие, представители и расщепление

[МР 1] **ТРЁХМЕРНОЕ СОЧЕТАНИЕ (3-С)**

УСЛОВИЕ. Задано подмножество $M \subseteq W \times \mathrm{X} \times Y$ (где W,X, Y – непересекающиеся множества, содержащие по q элементов).

ВОПРОС. Верно ли, что в M содержится 3-сочетание? Иными словами, существует ли такое подмножество $M' \in M$ что $|M'| = q$ и никакие два элемента не имеют ни одной одинаковой координаты?

Источник [280]. К этой задаче сводится 3-ВЫП (см. разд. 3.1.2).

Комментарий. Задача остаётся *NP*-полной и в том случае, если M «совместимо по парам», т. е. для любых элементов a, b и c имеет место свойство: если существуют такие элементы w, x и y, что $(a, b, y) \in M$, $(a, x, c) \in M$ и $(w, b, c) \in M$, то $(a, b, c) \in M$ (это утверждение следует из доказательства теоремы 3.1.2). Задача остаётся *NP*-полной и тогда, когда ни один элемент не входит более чем в три тройки, однако если нет элементов, содержащихся более чем в двух тройках, то задача разрешима за полиномиальное время [152]. Близкая задача ПАРОСОЧЕТАНИЕ (в которой $M \subseteq W \times X$) также разрешима за полиномиальное время (см., например, [324]).

[МР 2] **ТОЧНОЕ ПОКРЫТИЕ 3-МНОЖЕСТВАМИ (ТП-3)**

УСЛОВИЕ. Заданы множество X (такое, что $|X| = 3q$), набор C подмножеств множества X, содержащих по 3 элемента.

ВОПРОС. Верно ли, что C содержит точное покрытие множества X, т. е. такой поднабор $C' \in C$, что любой элемент из X принадлежит ровно одному подмножеству семейства C?

Источник [280]. К этой задаче сводится 3-С.

Комментарий. Задача остаётся *NP*-полной и тогда, когда ни один элемент из X не принадлежит более чем трем подмножествам семейства C, но если ни один элемент не принадлежит более чем двум подмножествам семейства C, то задача разрешима за полиномиальное время [152]. Близкая задача ТОЧНОЕ ПОКРЫТИЕ 2-МНОЖЕСТВАМИ также разрешима за полиномиальное время (с помощью метода паросочетаний).

[МР 3] **УПАКОВКА МНОЖЕСТВ**

УСЛОВИЕ. Заданы набор C конечных множеств, положительное число $K \leqslant |C|$.

ВОПРОС. Верно ли, что множество C содержит по крайней мере K непересекающихся множеств?

Источник [280]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной, даже если $|c| \leqslant 3$ для всех $c \in C$. С помощью метода паросочетаний задача решается за полиномиальное время, когда $|c| \leqslant 2$ для всех $c \in C$.

[МР 4] **РАСЩЕПЛЕНИЕ МНОЖЕСТВА**

УСЛОВИЕ. Задан набор C подмножеств множества S.

ВОПРОС. Существует ли такое разбиение множества S на два подмножества, что ни одно подмножество из C не содержится целиком ни в S_1, ни в S_2?

Источник [360]. К этой задаче сводится 3-ВЫП ПРИ РАЗЛИЧНЫХ ЛИТЕРАЛАХ. Эта задача известна также под названием РАСКРАШИВАЕМОСТЬ ГИПЕРГРАФА В ДВА ЦВЕТА.

Комментарий. Задача остаётся *NP*-полной, даже когда $|c| \leqslant 3$ для всех $c \in C$. Если $|c| \leqslant 2$ для всех $c \in C$, (такая задача называется РАСКРАШИВАЕМОСТЬ ГРАФА В ДВА ЦВЕТА), то задача разрешима за полиномиальное время.

[MP 5] **МИНИМАЛЬНОЕ ПОКРЫТИЕ**

УСЛОВИЕ. Заданы набор C подмножеств конечного множества S, положительное целое число $K \leqslant |C|$.

ВОПРОС. Верно ли, что C содержит покрытие мощности не более K? Иными словами, существует ли такое подмножество $C' \in C$, что $|C'| \leqslant K$ (и любой элемент из S принадлежит по крайней мере одному подмножеству из C'?

Источник [280]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $|c| \leqslant 3$ для всех $c \in C$. С помощью метода паросочетаний задача решается за полиномиальное время, если $|c| \leqslant 2$ для всех $c \in C$.

[MP 6] **МИНИМАЛЬНОЕ МНОЖЕСТВО ТЕСТОВ**

УСЛОВИЕ. Заданы набор C подмножеств конечного множества S и положительное целое число $K \leqslant |C|$.

ВОПРОС. Существует ли такой поднабор $C' \in C$, что $|C'| \leqslant K$ и для каждой пары различных элементов $u, v \in C$ существует тест $C \in C'$, содержащий ровно один из них?

Источник [152]. К этой задаче сводится 3-С.

Комментарий. Задача остаётся *NP*-полной, если $|c| \leqslant 3$ для любого $c \in C^2$. Вариант этой задачи, когда C может содержать не только множества из C, но и их объединения, также *NP*-полон [247].

[MP 7] **БАЗИС СЕМЕЙСТВА МНОЖЕСТВ**

УСЛОВИЕ. Заданы набор C подмножеств конечного множества S и положительное целое число $K \leqslant |C|$.

ВОПРОС. Существует ли такой набор B подмножеств множества S, что $|B| = K$ и для каждого подмножества $c \in C$ в B существует такой поднабор, что объединение множеств этого поднабора совпадает с c?

Источник [499]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача *NP*-полна, если $|c| \leqslant 3$ для любого $c \in C$, и становится тривиальной, если $|c| \leqslant 2$ для каждого $c \in C$.

[MP 8] **МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ**

УСЛОВИЕ. Заданы набор C подмножеств конечного множества S и положительное целое число $K \leqslant |S|$.

ВОПРОС. Существует ли такое подмножество $S' \subseteq S$, что $|S'| \leqslant K$ и S' содержит по крайней мере один элемент каждого подмножества из C?

Источник [280]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если $|c| \leqslant 2$ для всех $c \in C$.

---

[2] Дж. К. Ленстра указал нам, что эта задача остаётся *NP*-лолной даже в том случае, когда $|c| \leqslant 2$ для любого $c \in C$ – *Дополнение авторов.*

[МР 9] **МОДЕЛЬ ПЕРЕСЕЧЕНИЯ МНОЖЕСТВ**

УСЛОВИЕ. Задана матрица $A = (a_{ij})$ порядка $n \times n$ с элементами из Z_0^+.

ВОПРОС. Существует ли такой набор множеств $C = \{C_1, C_2, \dots, C_n\}$, что для всех $i, j (1 \leqslant i, j \leqslant n)$ имеет место равенство $a_{ij} = C_i \cap C_j|$?

Источник [77]. К этой задаче сводится РАСКРАШИВАЕМОСТЬ ГРАФА В ТРИ ЦВЕТА.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $a_{ii} = 3$ для всех $i, 1 \leqslant i \leqslant m$ (и следовательно, все множества C_i имеют мощность, равную 3). Если $a_{ii} = 2$, то задача эквивалентна задаче о распознавании рёберного графа и, следовательно, разрешима за полиномиальное время (см., например, [196]).

[МР 10] **СРАВНИТЕЛЬНОЕ СОДЕРЖАНИЕ**

УСЛОВИЕ. Заданы два набора $R = \{R_1, R_2, \dots, R_k\}$ и $S = \{S_1, S_2, \dots, S_l\}$ подмножеств конечного множества X и для всех i и $j (1 \leqslant i \leqslant k,\ 1 \leqslant j \leqslant l)$ веса $w(R_i), w(S_j) \in Z^+$.

ВОПРОС. Существует ли такое подмножество $Y \subseteq X$, что

$$\sum_{Y \subseteq R_i} w(R_i) \geqslant \sum_{Y \subseteq S_j} w(S_j)?$$

Источник [429]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если все подмножества из наборов R и S имеют веса, равные 1 [152].

[МР 11] **ПЕРЕСЕЧЕНИЕ ТРЁХ МАТРОИДОВ**

УСЛОВИЕ. Заданы три матроида: (E, F_1), (E, F_2), (E, F_3) и положительное целое число $K \leqslant |E|$. (Матроидом (E, F) называется множество элементов E и такое непустое семейство F подмножеств множества E, что выполнено два условия: (1) если $S \in F$, то и любое подмножество множества S принадлежит F, и (2) если для двух множеств $S, S' \in F$ выполнено соотношение $|S| = |S'| + 1$, то найдётся такой элемент $e \in S \setminus S'$, что $(S' \cup \{e\}) \in F$.)

ВОПРОС. Существует ли такое подмножество $E' \in E$, что $|E'| = K$ и $E' \in F_1 \cup F_2 \cup F_3$?

Источник. К этой задаче сводится 3-С.

Комментарий. Аналогичная задача ПЕРЕСЕЧЕНИЕ ДВУХ МАТРОИДОВ может быть решена за полиномиальное время даже в том случае, если матроиды заданы посредством полиномиальных алгоритмов, распознающих его составляющие. При этом можно допустить даже, чтобы элементы $e \in E$ имели веса $w(e) \in Z^+$, а цель заключается в отыскании такого множества $E^1 \in F_1 \cup F_2$, что его вес максимально возможный (см., например, [324]).

А3.2. Задачи о множествах с взвешенными элементами

[МР 12] **РАЗБИЕНИЕ**

УСЛОВИЕ. Заданы конечное множество A и размеры $s(a) \in Z^+$ всех элементов $a \in A$.

ВОПРОС. Существует ли такое подмножество $A' \subseteq A$, что

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

Источник [280]. К этой задаче сводится 3-С (см. разд. 3.1.5).

Комментарий. Задача остаётся *NP*-полной даже в том случае, если потребовать, чтобы $|A'| = |A|/2$, или если задано упорядочение $a_1, a_2, \dots, a_{2n}$ элементов множества A и требуется, чтобы A' содержало ровно один элемент из каждой пары $a_{2i-1}, a_{2i} (1 \leqslant i \leqslant n)$. Все эти задачи, однако, могут быть решены методом динамического программирования за псевдополиномиальное время (см. разд. 4.2).

[МР 13] **СУММА РАЗМЕРОВ**

УСЛОВИЕ. Заданы конечное множество A, размеры $s(a) = Z^+$ всех элементов $a \in A$ и положительное целое число B.

ВОПРОС. Существует ли такое подмножество $A' = A$, что сумма размеров его элементов равна B?

Источник [280]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача разрешима за псевдополиномиальное время (см. разд. 4.2).

[МР 14] **ПРОИЗВЕДЕНИЕ РАЗМЕРОВ**

УСЛОВИЕ. Заданы конечное множество A, размеры $s(a) \in Z^+$ всех элементов $a \in A$ и положительное целое число B.

ВОПРОС. Существует ли такое подмножество $A' = A$, что произведение размеров его элементов равно B?

Источник [538]. К этой задаче сводится ТП-3.

Комментарий[3]. Задача разрешима за псевдополиномиальное время от $|A|, \max\{s(a) : a \in A\}$.

[МР 15] **3-РАЗБИЕНИЕ**

УСЛОВИЕ. Заданы множество A, состоящее из $3m$ элементов, граница $B \in Z^+$ и такие размеры $s(a) \in Z^+$ всех элементов $a \in A$, что $B/4 < s(a) < B/2$ и $\sum_{a \in A} s(a) = mB$.

ВОПРОС. Можно ли множество A так разбить на m непересекающихся подмножеств $A_1, A_2, \dots, A_m$, что для любого $i (1 \leqslant i \leqslant m)$ выполнено соотношение $\sum_{a \in A_i} s(a) = B$ (заметим, что каждое множество A_i должно содержать ровно три элемента)?

Источник [145]. К этой задаче сводится 3-С (см. разд. 4.2).

Комментарий. Задача *NP*-полна в сильном смысле.

[МР 16] **ТРЁХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ**

[3]На неточность в комментарии к этой задаче, допущенную в первом издании, указал нам Т. Ибараки. – *Дополнение авторов.*

УСЛОВИЕ. Заданы непересекающиеся множества W, X и Y, содержащие по m элементов каждое, размер $s(a) \in Z^+$ каждого элемента $a \in W \cup X \cup Y$ и граница $B \in Z^+$.

ВОПРОС. Можно ли множество $W \cup X \cup Y$ так разбить на m непересекающихся множеств $A_1, A_2, \dots, A_m$, что каждое A_i содержит по одному элементу из каждого множества W, X и Y и для всех $i (1 \leqslant i \leqslant m)$ выполнены соотношения

$$\sum_{a \in A_i} s(a) = B?$$

Источник [152]. К этой задаче сводится 3-С (см. доказательство теоремы 4.4).

Комментарий. Задача *NP*-полна в сильном смысле.

[МР 17] **ПАРОСОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ**

УСЛОВИЕ. Заданы непересекающиеся множества X и Y, каждое из которых содержит m элементов, размеры $s(a) \in Z^+$ всех элементов $a \in X \cup Y$ и вектор ограничений $< B_1, B_2, \dots, B_m >$ с положительными целыми координатами.

ВОПРОС. Можно ли множество $X \cup Y$ так разбить на m непересекающихся множеств $A_1, A_2, \dots, A_m$, каждое из которых содержит по одному элементу из X и Y, что $\sum_{a \in A_i} s(a) \in B_i$ для всех $i (1 \leqslant i \leqslant m)$?

Источник. К этой задаче сводится ТРЁХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ.

Комментарий. Задача *NP*-полна в сильном смысле, но разрешима за полиномиальное время, если $B_1 = B_2 = \dots = B_m$.

[МР 18] **МАТЕМАТИЧЕСКОЕ ОЖИДАНИЕ СУММЫ КОМПОНЕНТ**

УСЛОВИЕ. Заданы набор C m-мерных векторов $v = (v_1, v_2, \dots, v_m)$ с целыми неотрицательными координатами, положительные целые числа K и B.

ВОПРОС. Можно ли так разбить C на непересекающиеся подмножества $C_1, C_2, \dots, C_K$, что

$$\sum_{i=1}^{K} \max_{1 \leqslant j \leqslant m} \left(\sum_{v \in C_i} v_j \right) \geqslant B$$

Источник [152]. К этой задаче сводится ТП-3. Эта задача была предложена в работе [534] и соответствует задаче об отыскании разбиения, максимизирующего математическое ожидание наибольшей из сумм компонент в предположении, что все множества разбиения равновероятны.

Комментарий. Задача *NP*-полна даже тогда, когда все компоненты векторов есть 0 или 1. Разрешима за полиномиальное время, если K фиксировано. Вариант этой задачи, в котором ищется разбиение на K непустых множеств, дающее сумму не более B, является *NP*-полным и тогда, когда $K = 3$ и все компоненты векторов есть 0 или 1.

[МР 19] **МИНИМУМ СУММЫ КВАДРАТОВ**

УСЛОВИЕ. Заданы конечное множество A, размер $s(a) \in Z^+$ каждого элемента $a \in A$, положительные целые числа K и J ($K \leqslant |A|$).

ВОПРОС. Можно ли множество A так разбить на K непересекающихся множеств $A_1, A_2, \dots, A_K$, что

$$\sum_{i=1}^{K} \left(\sum_{a \in A_i} s(a) \right)^2 \leqslant J?$$

Источник. К этой задаче сводится РАЗБИЕНИЕ или 3-РАЗБИЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле. Она *NP*-полна в обычном смысле и разрешима за псевдополиномиальное время при любом фиксированном K. Варианты этой задачи, в которых ограничение числа множеств K заменяется либо ограничением на максимальную мощность множеств, либо ограничением на максимум общего размера множеств, также *NP*-полны в сильном смысле [535]. Во всех случаях *NP*-полнота задачи сохраняется при замене показателя 2 на любое рациональное число $\alpha > 1$.

[МР 20] **K-е ПО ПОРЯДКУ ПОДМНОЖЕСТВО (*)**

УСЛОВИЕ. Заданы конечное множество A, размер $s(a) \in Z^+$ каждого элемента $a \in A$, положительные целые числа K и B.

ВОПРОС. Верно ли, что существует не менее K различных подмножеств $A' \in A$, сумма размеров элементов которых не превосходит B?

Источник [260]. К этой задаче сводится СУММА РАЗМЕРОВ.

Комментарий. Принадлежность этой задачи классу *NP* не установлена. Задача разрешима за псевдополиномиальное время (за время, ограниченное полиномом от K, $|A|$ и $In \sum s(a)$) [322]. Соответствующая задача перечисления KP-полна.

[МР 21] **ВЕКТОР K-й ПО ВЕЛИЧИНЕ (*)**

УСЛОВИЕ. Заданы подмножества $X_1, X_2, \dots, X_m \subseteq Z^+$, размер $s(x) \in Z^+$ каждого элемента $x \in X_i (1 \leqslant i \leqslant m)$ и положительные целые числа K, B.

ВОПРОС. Верно ли, что существует не менее K различных m-мерных векторов $(x_1, x_2, \dots, x_m) \in X_1 \times X_2 \times \dots \times X_m$, таких, что $\sum_{i=1}^{m} s(x_i) \geqslant B$?

Источник [261]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Принадлежность этой задачи классу *NP* не известна. Задача разрешима за полиномиальное время при фиксированном K, а в общем случае разрешима за псевдополиномиальное время (время, ограниченное полиномом от $K, \sum |X_i|$ и $ln \sum s(x)$). Соответствующая задача перечисления KP-полна.

А4. ХРАНЕНИЕ И ПОИСК ДАННЫХ

А4.1. Хранение данных

[ХП 1] **УПАКОВКА В КОНТЕЙНЕРЫ**

УСЛОВИЕ. Заданы конечное множество U предметов, размер $s(u) \in Z^+$ каждого предмета $u \in U$, положительное целое число B – вместимость контейнера, положительное целое число K.

ВОПРОС. Существует ли такое разбиение множества U на непересекающиеся подмножества $U_1, U_2, \dots, U_k$, что сумма размеров предметов из каждого подмножества u_i не превосходит B?

Источник К этой задаче сводятся РАЗБИЕНИЕ И 3-РАЗБИЕ-НИЕ.

Комментарий. Задача *NP*-полна в сильном смысле. Для каждого фиксированного $K \geqslant 2$ она *NP*-полна и разрешима за псевдополиномиальное время. Методом полного перебора при любом фиксированном B задача решается за полиномиальное время.

[ХП 2] **ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ**

УСЛОВИЕ. Заданы множество элементов данных A, размер $s(a) \in Z^+$ каждого элемента данных $a \in A$, время поступления $r(a) \in Z_0^+$ и время $d(a) \in Z^+$ окончания работы с элементом данных $a \in A$, положительное целое число D – размер области памяти.

ВОПРОС. Существует ли для множества элементов данных A допустимое распределение памяти? Иначе говоря, существует ли такая функция $\sigma : A \to \{1, 2, \dots, D\}$, что для каждого элемента $a \in A$ интервал

$$I(a) = [\sigma(a), \sigma(a) + s(a) - 1]$$

содержится в $[1, D]$, причём для любых $a, a' \in A$, если множество $I(a) \cap I(a')$ не пусто, то либо $d(a) \leqslant r(a')$, либо $d(a') \leqslant r(a)$?

Источник [501]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле даже в том случае, если $s(a) \in \{1, 2\}$ для всех $a \in A$. Если размеры всех элементов данных одинаковы, то задача решается за полиномиальное время алгоритмами раскраски интервальных графов (см., например, [163]).

[ХП 3] **МИНИМИЗАЦИЯ ПАМЯТИ ДЛЯ СТРУКТУРЫ ДАННЫХ «PRUNED TRIE»**[4]

УСЛОВИЕ. Заданы конечное множество S, семейство F функций вида $f : S \to Z^+$ и положительное целое число K.

ВОПРОС. Существует ли такая последовательность $f : S \to Z^+$ различных функции из F, что для каждой пары элементов $a, b \in S$ существует такое $i (1 \leqslant i \leqslant m)$, что $f_i(a) \neq f_i(b)$ и $\sum_{i=1}^{m} N(i) \leqslant K$ (где $N(i)$–число различных i-мерных векторов $X = (x_1, x_2, \dots, x_i)$, для которых имеются по крайней мере 2 элемента c и d, такие, что $(f_1(c), f_2(c), \dots, f_i(c)) = (f_i(d) f_2(d), ldots, f_i(d)) = X)$)?

Источник [87]. К этой задаче сводится 3-С.

Комментарий. Задача остаётся *NP*-полной и тогда, когда все $f \in F$ принимают значения в множестве $\{0, 1\}$. Варианты задачи, возникающие при замене структуры данных «pruned trie» структурой данных «full trie»,

[4]Термин «trie» восходит к работе E. Fredkin «Trie memory». Comm. of the ACM, 1960, 3, 9, и происходит, по-видимому, от слова reTRIEval. – *Прим. ред.*

«collapsed trie» или «pruned 0-trie», также *NP*-полны. Близкая задача «минимизация времени доступа» также *NP*-полна для задачи со структурой данных pruned trie, в которой ищется последовательность $< f_1, f_2, \dots, f_n >$ функций из F, различающих (как и выше) любые два элемента из S, и обладающая тем свойством, что $\sum_{a \in S} L(a) \leqslant K$, (где время доступа $L(a)$ к элементу $a \in S$ определяется как наименьшее из таких i, для которых не существует такого элемента $b \in S$, что последовательность $(f_1(b), f_2(b), \dots, f_i(b))$ совпадает с $(f_1(a), f_2(a), \dots, f_i(a)))$.

[ХП 4] **ОЖИДАЕМАЯ СТОИМОСТЬ ПОИСКА**

УСЛОВИЕ. Заданы множество R записей, рациональное число $p(r) \in [0, 1]$ – вероятность обращения к записи $r \in R$ (причём $\sum_{r \in R} p(r) = 1$), число сегментов m и положительное целое число K.

ВОПРОС. Существует ли разбиение множества R на такие непересекающиеся подмножества $R_1, R_2, \dots, R_m$, что сумма по всем упорядоченным парам $i, j (1 \leqslant i, j \leqslant m)$ величин $p(R_i) \cdot p(R_j) \cdot d(i, j)$ не превосходит K (где $p(R_i) = \sum_{r \in R_i} p(r)$ и стоимость перехода $d(i, j)$ равна $j - i - 1$, если $1 \leqslant i < j \leqslant m$, и равна $m - i + j - 1$, если $1 \leqslant j \leqslant i \leqslant m$)?

Источник [84]. К этой задаче сводятся РАЗБИЕНИЕ и 3-РАЗБИЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле. Для каждого фиксированного $m \geqslant 2$ она *NP*-полна и разрешима за псевдополиномиальное время.

[ХП 5] **ОРГАНИЗАЦИЯ ПАМЯТИ В ВИДЕ КОРНЕВОГО ДЕРЕВА**

УСЛОВИЕ. Заданы конечное множество X, набор $C = \{X_1, X_2, \dots, X_n\}$ подмножеств множества X, положительное целое число K.

ВОПРОС. Существует ли такой набор $C = \{X'_1, X'_2, \dots, X'_n)$ подмножеств множества X, что $X_i \subseteq X'_i$ при всех $i (1 \leqslant i \leqslant n)$, $\sum_{i=1}^{n} |X'_i - X_i| \leqslant K$ и существует ориентированное корневое дерево $T(x, A)$, в котором элементы каждого из подмножеств $X'_i, 1 \leqslant i \leqslant n$, образуют ориентированный путь?

Источник [168]. К этой задаче сводится ПРЕДСТАВЛЕНИЕ В ВИДЕ КОРНЕВОГО ДЕРЕВА.

[ХП 6] **РАЗМЕЩЕНИЕ ФАЙЛА С ДУБЛИРОВАНИЕМ**

УСЛОВИЕ. Задан граф $G = (V, E)$, для каждой вершины $v \in V$ заданы количество обращений $u(v) \in Z^+$ и стоимость запоминания $s(v) \in Z^+$, а также положительное целое число K.

ВОПРОС. Существует ли такое подмножество $V' \subseteq V$, что

$$\sum_{v \in V'} s(v) + \sum_{v \in V} d(v) \cdot u(v) \leqslant K?$$

(Здесь d(v) означает число рёбер кратчайшего пути в G, от v до множества V'.)

Источник [529]. К данной задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача *NP*-полна в сильном смысле даже в том случае, когда все вершины $v \in V$ имеют одинаковые величины $u(v)$ и одинаковые величины $s(v)$.

[ХП 7] **ВЫБОР ПРОПУСКНЫХ СПОСОБНОСТЕЙ**

УСЛОВИЕ. Заданы множество C линий связи, множество пропускных способностей $M \subseteq Z^+$, функция стоимости $g : C \times M \to Z^+$, функция $d : C \times M \to Z^+$ штрафа за задержку, такие, что для всех $c \in C$ и $i < j, i, j \in M$ выполнены неравенства $g(c, i) \leqslant g(c, j)$ и $d(c, j) \geqslant d(c, j)$, а также заданы положительные целые числа K и J.

ВОПРОС. Существует ли такой выбор пропускных способностей $\sigma : C \to M$, что общая стоимость $\sum_{c \in C} g(c, \sigma(c))$ не превосходит K, а общий штраф за задержку $\sum d(c, \sigma(c))$ не превосходит J?

Источник [529]. К этой задаче сводится СУММА РАЗМЕРОВ.

Комментарий. Задача разрешима за псевдополиномиальное время.

А4.2. Сжатие и представление информации

[ХП 8] **КРАТЧАЙШАЯ ОБЩАЯ НАДПОСЛЕДОВАТЕЛЬНОСТЬ**

УСЛОВИЕ. Заданы конечный алфавит Σ, конечное множество R слов в алфавите Σ и положительное целое число K.

ВОПРОС. Существует ли такое слово $w \in \Sigma^*$, что $|w| \leqslant K$ и каждое слово $x \in R$ есть подпоследовательность слова w (т. е. $w = w_0 x_1 w_1 x_2 w2 \ldots x_k w_k$, где $w_i \in \Sigma^*$, а $x = x_1, x_2, \ldots, x_k$?

Источник [371]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $|\Sigma| = 2$. Если $|R| = 2$, то она решается за полиномиальное время (для чего вначале вычисляется наибольшая общая подпоследовательность). То же самое верно, если $|x| \leqslant 2$ для всех $x \in R$.

[ХП 9] **КРАТЧАЙШЕЕ ОБЩЕЕ «НАДСЛОВО»**

УСЛОВИЕ. Заданы конечный алфавит Σ, конечное множество R слов из Σ^* и положительное целое число K.

ВОПРОС. Существует ли такое слово $w \in \Sigma^*$, что $|w| \leqslant K$ и любое слово $x \in R$ есть подслово w, т. е. $w = w_0 x w_1$, где $w_i \in \Sigma^*$?

Источник [372]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ для кубических графов.

Комментарий. Задача остаётся *NP*-полной и в том случае, если $|\Sigma| = 2$ или если $|x| \leqslant 8$ для всех слов $x \in R$ и все слова из R не содержат повторяющихся символов. Когда $|x| \leqslant 2$ для всех $x \in R$, то задача разрешима за полиномиальное время.

[ХП 10] **ОБЩАЯ ПОДПОСЛЕДОВАТЕЛЬНОСТЬ НАИБОЛЬШЕЙ ДЛИНЫ**

УСЛОВИЕ. Заданы конечный алфавит Σ, конечное множество R слов из Σ^* и положительное целое число K.

ВОПРОС. Существует ли такое слово $w \in \Sigma^*$, что $|w| \geqslant K$ и w – подпоследовательность каждого слова из R?

Источник [371]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $|\Sigma| = 2$. Если K или $|R|$ фиксировано, то задача решается за полиномиальное время (методом динамического программирования (см., например, [531]). Аналогичная задача ОБЩЕЕ ПОДСЛОВО НАИБОЛЬШЕЙ ДЛИНЫ тривиально решается за полиномиальное время.

[ХП 11] **ОГРАНИЧЕННОЕ СООТВЕТСТВИЕ СООБЩЕНИЙ**

УСЛОВИЕ. Заданы конечный алфавит Σ, две последовательности $a = (a_1, a_2, \dots, a_n)$ и $b = (b_1, b2, \dots, b_n)$ слов из Σ^* и положительное целое число $K \leqslant n$.

ВОПРОС. Существует ли такая последовательность $i_1, i_2, \dots, i_k$, состоящая из $k, k \in K$ (не обязательно различных), целых чисел из отрезка $[1, n]$, что слова $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ и $b_{i_1}, b_{i_2}, \dots, b_{i_k}$ совпадают?

Источник [88]. Общая сводимость[5].

Комментарий. Если на k не наложено верхнее ограничение, то задача неразрешима (см., например, [215]).

[ХП 12] **ПРЕДСТАВЛЯЮЩЕЕ СЛОВО**

УСЛОВИЕ. Задано конечное множество A слов в алфавите $\{0, 1, *\}$, имеющих длину, равную n.

ВОПРОС. Существует ли такое слово $x \in \{0, 1\}^*$, что $|x| = n$ и для каждого слова $a \in A$ найдётся такое $i, 1 \leqslant i \leqslant n$, что в словах x и a i-е символы совпадают?

Источник [120]. К этой задаче сводится 3-ВЫП.

[ХП 13] **СЖАТИЕ РАЗРЕЖЕННОЙ МАТРИЦЫ**

УСЛОВИЕ. Заданы $(m \times m)$-матрица A с элементами $a_{ij} \in \{0, 1\}$, $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, и положительное целое число $K \leqslant mn$.

ВОПРОС. Существует ли такая последовательность $(b_1, b_2, \dots, b_{n+K})$, состоящая из целых чисел $b_i, 1 \leqslant b_i \leqslant m$, и такая функция $s : \{1, 2, \dots, m\} \to \{1, 2, \dots, K\}$, что для всех i и $j (1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n) a_{ij} = 1$ тогда и только тогда, когда $b_{s(i)+j} = i$?

Источник [116]. К этой задаче сводится РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА.

Комментарий. Задача остаётся *NP*-полной при $K = 3$.

[ХП 14] **ПОДМАТРИЦА СО СВОЙСТВОМ СВЯЗАННОСТИ**

УСЛОВИЕ. Заданы $(m \times m)$-матрица A с элементами 0 и 1 и положительное целое число K.

ВОПРОС. Существует ли в матрице A подматрица B порядка $m \times K$, обладающая свойством «связности единиц» (т. е. подматрица B, столбцы которой можно так переставить, что в каждой строке все единицы идут подряд)?

---

[5] Имеется в виду сводимость, описанная при доказательстве теоремы Кука. – *Прим. ред.*

Источник [51]. К этой задаче сводится ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Вариант этой задачи, в котором ищется подматрица B, обладающая свойством «циклической связности», т.е. подматрица, столбцы которой можно переставить так, что в каждой строке либо все единицы, либо все нули идут подряд, также NP-полон. Обе задачи могут быть решены за полиномиальное время, если $K = n$ (т. е. если спрашивается: «Верно ли, что нужным свойством обладает сама матрица A?»). См., например, [132, 54].

[ХП 15] **РАЗБИЕНИЕ НА ПОДМАТРИЦЫ СО СВОЙСТВОМ связности**

УСЛОВИЕ. Задана (0-1)-матрица A порядка $m \times n$.

ВОПРОС. Можно ли так разбить строки матрицы на две подгруппы, чтобы получившиеся в результате матрицы порядков $m_1 \times n$ и $m_2 \times n (m_1 + m_2 = m)$ обладали свойством связности единиц?

Источник [352]. К этой задаче сводится ГАМИЛЬТОНОВ ПУТЬ в кубическом графе.

[ХП 16] **ДОПОЛНЕНИЕ ДО МАТРИЦЫ СО СВОЙСТВОМ связности**

УСЛОВИЕ. Заданы (0-1)-матрица A порядка $m \times n$ и положительное целое число K.

ВОПРОС. Существует ли матрица A', получаемая из матрицы A заменой в ней не более K нулевых элементов единицами, такая, что A обладает свойством связности единиц?

Источник [51,407]. К этой задаче сводится ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ.

Комментарий. Вариант задачи, когда ищется матрица A, обладающая свойством циклических единиц, также NP-полон.

[ХП 17] **МИНИМИЗАЦИЯ ЧИСЛА БЛОКОВ**

УСЛОВИЕ. Заданы (0-1)-матрица A порядка $m \times n$ и положительное целое число K.

ВОПРОС. Существует ли такая перестановка столбцов матрицы A, что получившаяся в результате матрица B имеет не более K блоков из последовательно идущих единиц, т. е. не более K таких пар индексов i, j, что $b_{ij} = 1$ и либо $b_{i,j+1} = 0$, либо $j = n$?

Источник [297]. К этой задаче сводится ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Если свойство $<< j = n >>$ заменить свойством «$j = n$ и $b_{i1} = 0$», то задача остаётся NP-полной [51]. Если K равно числу строк (матрицы A), содержащих хотя бы одну единицу, то эти задачи эквивалентны задачам о проверке наличия у матрицы A свойства связности единиц или циклической связности соответственно, и они могут быть решены за полиномиальное время.

[ХП 18] **ПРЕДСТАВЛЕНИЕ СО СВОЙСТВОМ СВЯЗНОСТИ**

УСЛОВИЕ.Заданы конечный алфавит Σ, набор $C = \{\Sigma_1, \Sigma_2, \dots, \Sigma_n\}$ подмножеств множества Σ и положительное целое число K.

ВОПРОС. Существует ли такое слово $w \in \Sigma^*$, что $|w| \leqslant K$ и для каждого i элементы множества Σ_i образуют в слове w подслово длины $|\Sigma_i|$?

Источник [297]. К данной задаче сводится ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Вариант задачи, в котором требуется только, чтобы элементы каждого множества $|sigma_i$ образовывали подслово длины $|\Sigma_i|$ в слове ww (т. е. допускаются подслова слова w), записанном по окружности, также NP-полон [51]. Если K равно числу различных символов из Σ_i, то эти задачи эквивалентны задачам выяснения того, обладает ли матрица свойством связности или циклической связности единиц соответственно, и разрешимы за полиномиальное время.

[ХП 19] **РАЗБИЕНИЕ СО СВОЙСТВОМ СВЯЗНОСТИ**

УСЛОВИЕ. Заданы конечный алфавит Σ и набор $C = \{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$ подмножеств из Σ.

ВОПРОС. Существует ли разбиение Σ на такие непересекающиеся множества $X_1, X_2, \ldots, X_k$, что каждое из X_i имеет не более одного общего элемента с каждым из Σ_j и для каждого $\Sigma_j \in C$ существует такой индекс $l(j)$, что множество Σ_j содержится в множестве

$$X_{l(j)} \cup X_{l(j)+1} \cup \ldots \cup X_{l(j)+|\Sigma_j|-1}?$$

Источник [350, 351]. К этой задаче сводится РАСКРАШИВАЕМОСТЬ ГРАФА В ТРИ ЦВЕТА.

Комментарий. Если $|\Sigma_j| \leqslant 5$ для всех $\Sigma_j \in C$, то задача остаётся NP-полной, но если $|\Sigma_j| \leqslant 2$ для всех $\Sigma_j \in C$, то она разрешима за полиномиальное время.

[ХП 20] **РЕДАКТИРОВАНИЕ СЛОВА**

УСЛОВИЕ. Заданы конечный алфавит Σ, два слова $x, y \in \Sigma^*$ и положительное целое число K.

ВОПРОС. Можно ли получить из слова x слово y, применив не более K операций вычеркивания символа и перестановки соседних символов?

Источник [530]. К этой задаче сводится ПОКРЫТИЕ МНОЖЕСТВА.

Комментарий. Задача разрешима за полиномиальное время, если в множество операций дополнительно включить операции замены и вставления отдельной буквы даже тогда, когда перестановка букв не допускается (см., например, [531]), или тогда, когда единственной операцией является операция перестановки соседних букв [530]. В указанных работах приводятся также и другие аналогичные результаты для задач, в которых различные операции имеют различную стоимость.

[ХП 21] **ГРУППИРОВКА С ПОМОЩЬЮ ТРАНСПОЗИЦИЙ**

УСЛОВИЕ. Заданы конечный алфавит Σ, слово $x \in \Sigma^*$ и положительное целое число K.

ВОПРОС. Существует ли последовательность, состоящая не более чем из K транспозиций соседних символов и превращающая слово x в слово y, у которого все вхождения каждого символа $a \in \Sigma$ образуют один блок (т. е. в слове y нет подпоследовательностей вида aba, где $a, b \in \Sigma$ и $a \neq b$)?

Источник [224]. К этой задаче сводится МНОЖЕСТВО ДУГ, РАЗРЕЗАЮЩИХ КОНТУРЫ.

[ХП 22] **СЖАТИЕ ВНЕШНИХ МАКРОДАННЫХ**

УСЛОВИЕ. Заданы конечный алфавит Σ, слово $s \in \Sigma^*$, стоимость указателя $h \in Z^+$ и граница $B \in Z^+$.

ВОПРОС. Существуют ли такие слова D (слово-словарь) и C (сжатое слово) в алфавите $(\Sigma \cup \{p_i : 1 \leqslant i \leqslant |s|\})$ (символы p_i называются «указателями»), что (1) $|D| + |C| + (h-1)\cdot$ (число вхождений указателей в слова D и C) $\leqslant B$ и (2) существует такое соответствие указателей с подсловами слова D, что слово S может быть получено из слова C последовательной заменой указателей из C соответствующими подсловами из D?

Источник [505, 506]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Если h – любое фиксированное целое число, не меньшее 2, то задача остаётся *NP*-полной. Многие её варианты, в том числе задачи, в которых D не содержит указателей и/ или если указатели соответствуют неперекрывающимся словам, также являются *NP*-полными. Если алфавит содержит фиксированное число букв (но не менее 3) и стоимость указателя равна $\lceil h \log |s| \rceil$, то задача также *NP*-полна. В упомянутых выше работах приводятся другие варианты задачи, в том числе задачи с «исходными указателями».

[ХП 23] **СЖАТИЕ ВНУТРЕННИХ МАКРОДАННЫХ**

УСЛОВИЕ. Заданы алфавит Σ, слово $s \in \Sigma^*$, стоимость $h \in Z^+$ указателя и граница $B \in Z^+$.

ВОПРОС. Существует ли такое слово $C(\Sigma \cup \{p_i : 1 \leqslant i \leqslant |s|\})^*$, что (1) $|C| + (h-1)\cdot$ (число вхождений указателей в слово C) $\leqslant B$ и (2) существует такое соответствие указателей с подсловами слова C, что слово s может быть получено из C при использовании C одновременно в качестве «словаря» и сжатого слова способом, изложенным в предыдущей задаче?

Источник [505, 506]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда h – любое фиксированное целое число, не меньшее 2. Другие *NP*-полные варианты этой задачи (как и для предыдущей) указаны в упомянутых работах.

[ХП 24] **ПОДСТАНОВКА РЕГУЛЯРНОГО ВЫРАЖЕНИЯ**

УСЛОВИЕ. Заданы два алфавита $X = \{x_1, x_2, \dots, x_n\}$ и $Y = \{y_1, y_2, \dots, y_m\}$, регулярное выражение R над $X \cup Y$, регулярные выражения $R_1, R_2, \dots, R_n$ над Y и слово $w \in Y^*$.

ВОПРОС. Существуют ли в языках, определяемых регулярными выражениями R и $R_i (1 \leqslant i \leqslant n)$, такие слова z и w_i соответственно, что если каждое слово w_i подставить в слово z вместо каждого вхождения символа x_i, то получившееся в результате слово будет совпадать с w?

Источник [15]. К этой задаче сводится ТП-3.

[ХП 25] **ПРЯМОУГОЛЬНОЕ СЖАТИЕ КАРТИНКИ**

УСЛОВИЕ. Заданы (0-1)-матрица M порядка $n \times n$ и положительное целое число K.

ВОПРОС. Можно ли покрыть все единичные элементы матрицы A не более чем K прямоугольниками? Иначе говоря, существует ли такая последовательность четвёрок $(a_i, b_i, c_i, d_i), 1 \leqslant i < K$. где $a_i \leqslant b_i$, $c_i \leqslant d_i$, $1 \leqslant i \leqslant K$, что для каждой пары индексов $(i, j), 1 \leqslant i, j \leqslant n, M_{ij}$ равно 1 тогда и только тогда, когда существует такое $k, 1 \leqslant k \leqslant K$, что $a_k \leqslant i \leqslant b_k$ и $c_k \leqslant j \leqslant d_k$?

Источник [374]. К этой задаче сводится 3-ВЫП.

А4.3. Задачи о базах данных

[ХП 26] **НАИМЕНЬШИЙ ПО МОЩНОСТИ КЛЮЧ**

УСЛОВИЕ. Заданы множество «имён атрибутов» A, семейство F упорядоченных пар подмножеств из A (называемых «функциональными зависимостями» на A) и положительное целое число M.

ВОПРОС. Существует ли для реляционной системы $< A, F >$ ключ мощности не более M? Иначе говоря, существует ли такое подмножество $K \subseteq A$, что $|K| \leqslant M$ и упорядоченная пара (K, A) принадлежит «замыканию» F^* семейства F? (Замыкание F^* семейства F определяется следующим образом: (1) $F \subseteq F^*$, (2) из того что $B \subseteq C \subseteq D$ следует, что $(C, B) \in F^*$, (3) из того что $(B, C), (B, D) \in F^*$, следует, что $(B, D) \in F^*$, и (4) из того, что $(B, C), (B, D) \in F^*$, следует, что $(B, C \cup D) \in F^*$.)

Источник [362, 350]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ. Общие основы баз данных с реляционными системами можно найти в книге [101].

[ХП 27] **ДОПОЛНИТЕЛЬНЫЙ КЛЮЧ**

УСЛОВИЕ. Заданы множество имён атрибутов A, семейство F функциональных зависимостей на A, подмножество $R \subseteq A$ и множество K ключей для реляционной схемы $< R, F >$.

ВОПРОС. Верно ли, что R имеет ключ, не содержащийся в K? Иначе говоря, верно ли, что существует такое подмножество $R' \subseteq R$, что $R' \notin K$, $(R', R) \in F^*$, и не существует такого подмножества $R'' \subseteq R'$, что $(R'', R) \in F^*$?

Источник [34]. К этой задаче сводится МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ.

[ХП 28] **ПРИМИТИВНОЕ ИМЯ АТРИБУТА**

УСЛОВИЕ. Заданы множество имён атрибутов A, семейство F функциональных зависимостей на A и выделенное имя атрибута $x \in A$.

ВОПРОС. Верно ли, что x является «примитивным именем атрибута» для $< A, F >$? Иными словами, существует ли такой ключ K для $< A, F >$, что $x \in K$?

Источник [362]. К этой задаче сводится НАИМЕНЬШИЙ ПО МОЩНОСТИ КЛЮЧ.

[ХП 29] **НАРУШЕНИЕ НОРМАЛЬНОЙ ФОРМЫ БОЙСА-КОДДА**

УСЛОВИЕ. Заданы множество имён атрибутов A, семейство F функциональных зависимостей на A и подмножество $A' \subseteq A$.

ВОПРОС. Верно ли, что множество A нарушает нормальную форму Бойса-Кодда для реляционной системы $< A, F >$? Иначе говоря, существуют ли подмножество $X \in A'$ и два имени атрибутов $y, z \in A' \setminus X$, такие, что $(X, \{y\}) \in F^*$ и $(X, \{z\}) \in F^*$ (где F^* – замыкание F)?

Источник [40,34]. К данной задаче сводится МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ.

Комментарий. Задача остаётся *NP*-полной и в том случае, если потребовать, чтобы A' имело «третью нормальную форму» (т.е. если $X \subseteq A'$ – ключ для реляционной системы $< A', F >$ и два имени атрибутов $y, z \in A' \setminus X$ таковы, что $(X, \{y\}) \in F^*$ и $(X, \{z\}) \in F^*$, то z – примитивное имя атрибута для реляционной системы $< A', F >$).

[ХП 30] **ПРЕОБРАЗУЕМОСТЬ КОНЪЮНКТИВНОГО ЗАПРОСА**

УСЛОВИЕ. Заданы конечный домен D, семейство отношений $R = \{R_1, R_2, \dots, R_m\}$ (где R_i есть подмножество множества d_i-мерных кортежей с элементами из D), множество выделенных переменных X, множество невыделенных переменных Y и два запроса Q_1 и Q_2 над X, Y, D и R; запрос Q имеет вид

$$(x_1, x_2, \dots, x_k)(\exists\, y_1, y_2, \dots, y_l)(A_1 \wedge A_2 \wedge \dots \wedge A_r)$$

для некоторых k, l и r, где $X' = \{x_1, x_2, \dots, x_k\} \subseteq X, Y' = \{y_1, y_2, \dots, y_l\} \subseteq Y$, и каждое A_i имеет вид $R_j(u_1, u_2, \dots, u_{d_j}, u_s \in D \cup X' \cup Y', 1 \leqslant s \leqslant d_j$; интерпретация таких выражений в терминах баз данных дана в работе [67].

ВОПРОС. Существует ли такая функция $\sigma : Y \to X \cup Y \cup D$, что если для каждого $y \in Y$ вместо любого вхождения y в Q_1 подставить символ $\sigma(y)$, то в результате получится запрос Q_2?

Источник [67]. К данной задаче сводится РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА.

Комментарий. Задача изоморфизма конъюнктивных запросов (два запроса называются изоморфными, если они совпадают с точностью до взаимно-однозначного переименования переменных, переупорядочения конъюнктивных членов и переупорядочения кванторов) полиномиально эквивалентна задаче об изоморфизме графов.

[ХП 31] **КОНЪЮНКТИВНЫЙ БУЛЕВСКИЙ ЗАПРОС**

УСЛОВИЕ. Заданы домен D, набор отношений $R = \{R_1, R_2, \dots, R_m\}$ (где R_i есть подмножество множества D_i^d), конъюнктивный булевский запрос Q над R и D, т. е. запрос вида

$$(\exists\, y_1, y_2, \dots, y_l)(A_1 \wedge A_2 \wedge \dots \wedge A_r),$$

где A_i имеет вид $R_j(u_1, u_2, \dots, u_{d_j}$, а переменные u принадлежат множеству $(\{y_1, y_2, \dots, y_l\} \cup D)$.

ВОПРОС. Верно ли, что Q истинен при интерпретации его как утверждения об R и D?

Источник [67]. К этой задаче сводится КЛИКА.

Комментарий. Если заменить конъюнктивный запрос произвольным предложением первого порядка, включающим предикаты из набора R, то задача становится $P - SPASE$-полной (даже если $D = \{0, 1\}$).

[ХП 32] **ЭКВИВАЛЕНТНОСТЬ ТАБЛИЦ**

УСЛОВИЕ. Заданы множество имён атрибутов A, семейство F упорядоченных пар подмножеств множества A, множество выделенных переменных X, множество невыделенных переменных Y, для каждого $a \in A$ множество констант C_a, две «таблицы» T_1 и T_2 над X, Y и C_a. (Таблица, по существу,– это матрица, в которой каждому имени атрибута соответствует столбец, а элементами являются пустой символ и элементы множеств X, Y и C_a. Детали определения и интерпретацию в терминах реляционных выражений можно найти в работе [12].)

ВОПРОС. Верно ли, что T_1 и T_2 «слабо эквивалентны»? Т. е. верно ли, что T_1 и T_2 при «универсальной интерпретации» определяют идентичные отношения?

Источник [12]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной в следующих случаях: (1) таблицы получаются из «выражений», не содержащих операций «выбрать»; (2) таблицы получаются из выражений, содержащих операцию «выбрать», при этом семейство F пусто; (3) семейство F пусто, таблицы не содержат констант, причём таблицы необязательно получаются из каких-то выражений. Для «простых» таблиц задача решается за полиномиальное время. Аналогичные результаты имеют место для «сильной эквивалентности», при которой таблицы должны представлять идентичные отношения при всех интерпретациях. Задача «включения» таблиц, однако, *NP*-полна даже для простых таблиц и ещё более ограниченных видов таблиц [462].

[ХП 33] **СОГЛАСУЕМОСТЬ ИСТОРИИ БАЗ ДАННЫХ**

УСЛОВИЕ. Заданы множество V переменных базы данных, набор T «транзакций» (R_i, W_i), $1 \leqslant i \leqslant n$ (где R_i и W_i –подмножества множества V, называемые «читаемое множество» и «записываемое множество» соответственно), и «история» H набора T (где «история» есть такая перестановка множеств R_i и W_i, в которой любое множество R_i идёт раньше соответствующего множества W_i).

ВОПРОС. Существует ли согласованная история H' набора T (т. е. такая история, в которой каждое R_i идёт непосредственно перед соответствующим множеством W_i), эквивалентная истории H? (История H называется эквивалентной истории H', если (1) обе истории имеют одинаковое множество «живых» транзакций (транзакция (R_i, W_i) называется живой в истории H, если существует такая переменная $v \in V$, что либо W_i – последнее записываемое множество, содержащее v, либо W_i – последнее записываемое множество, содержащее переменную v перед некоторой живой транзакцией, содержащей v в читаемом множестве) и (2) для любых двух живых транзакций (R_i, W_i) и (R_j, W_j) и любой переменной $v \in W_i \cap R_j$ множество W_i является последним записываемым множеством, содержащим v перед R_j в истории H, тогда и только тогда, когда W_i – последнее записываемое множество, содержащее v перед R_j в истории H'.)

Источник [415, 413]. К этой задаче сводится МОНОТОННАЯ 3-ВЫП.

Комментарий. Другие варианты и подслучаи этой задачи, разрешимые за полиномиальное время, можно найти в работе [413].

[ХП 34] **БЕЗОПАСНОСТЬ СИСТЕМЫ ТРАНЗАКЦИИ БАЗЫ ДАННЫХ (*)**

УСЛОВИЕ. Заданы множество V переменных базы данных и набор T «транзакций» $(R_i, W_i), 1 \leqslant i \leqslant n$, где R_i и W_i – подмножества V.

ВОПРОС. Верно ли, что любая история H транзакции T эквивалентна некоторой согласованной истории?

Источник [415]. К этой задаче сводится МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ.

Комментарий. Не известно, принадлежит ли задача к классу NP или к классу $co - NP$. Проверка того, что любая история H набора T «D-эквивалентна» некоторой согласованной истории, может быть осуществлена за полиномиальное время (две истории называются D-эквивалентными, если одна из них может быть получена из другой такой последовательностью перестановок соседних множеств, что после каждой перестановки новая история остаётся эквивалентной предыдущей).

[ХП 35] **СОВМЕСТИМОСТЬ ТАБЛИЦ ЧАСТОТ БАЗЫ ДАННЫХ**

УСЛОВИЕ. Заданы множество A имён атрибутов, для каждого $a \in A$ домен D_a, множество объектов V, набор F таблиц частот для некоторых пар элементов $a, b \in A$ (таблицей частот для имён атрибутов $a, b \in A$ называется функция $f_{a,b} : D_a \times D_b \to Z^+$, у которой сумма значений $f_{a,b}(x, y)$ по всем $x \in D_a$ и $y \in D_b$ равна $|V|$) и множество K троек (v, a, x), где $v \in V$, $a \in A$ и $x \in D_a$, представляющее множество известных значений атрибутов.

ВОПРОС. Совместимы ли таблицы из набора F с известными значениями атрибутов из K? Иначе говоря, существует ли такой набор функций $g_a : V \to D_a$ (для каждого $a \in A$), что (1) $g_a(v) = x$, если $(v, a, x) \in K$, и (2) для всех $f_{a,b} \in F, x \in D_a$ и $y \in D_b$ число таких $v \in V$, для которых $g_a(v) = x$ и $g_b(v) = y$, совпадает с $f_{a,b}(x, y)$?

Источник [447]. К этой задаче сводится 3-ВЫП.

Комментарий. Из результата, сформулированного выше, следует, что если $P \neq NP$, то не существует полиномиального алгоритма для «согласования» базы данных на основе её таблиц часто посредством вывода заранее указанных имён (детали см. в [447]).

[ХП 36] **НАДЕЖНОСТЬ СИСТЕМ ЗАЩИТЫ ФАЙЛОВ (*)**

УСЛОВИЕ. Заданы множество R «полномочий», множество объектов O, множество $S \subseteq O$ субъектов, множество $P(s, o) \subseteq R$ полномочий для каждой упорядоченной пары $s \in S$ и $o \in O$, конечное множество C команд (каждая из которых имеет вид, если $r_1 \in P(X_1, Y_1)$, $r_2 \in P(X_2, Y_2)$ и ... $r_m \in P(X_m, Y_m)$, то $\theta_1, \theta_2, \dots, \theta_m$ для всех $m, n \geqslant 0$ и каждое θ_i имеет вид «ввести r_i в $P(X_j, Y_k)$» или «вывести r_i из $P(K_j, Y_k)$») и выделенное «полномочие» $r' \in R$.

ВОПРОС. Существует ли такая последовательность команд из C и метод идентификации всех r_i, X_j и Y_k конкретными элементами из множеств R, S и O соответственно, что в некоторый момент исполнения последовательности полномочие r' заносится в множество $P(s, o)$, не содержавшее r' до этого момента? (Детали о работе таких последовательностей см. в указанной ниже работе.)

Источник [199]. К этой задаче сводится ДОПУСКАЕМОСТЬ ЛИНЕЙНО ОГРАНИЧЕННЫМ АВТОМАТОМ.

Комментарий. Задача $P - SPACE$-полна. Если допускаются операции, позволяющие создать или убрать «объекты» и «субъекты», то задача неразрешима даже тогда, когда для некоторых «фиксированных» систем допускается изменение начальных значений $P(s, o)$. Если ни одна из команд не может содержать более одной операции, то в общем случае задача является *NP*-полной и разрешима за полиномиальное время для фиксированных систем.

А5. ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ

А5.1. Задачи составления расписания в случае одного процессора

[ТР 1] **УПОРЯДОЧЕНИЕ С НЕОДНОВРЕМЕННЫМ ПОСТУПЛЕНИЕМ И ДИРЕКТИВНЫМИ СРОКАМИ**

УСЛОВИЕ. Заданы множество T заданий, а также для каждого задания $t \in T$ длительность $l(t) \in Z^+$, момент его поступления $r(t) \in Z_0^+$ и директивный срок $d(t) \in Z^+$.

ВОПРОС. Существует ли однопроцессорное расписание выполнения заданий из T, согласованное с моментами поступления и директивными сроками, т. е. существует ли однозначная функция $\sigma : T \to Z_0^+$, такая, что, во-первых, из условия $\sigma(t) > \sigma(t')$ следует, что $\sigma(t) \geqslant \sigma(t') + l(t')$, и, во-вторых, для всех $t \in T \sigma(t) \geqslant r(t)$ и $\sigma(t) + l(t) \geqslant d(t)$?

Источник [150]. К этой задаче сводится 3-РАЗБИЕНИЕ (см. разд. 4.2).

Комментарий. Рассматриваемая задача *NP*-полна в сильном смысле. Она разрешима за псевдополиномиальное время, если число значений, принимаемых величинами $r(t)$ и $d(t)$, ограничено константой, и остаётся *NP*-полной (в обычном смысле), даже когда эта константа равна 2. В тех случаях когда все длительности равны 1, либо разрешены прерывания, либо все моменты поступления заданий равны 0, задача может быть решена за полиномиальное время, при этом в условии задачи могут присутствовать ограничения предшествования [323], [319]. Задача решается за полиномиальное время также и тогда, когда моменты поступления и директивные сроки являются произвольными рациональными числами и, кроме того, имеются ограничения предшествования; для этого требуется, чтобы все задания имели одинаковые длительности [65, 492, 156], либо чтобы разрешались прерывания [41].

[ТР 2] МИНИМИЗАЦИЯ ЧИСЛА НЕВЫПОЛНЕННЫХ ЗАДАНИЙ

УСЛОВИЕ. Заданы множество T заданий, частичный порядок $\lessdot$ на T, для каждого задания $t \in T$ длительность $l(t) \in Z^+$, директивный срок $d(t) \in Z^+$ и положительное целое число $K \geqslant |T|$.

ВОПРОС. Существует ли однопроцессорное расписание σ для заданий из T, удовлетворяющее ограничениям частичного порядка (т. е. такое, что из $t \lessdot t'$ следует $\sigma(t) + l(t) < \sigma(t')$) и такое, что существует не более K заданий из T, для которых $\sigma(t) + l(t) > d(t)$?

Источник [148]. К этой задаче сводится КЛИКА (см. разд. 3.2.3).

Комментарий. Задача остаётся *NP*-полной, даже если все длительности равны 1 и если условия частичного порядка имеют вид «цепочки» (каждое задание имеет не более чем одно непосредственно предшествующее и не более чем одно непосредственно за ним следующее) [334]. Она разрешима за полиномиальное время, если $K = 0$ [323] или если отношение $\lessdot$ пусто [392, 488]. В случае когда отношение $\lessdot$ пусто, задача остаётся полиномиально разрешимой, если в неё добавить «согласованные» моменты поступления заданий (это значит, что из условий $r(t) < r(t')$ следует $d(t) \leqslant \sigma(t')$) [289]. Если же эти моменты поступления произвольны, то задача *NP*-полна (см. предыдущую задачу).

[ТР 3] МИНИМИЗАЦИЯ ВЕСА НЕВЫПОЛНЕННЫХ ЗАДАНИЙ

УСЛОВИЕ. Заданы множество T заданий, а также для каждого задания $t \in T$ длительность $l(t) \in Z^+$, вес $w(t) \in Z^+$, директивный срок $d(t) \in Z^+$ и положительное число K.

ВОПРОС. Существует ли однопроцессорное расписание σ для заданий из T, такое, что сумма весов $\sum w(t)$ (взятая по тем $t \in T$, для которых $\sigma(t) + l(t) > d(t)$) не превосходит K?

Источник [280]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача может быть решена за псевдополиномиальное время (время, полиномиальное относительно $|T|$, $\sum l(t)$ и $\log \sum w(t)$ [332]). Она разрешима за полиномиальное время, если веса «согласованы», т. е. если из $w(t) < w(t')$ следует $l(t) \geqslant l(t')$ [326].

[ТР 4] МИНИМИЗАЦИЯ ВЗВЕШЕННОГО МОМЕНТА ОКОНЧАНИЯ

УСЛОВИЕ. Заданы множество T заданий, частичный порядок $\lessdot$ на T, для каждого значения $t \in T$ длительность $l(t) \in Z^+$, вес $w(t) \in Z^+$ и положительное число K.

ВОПРОС. Существует ли однопроцессорное расписание σ для заданий из T, удовлетворяющее условиям частичного порядка и такое, что

$$\sum_{t\in T}(\sigma(t)+l(t)w(t)\leqslant K?$$

Источник [330]. К этой задаче сводится ОПТИМАЛЬНОЕ ЛИНЕЙНОЕ УПОРЯДОЧЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле и остаётся таковой, даже если все длительности равны 1 или если все веса равны 1. Она разрешима за полиномиальное время, если порядок $\lessdot$ является «лесом» [217, 5, 139, 489] или последовательно-параллельным графом [293, 330, 4, 391].

Если вместо условий $\lessdot$ в задачу ввести индивидуальные директивные сроки, задача будет *NP*-полной в сильном смысле [334], если же все веса заданий равны, то она станет полиномиально разрешимой [334]. Если директивные сроки заменить индивидуальными моментами поступления, полученная задача будет *NP*-полна в сильном смысле, даже если все веса заданий равны 1 [338]. Версия последней задачи, в которой допустимы прерывания, *NP*-полна в сильном смысле [311], однако может быть решена за полиномиальное время, если все веса равны [186].

[TP 5] МИНИМИЗАЦИЯ ШТРАФА ЗА НЕВЫПОЛНЕНИЕ ЗАДАНИЙ

УСЛОВИЕ. Заданы множество T заданий, а также для каждого задания $t\in T$ длительность $l(t)\in Z^+$, вес $w(t)\in Z^+$, директивный срок $d(t)\in Z^+$ и положительное целое число K.

ВОПРОС. Существует ли однопроцессорное расписание σ для заданий из T, такое, что сумма $\sum_t(\sigma(t)+l(t)-d(t))w(t)$, взятая по тем $t\in T$, для которых $\sigma(t)+l(t)>d(t)$, не превосходит K?

Источник [328]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле. Если все веса равны, она разрешима за псевдополиномиальное время [328], её *NP*-полнота (в обычном смысле) не установлена. Если все длительности равны (а веса произвольны), задача разрешима за полиномиальное время с помощью двудольных паросочетаний. Если добавить условия предшествования, то она будет *NP*-полной, даже в случае равных длительностей и равных весов [336]. Если же вместо этого добавить моменты поступления заданий, то задача будет *NP*-полной в сильном смысле при равных весах (см. задачу УПОРЯДОЧЕНИЕ С НЕОДНОВРЕМЕННЫМ ПОСТУПЛЕНИЕМ И ДИРЕКТИВНЫМИ СРОКАМИ), причём при равных длительностях и произвольных весах эта задача полиномиально разрешима с помощью двудольных паросочетаний [186].

[TP 6] УПОРЯДОЧЕНИЕ С ДИРЕКТИВНЫМИ СРОКАМИ И ВРЕМЕНАМИ ПЕРЕНАЛАДОК

УСЛОВИЕ. Заданы: множество C «компиляторов», множество T заданий, для каждого задания $t\in T$ длительность $l(t)\in Z^+$, директивный срок

$d(t) \in Z^+$, свой компилятор $k(t) \in C$, а также для каждого $c \in C$ «время переналадки» $l(c) \in Z_0^+$.

ВОПРОС. Существует ли однопроцессорное расписание σ для заданий из T, удовлетворяющее всем директивным срокам и следующему дополнительному ограничению: если задание t начинается раньше t' (т. е. $\sigma(t) < \sigma(t')$), причём задания t и t' «сопряжённые», то $\sigma(t') \geqslant \sigma(t) + l(t) + l(k(t'))$? (Задания t и t', где $\sigma(t) < \sigma(t')$, называются «сопряжёнными», если между ними нет промежуточного задания (т. е. такого t'', что $\sigma(t) < \sigma(t'') < \sigma(t')$) и у них различные компиляторы, $k(t) \neq k(t')$.)

Источник [60]. К этой задаче сводится РАЗБИЕНИЕ,

Комментарий. Задача остаётся *NP*-полной, если все времена переналадки равны. Имеется близкая задача, в которой времена переналадок заменены стоимостями переналадок, и требуется выяснить, существует ли расписание, удовлетворяющее всем директивным срокам и имеющее суммарную стоимость переналадок не больше K; эта задача *NP*-полна, даже если все стоимости переналадок равны. Обе задачи могут быть решены за псевдополиномиальное время, когда число различных значений, принимаемых директивными сроками, ограничено константой. Если это число не ограничено, не известно, являются ли эти задачи *NP*-полными в сильном смысле.

[ТР 7] **МИНИМИЗАЦИЯ МАКСИМАЛЬНЫХ ЗАТРАТ**

УСЛОВИЕ. Заданы множество заданий T, частичный порядок $\lessdot$ на T, «затраты» $c(t) \in Z$ для каждого $t \in T$ (если $c(t) < 0$, их можно рассматривать как «доход») и константа $K \in Z$.

ВОПРОС. Существует ли однопроцессорное расписание σ для заданий из T, которое удовлетворяет условиям частичного порядка и имеет следующее свойство: для каждого $t \in T$

$$\sum_{t' : \sigma(t') \leqslant \sigma(H)} c(t) \leqslant K?$$

Источник [1]. К этой задаче сводится ДОСТАТОЧНОСТЬ ЧИСЛА РЕГИСТРОВ.

Комментарий. Задача остаётся *NP*-полной, даже если $c(t) \in -1, 0, 1$ для всех $t \in T$. Она разрешима за полиномиальное время, если условия частичного порядка c представимы в виде параллельно-последовательного графа [2, 391].

А5.2. Многопроцессорные расписания для параллельных процессоров

[ТР 8] **МНОГОПРОЦЕССОРНОЕ РАСПИСАНИЕ**

УСЛОВИЕ. Заданы множество заданий T, число процессоров $m \in Z^+$, длительность $l(t) \in Z^+$ для каждого

ВОПРОС. Существует ли m-процессорное расписание для заданий из T, которое удовлетворяет общему директивному сроку D? Другими словами, существует ли функция $\sigma : T \to Z_0^+$, такая, что для всех $u \geqslant 0$ число заданий $t \in$, для которых $\sigma(t) \leqslant u < \sigma(t) + l(t)$, не превышает m и, кроме того, для всех $t \in T$ имеет место $\sigma(t) + l(t) \leqslant D$?

Источник – настоящая книга. К этой задаче сводится задача РАЗБИЕНИЕ (см. разд. 3,2.1).

Комментарий. Задача остаётся *NP*-полной при $m = 2$; она разрешима за псевдополиномиальное время при любом фиксированном m. Задача *NP*-полна в сильном смысле при произвольном m (задача 3-РАЗБИЕНИЕ – её частный случай). Если все задания имеют одинаковую длительность, задача тривиально решается за полиномиальное время, даже если процессоры имеют различные производительности.

[ТР 9] **РАСПИСАНИЕ С УСЛОВИЯМИ ПРЕДШЕСТВОВАНИЯ**

УСЛОВИЕ. Заданы множество заданий T, каждое из которых имеет длительность $l(t) = 1$, число процессоров $m \in Z^+$, частичный порядок $\lessdot$ на T и директивный срок $D \in Z^+$.

ВОПРОС. Существует ли m-процессорное расписание σ для заданий из T, которое удовлетворяет общему директивному сроку D и условиям предшествования (т. е. из соотношения $t \lessdot t'$ следует $\sigma(t') \geqslant \sigma(t) + l(t) = \sigma(t) + 1$)?

Источник [518]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной при $D = 3$ [336]. Она разрешима за полиномиальное время, если $m = 2$ (см. например [85]), а также если m произвольно, а условия $\lessdot$ представимы в виде «леса» [225] или имеют дополнением триангулированный граф [419]. Для случая, когда условия $\lessdot$ общего вида, а m фиксировано ($m \geqslant 3$), сложность задачи не установлена. При $m = 2$ задача становится *NP*-полной, если длительности заданий принимают значения 1 и 2 [518]. Если для каждого задания t назначен процессор $p(t)$, задача является *NP*-полной при $m = 2$ и произвольных условиях $\lessdot$, а также когда m произвольно, а условия $\lessdot$ – «лес»; она разрешима за полиномиальное время, если m произвольно, а условия $\lessdot$ – «циклический лес» [184].

[ТР 10] **РАСПИСАНИЕ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ**

УСЛОВИЕ. Задано множество заданий T, каждое из которых имеет длительность $l(t) = 1$, число процессоров $m \in Z^+$ число видов ресурсов $r \in Z^+$, границы ресурсов $B_i, 1 \leqslant i \leqslant r$, потребности в ресурсах $R_i(t), 0 \leqslant R_i(t) \leqslant B_i$ (для всех заданий t и ресурсов i, и общий директивный срок $D \in Z^+$.

ВОПРОС. Существует ли m-процессорное расписание σ для заданий из T, которое удовлетворяет общему директивному сроку D и ограничениям на ресурсы (последнее условие означает, что если через $S(u)$ обозначить множество всех тех заданий t из T, для которых $\sigma(t) \leqslant u < \sigma(t) + l(t)$, то для всех $u > 0$ и всех видов ресурсов i сумма $\sum_{t \in S(u)} R_i(t)$ не превышает B_i?

Источник [145]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Задача является *NP*-полной в сильном смысле, даже если $r = 1$ и $m = 3$. Она решается за полиномиальное время с помощью техники паросочетаний при $m = 2$ и произвольном r. Если добавить к условию задачи отношения частичного порядка $\lessdot$, она становится *NP*-полной в сильном смысле при $r = 1$, $m = 2$ и условиях $\lessdot$, представимых в виде «леса». Если потребности в ресурсах принимают дискретные значения 0 или B_i, задача является *NP*-полной при $m = 2, r = 1$ и условиях $\lessdot$ общего вида [519].

[ТР 11] **РАСПИСАНИЕ С ИНДИВИДУАЛЬНЫМИ ДИРЕКТИВНЫМИ СРОКАМИ**

УСЛОВИЕ. Заданы множество заданий T, каждое из которых имеет длительность $l(t) = 1$, число процессоров $m \in Z^+$, частичный порядок $\lessdot$ на T и для каждого $t \in T$ директивный срок $d(t) \in Z^+$.

ВОПРОС. Существует ли m-процессорное расписание σ для заданий из T, которое удовлетворяет условиям предшествования и всем директивным срокам (т. е. $\sigma(t) + l(t) \leqslant d(t)$ для всех $t \in T$?

Источник [58]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача остаётся *NP*-полной, даже если условия $\lessdot$ представимы в виде «прямого дерева» (т. е. у каждого задания имеется не более одного непосредственно за ним следующего). Она разрешима за полиномиальное время, если $m = 2$, а условия $\lessdot$ общего вида [148], причём этот факт справедлив, даже если включить в условие задачи индвидуальные моменты поступления [150]. Если условия предшествования $\lessdot$ отсутствуют, задача решается за полиномиальное время с помощью техники паросочетаний для произвольного значения m, даже если в условии есть моменты поступления и имеется один вид ресурса, потребность в котором 0 или 1 [43, 44].

[ТР 12] **РАСПИСАНИЕ С ПРЕРЫВАНИЯМИ**

УСЛОВИЕ. Заданы множество заданий T, число процессоров $m \in Z^+$, частичный порядок $\lessdot$ на T, длительности $l(t) \in Z^+$ для всех $t \in T$ и общий директивный срок $D \in Z^+$.

ВОПРОС. Существует ли m-процессорное расписание для заданий из T, которое допускает прерывания заданий и удовлетворяет как условиям предшествования, так и общему директивному сроку? (Такое расписание σ идентично обычному m-процессорному расписанию, за исключением того, что разрешается разбивать каждое задание $t \in T$ на любое число «частичных заданий» $t_1, t_2, \ldots, t_k$, таких, что

$$\sum_{i=1}^{k} l(t_i) = l(t) \text{ и } \sigma(t_{i+1}) \geqslant \sigma(t_i) + l(t_i)$$

для всех $i, 1 \leqslant i \leqslant k$. Условия частичного предшествования распространяются на частичные задания следующим образом: каждое частичное задание, полученное из t, предшествует всем частичным заданиям t', если $t \lessdot t'$.)

Источник [518]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача разрешима за полиномиальное время в следующих случаях: если $m = 2$ [394], если ограничения $\lessdot$ – «лес» [395], если ограничений предшествования нет и добавлены индивидуальные директивные сроки [219]. Когда «однородные» процессоры имеют «различные производительности», задача разрешима за полиномиальное время при $m = 2$ или при отсутствии ограничений предшествования [223, 183]; если выполнено последнее условие, задача остаётся полиномиально разрешимой при наличии индивидуальных директивных сроков [467]; если же одновременно $m = 2$ и отсутствуют ограничения предшествования, то задача полиномиально разрешима, даже если добавлены целочисленные моменты поступления работ и директивные сроки [311]. В случае «независимых» процессоров задача решается за полиномиальное время при фиксированном m и отсутствии ограничений предшествования [180]; при произвольном m и отсутствии ограничений предшествования она может быть решена методом линейного программирования [331].

[ТР 13] МИНИМИЗАЦИЯ ВЗВЕШЕННОГО МОМЕНТА ОКОНЧАНИЯ (m ПРОЦЕССОРОВ)

УСЛОВИЕ. Заданы множество заданий T, число процессоров $m \in Z^+$, для каждого задания $t \in T$ длительность $l(t) \in Z^+$ и вес $w(t) \in Z^+$, а также положительное целое число K.

ВОПРОС. Существует ли m-процессорное расписание σ для заданий из T, такое, что

$$\sum_{t\in T}(\sigma(t) + l(t))w(t) \leqslant K?$$

Источник [338]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача остаётся *NP*-полной при $m = 2$, она является *NP*-полной в сильном смысле при произвольном m [318]. Задача разрешима за псевдополиномиальное время при фиксированных значениях m. Эти результаты сохраняются, если допустимы «прерывания» заданий [379]. Задача решается (с помощью техники паросочетаний) за полиномиальное время, если все длительности равны. Если же длительности не равны, а все веса равны между собой, то она разрешима за полиномиальное время даже для процессоров разной производительности [90] и для «независимых» процессоров [218, 59]. В случае процессоров разной производительности она разрешима за полиномиальное время и тогда, когда разрешены прерывания [179]. Если добавлены ограничения предшествования, исходная задача *NP*-полна в сильном смысле, даже если все веса равны, $m = 2$ и частичный порядок есть либо «внутреннее дерево», либо «внешнее дерево» [478]. Если в условия добавить ограничения на ресурсы, то частные случаи, приведённые в задаче РАСПИСАНИЕ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ, будут все *NP*-полными, даже если все веса равны [42].

А5.3. Многопроцессорные расписания конвейерного типа

[ТР 14] **ОТКРЫТОЕ КОНВЕЙЕРНОЕ РАСПИСАНИЕ**

УСЛОВИЕ. Заданы число процессоров $m \in Z^+$, множество заданий J, каждое из которых состоит из m операций (т. е. задание $j \in J$ состоит из операций $t_1[j], t_2[j], \dots, t_m[j]$, причём операция $t_i[j]$ выполняется процессором i), длительность $l(t) \in Z$ для каждой операции t и общий директивный срок $D \in Z^+$.

ВОПРОС. Существует ли открытое расписание для заданий из J, удовлетворяющее директивному сроку? Другими словами, существует ли набор из m однопроцессорных расписаний $\sigma_i : J \to Z_0^+, 1 \leqslant i \leqslant m$, таких, что: (1) из условия $\sigma_i(j) > \sigma_i(k)$ следует $\sigma_i(j) \geqslant \sigma_i(k)+l(t_i[k])$; (2) для каждого $j \in J$ все интервалы $[\sigma_i(j), \sigma_i(j)+l(t_i[j])$ не пересекаются и (3) $\sigma_i(j)++l(t_i[j]) \leqslant D$ для всех $i, j : 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant |J|$?

Источник [181]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача остаётся *NP*-полной при $m = 3$, но разрешима за полиномиальное время, если $m = 2$. Она является *NP*-полной в сильном смысле, если m произвольно [334]. В общем случае задача разрешима за полиномиальное время, если разрешены прерывания заданий [181], этот факт остаётся в силе и тогда, когда заданы два различных момента поступления заданий [69]. При $m = 2$ и разрешённых прерываниях она разрешима за полиномиальное время, даже если заданы произвольные моменты поступления заданий; в общем случае задача с разрешёнными прерываниями, произвольными моментами поступления и директивными сроками сводится к задаче линейного программирования [69].

[ТР 15] **ПОСЛЕДОВАТЕЛЬНОЕ КОНВЕЙЕРНОЕ РАСПИСАНИЕ**

УСЛОВИЕ. Заданы: число процессоров $m \in Z^+$, множество заданий J, причём каждое задание $j \in J$ состоит из m операций $t_1[j], t_2[j], \dots, t_m[j]$, длительность операции $l(t) \in Z_0^+$ для каждой операции t, общий директивный срок $D \in Z^+$.

ВОПРОС. Существует ли последовательное расписание для заданий из J, удовлетворяющее общему директивному сроку? (Последовательным расписанием называется открытое расписание, удовлетворяющее следующему дополнительному ограничению: для всех $j \in J$ и $1 \leqslant i \leqslant m\sigma_{i+1}(j) \geqslant \sigma_i(j) + l(t_i[j].)$

Источник [155]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле при $m = 3$. Она разрешима за полиномиальное время при $m = 2$ [268]. Эти два результата справедливы, если допустимы прерывания [182], однако, если добавить моменты поступления заданий, задача будет *NP*-полной в сильном смысле даже при $m = 2$ [69]. Если добавить условие

$$\sum_{j \in J} \sigma_m(j) + l(t_m[j]) \leqslant K,$$

то полученная задача (без прерываний) будет *NP*-полна в сильном смысле, даже если $m = 2$ [155].

[ТР 16] **ПОСЛЕДОВАТЕЛЬНОЕ РАСПИСАНИЕ БЕЗ ОЖИДАНИЯ**

УСЛОВИЕ. То же самое, как и в предыдущей задаче, ПОСЛЕДОВАТЕЛЬНОЕ РАСПИСАНИЕ.

ВОПРОС. Существует ли последовательное расписание для заданий из J, удовлетворяющее директивному сроку и дополнительному условию, для всех $j \in J$ и $1 \leqslant i \leqslant m, \sigma_i(j) = \sigma_i(j) + l(t_i[j])$?

Источник [338]. К этой задаче сводится ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Задача *NP*-полна в сильном смысле для любого фиксированного $m \geqslant 4$ [416]. Она разрешима за полиномиальное время при $m = 2$ [172], однако *NP*-полна в сильном смысле при $m = 2$, если допускаются задания не имеющие операций, выполняемых на первом процессоре [468]. Сложность не установлена для случая $m = 3$. Если добавить условие

$$\sum_{j \in J} \sigma_m(j) + l(t_m[j]) \leqslant K,$$

а m – произвольно, то задача будет *NP*-полной в сильном смысле [338], причём если m фиксировано ($m \geqslant 2$), то сложность такой задачи не установлена. Аналогично версии задач ОТКРЫТОЕ РАСПИСАНИЕ и ОБЩЕЕ КОНВЕЙЕРНОЕ РАСПИСАНИЕ (см. ниже), когда добавлено условие, запрещающее ожидание, являются *NP*-полными в сильном смысле при $m = 2$ [468].

[ТР 17] **ДВУХПРОЦЕССОРНОЕ ПОСЛЕДОВАТЕЛЬНОЕ РАСПИСАНИЕ С БУФЕРОМ**

УСЛОВИЕ. То же самое, как в задаче ПОСЛЕДОВАТЕЛЬНОЕ РАСПИСАНИЕ, при условии, что $m = 2$ и дополнительно задана «ёмкость буфера» $B \in Z_0^+$.

ВОПРОС. Существует ли последовательное расписание для заданий из J, удовлетворяющее общему директивному сроку и следующему условию: для любого $u \geqslant 0$ число заданий $j \in J$, для которых одновременно $\sigma_1(j) + l(t_1[j]) \leqslant u$ и $\sigma_2(j) > u$, не превосходит B?

Источник [416]. К этой задаче сводится ТРЁХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ.

Комментарий. Задача *NP*-полна в сильном смысле для любого фиксированного $B, 1 \leqslant B < \infty$. Она разрешима за полиномиальное время, если $B = 0$ [172] или если $B \geqslant |J| - 1$ [268].

[ТР 18] **ОБЩЕЕ КОНВЕЙЕРНОЕ РАСПИСАНИЕ**

УСЛОВИЕ. Заданы: число процессоров $m \in Z^+$, множество заданий J (причём каждое задание $j \in J$ состоит из упорядоченного набора операций

$t_k[j], 1 \leqslant k \leqslant n_j$), для каждой операции t длительность $l(t) \in Z_0^+$, определённый процессор

$$p(t) \in \{1, 2, \dots, m\}, \text{ где } p(t_k[j]) \neq p(t_{k+1}[j])$$

для всех $j \in J$ и $1 \leqslant k \leqslant n_j$, и директивный срок $D \in Z^+$.

ВОПРОС. Существует ли общее конвейерное расписание для заданий из J, удовлетворяющее общему директивному сроку? Другими словами, существует ли набор однопроцессорных расписаний σ_i, отображающих множество $\{t : p(t) = i\}$ в Z_0^+, $1 \leqslant i \leqslant m$ и удовлетворяющих следующим условиям:

(1) из $\sigma_i(t) > \sigma_i(t')$ следует, что $\sigma_i(t) \geqslant \sigma_i(t') + l(t')$;

(2) $\sigma(t_{k+1}[j]) \geqslant \sigma(t_k[j]) + l(t_k[j])$

(где индексы соответствуют расписанию σ) для всех $j \in J$ и $1 \leqslant k \leqslant n_j$;

(3) для всех $j \in J$

$$\sigma(t_n[j]) + l(t_{n_j}[j]) \leqslant D$$

(где опять индексы соответствуют расписанию σ)?

Источник [155]. К этой задаче сводится 3-РАЗБИЕНИЕ.

Комментарий. Задача *NP*-полна в сильном смысле при $m = 2$. Она может быть решена за полиномиальное время, если $m = 2$ и $n_j \leqslant 2$ для всех $j \in J$ [256]. Она *NP*-полна (в обычном смысле), если $m = 2$ и $n_j \leqslant 3$ для всех $j \in J$ или если $m = 3$ и $n_j \leqslant 2$ для всех $j \in J$ [182]. Все эти результаты имеют место и в случае, когда разрешаются прерывания заданий [182]. Если прерывания не разрешаются и все операции имеют одинаковую длительность, задача *NP*-полна для $m = 3$; при $m = 2$ её сложность не установлена [337][6]).

А5.4. Разные задачи

[ТР 19] **СОСТАВЛЕНИЕ УЧЕБНОГО РАСПИСАНИЯ**

УСЛОВИЕ. Заданы множество H «рабочих часов», множество C «преподавателей», множество T «учебных дисциплин», для каждого $c \in C$ дано подмножество $A(c) \subseteq H$, называемое «допустимыми часами для преподавателя c», для каждой дисциплины $t \in T$ – подмножество $A(t) \subseteq H$, называемое «допустимыми часами для дисциплины t», и для каждой пары $(c, t) \in C \times T$ – число $R(c, t) \in Z_0^+$, называемое «требуемой нагрузкой».

ВОПРОС. Существует ли учебное расписание, обслуживающее все дисциплины? Другими словами, существует ли функция $f : C \times T \times H \to \{0, 1\}$ (где $f(c, t, h) = 1$ означает, что преподаватель c занимается дисциплиной t в момент h) , удовлетворяющая следующим условиям: (1) $f(c, t, h) = 1$ только тогда, когда $h \in A(c) \cap A(t)$, (2) для каждого $h \in H$ и $c \in C$ существует не более одного $t \in T$, такого, что $f(c, t, h) = 1$, (3) для каждого $h \in H$ и $t \in T$ существует не более одного $c \in C$, такого, что $f(c, t, h) = 1$,

[6]В. Г. Тимковский показал, что последняя задача принадлежит классу P (результат ещё не опубликован). – *Прим. ред.*

(4) для каждой пары $(c, t) \in C \times T$ существует ровно $R(c, t)$ значений h, для которых $f(c, t, h) = 1$?

Источник [114]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, даже если $|H| = 3$, $A(t) = H$ для $t \in T$ и все $R(c, t) \in \{0, 1\}$. Общая задача разрешима за полиномиальное время, если $|A(c)| \leqslant 2$ для всех $c \in C$ или если $A(c) = A(t) = H$ для всех $c \in C$ и $t \in T$.

[ТР 20] **КОМПЛЕКТОВАНИЕ ШТАТА ИСПОЛНИТЕЛЕЙ**

УСЛОВИЕ. Заданы положительные целые числа m и k, набор C m-мерных векторов, каждый из которых состоит из k единиц и $m - k$ нулей (и представляет собой возможное распределение исполнителей), m-мерный вектор R неотрицательных чисел (называемый вектором «потребностей») и число исполнителей n.

ВОПРОС. Существует ли расписание $f : C \to Z_0^*$, такое, что

$$\sum_{\bar{c} \in C} f(c) \leqslant n \text{ и } \sum_{\bar{c} \in C} f(\bar{c}) \cdot c \geqslant R?$$

Источник [152]. К этой задаче сводится ТП-3.

Комментарий. Задача разрешима за полиномиальное время, если все $\bar{c} \in C$ обладают свойством цикличности (т. е. все единицы расположены в одном и том же порядке, если считать, что первая позиция следует за позицией m) [30]. Этот случай соответствует ситуации, когда исполнители доступны в определённые часы рабочего дня или дни недели.

[ТР 21] **ПЛАНИРОВАНИЕ ПРОИЗВОДСТВА**

УСЛОВИЕ. Заданы число $n \in Z^+$ плановых периодов, для каждого периода $i, 1 \leqslant i \leqslant n$, «потребность» $r_i \in Z_0^+$, «пропускная способность производства» $c_i \in Z_0^+$, стоимость переналадок производства $b_i \in Z_0^+$, коэффициент приращения стоимости продукции $p_i \in Z_0^+$, коэффициент стоимости запасов $h_i \in Z_0^+$, общая граница $B \in Z^+$.

ВОПРОС. Существуют ли объёмы производства $x_i \in Z_0^+$ и соответствующие уровни запасов $I_i = \sum_{j=1}^{i}(x_j - r_j), 1 \leqslant i \leqslant n$, такие, что все $x_i \leqslant c_i$, все $I_i \geqslant 0$ и

$$\sum_{i=1}^{n} p_i x_i + h_i I_i + \sum_{x_i > 0} \leqslant B?$$

Источник [339]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача разрешима за псевдополиномиальное время; она остаётся *NP*-полной, даже если все «потребности» равны, все стоимости переналадок равны и все стоимости запасов равны 0. Если все пропускные способности равны, то задача разрешима за полиномиальное время [124]. Упомянутые алгоритмы можно обобщить на случай произвольных монотонных неубывающих выпуклых функций стоимости (если они вычислимы за полиномиальное время).

[ТР 22] **ИЗБЕЖАНИЕ ТУПИКА**

УСЛОВИЕ. Заданы множество диаграмм переходов для m процессов $\{P_1, P_2, \dots, P_m\}$ (где каждая диаграмма представляет ориентированный ациклический граф), множество Q «ресурсов», состояние S системы, задающее текущий «активный» узел для каждого процесса, и «распределение ресурсов» (подробности можно найти в указанных ниже работах).

ВОПРОС. Является ли состояние S «опасным»? Другими словами, существуют ли управляющие потоки для различных процессов, такие, что никакое распределение и перераспределение ресурсов не переведёт систему из состояния S в некоторое «конечное» состояние?

Источник [21, 508]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, даже если требования к размещению «правильные» и каждое из них относится не более чем к двум ресурсам. В указанной литературе приводятся дополнительные результаты по сложности этой задачи. В работе [175] даны результаты и алгоритмы, относящиеся к близкой задаче о тупиковых ситуациях.

А6. МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

[МП 1] ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ

УСЛОВИЕ. Заданы конечное множество X, состоящее из пар $(\bar{x}, b)$ (где $\bar{x}$ есть m-мерный целочисленный вектор, b – целое число), m-мерный целочисленный вектор $\bar{c}$ и целое число B.

ВОПРОС. Существует ли такой m-мерный вектор $\bar{y}$ с целыми координатами, что $\bar{x} \cdot \bar{y} \leqslant b$ для всех $(\bar{z}, b) \in X$ и $\bar{c} \cdot \bar{y} \geqslant B$ (где произведение m-мерных векторов $\bar{u} = (u_1, u_2, \dots, u_m)$ и $\bar{v} = (v_1, v_2, \dots, v_m)$ задаётся формулой $\bar{u} \cdot \bar{v} = \sum_{i=0}^{m} u_i v_i$)?

Источник [280, 55]. К этой задаче сводится 3-ВЫП. Во второй работе доказана принадлежность задачи классу *NP*.

Комментарий. Задача *NP*-полна в сильном смысле. Её вариант, когда ищется вектор $\bar{y}$, координаты которого принадлежат множеству $\{0, 1\}$ (так называемая задача 0-1 ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ), остаётся *NP*-полным даже в том случае, когда каждое b, все координаты векторов $\bar{x}$ и все координаты вектора $\bar{c}$ принадлежат множеству $\{0, 1\}$. *NP*-полной является задача с тем же условием, но в которой спрашивается о существовании такого вектора $\bar{y}$ с неотрицательными целыми компонентами, что $\bar{x} \cdot \bar{y} = b$ для всех $(\bar{x}, b) \in X$, или такого целочисленного вектора $\bar{y}$, что $\bar{x} \cdot \bar{y} \geqslant 0$ для всех $(\bar{x}, b)X$ (см. [463])[7].

[МП 2] **КВАДРАТИЧНОЕ ПРОГРАММИРОВАНИЕ (*)**

УСЛОВИЕ. Заданы конечное множество X, состоящее из пар $(\bar{x}, b)$ (где $\bar{x}$ есть m-мерный вектор с рациональными координатами, b – рациональ-

[7] Из результата, полученного Х. У. Ленстрой-младшим [1981], следует, что эта задача при каждом фиксированном $|U|$ разрешима за полиномиальное время. – ***Прим. ред.***

ное число), два m-мерных вектора $\bar{c}$ и $\bar{d}$ с рациональными координатами и рациональное число B.

ВОПРОС. Существует ли такой m-мерный вектор $\bar{y}$ с рациональными координатами, что $\bar{x} \cdot \bar{y} \leqslant b$ для всех пар $(\bar{x}, b) \in X$ и $\sum_{i=1}^{m}(c_i y_i^2 + d_i yi) \geqslant B$, где c_i, y_i и d_i – это i-е компоненты векторов $\bar{c}, \bar{y}$ и $\bar{d}$ соответственно?

Источник [463]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Если не все компоненты вектора $\bar{c}$ неотрицательны, то принадлежность задачи классу *NP* не установлена [290]. Если ограничения квадратичны, а целевая функция линейна (в противоположность сформулированной выше задаче), то задача *NP*-трудна [463]. Если к последней задаче добавить условие, что все компоненты вектора $\bar{y}$ целочисленны, то задача станет неразрешимой [258].

[МП 3] ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ С ПАРАМЕТРИЧЕСКОЙ ЦЕЛЕВОЙ ФУНКЦИЕЙ

УСЛОВИЕ. Заданы конечное множество X, состоящее из пар $(\bar{x}, b)$ (где $\bar{x}$ есть m-мерный целочисленный вектор, b – целое число), множество $J \subseteq \{1, 2, \dots, m\}$ и положительное рациональное число q.

ВОПРОС. Существует ли такой m-мерный вектор $\bar{c}$ с рациональными координатами, что $(\bar{c} \cdot \bar{c})^{1/2} \leqslant q$ и минимум по всем $\bar{y} \in Y$ сумм вида $\sum_{i \in J} c_j y_j$ больше, чем

$$\frac{1}{2} \max\{|c_j| : j \in J\} + \sum_{j \in J} \min\{0, c_j\},$$

где Y – множество всех m-мерных векторов с неотрицательными рациональными координатами, удовлетворяющих условию $\bar{x} \cdot \bar{y} \geqslant b$ для всех пар $(\bar{x}, b) \in X$?

Источник [259]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной для любого фиксированного $q > 0$. Она возникает при анализе влияния ошибок первого порядка в задаче линейного программирования.

[МП 4] ПРОДОЛЖЕНИЕ ДО ДОПУСТИМОГО БАЗИСА

УСЛОВИЕ. Заданы целочисленная матрица A порядка $m \times n$, $m < n$, вектор-столбец $\bar{a}$ размерности m и подмножество $S(|S| \leqslant m)$ множества столбцов матрицы A.

ВОПРОС. Существует ли для системы уравнений $A\bar{x} = a, \bar{x} \geqslant 0$, такой допустимый базис B, что B содержит все столбцы из множества S? (Допустимым базисом называется неособая, имеющая порядок $m \times m$ подматрица B матрицы A, такая, что $B^{-1}\bar{a} \geqslant 0$.)

Источник [396]. К этой задаче сводится ГАМИЛЬТОНОВ ЦИКЛ.

[МП 5] МИНИМАЛЬНОЕ ПО ВЕСУ РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ

УСЛОВИЕ. Заданы конечное множество X, состоящее из пар $(\bar{x}, b)$ (где x есть m-мерный целочисленный вектор, b – целое число) и положительное целое число $K \leqslant m$.

ВОПРОС. Существует ли такой m-мерный вектор $\bar{y}$ с рациональными координатами, что $\bar{y}$ имеет не более K отличных от нуля координат и для всех $(\bar{x}, b) \in X$ выполнено равенство $\bar{x} \cdot \bar{y} = b$?

Источник [152]. К этой задаче сводятся ТП-3.

Комментарий. Задача *NP*-полна в сильном смысле. Разрешима за полиномиальное время, если $K = m$.

[МП 6] **ОТКРЫТАЯ ПОЛУСФЕРА**

УСЛОВИЕ. Заданы конечное множество X m-мерных векторов и положительное целое число $K \leqslant |X|$.

ВОПРОС. Существует ли такой m-мерный вектор $\bar{y}$ с рациональными координатами, что $\bar{x} \cdot \bar{y} > 0$ по крайней мере для K векторов $\bar{x} \in X$?

Источник [267]. К этой задаче сводится МАКСИМАЛЬНАЯ 2-ВЫПОЛНИМОСТЬ.

Комментарий. Задача *NP*-полна в сильном смысле, но для любого фиксированного m разрешима за полиномиальное время, даже во «взвешенном» варианте. Аналогичные результаты имеют место для задачи ЗАМКНУТАЯ ПОЛУСФЕРА, в которой ищется вектор $\bar{y}$, удовлетворяющий условиям $\bar{x} \cdot \bar{y} \geqslant b$ не менее чем для K векторов $\bar{x} \in X$ [267]. Если $K = 0$ или $K = |X|$, обе задачи полиномиально эквивалентны задаче линейного программирования [448].

[МП 7] **K-ОПРЕДЕЛЯЕМОСТЬ**

УСЛОВИЕ. Заданы конечное множество X, состоящее из пар $(\bar{x}, b)$ (где $\bar{x}$ есть m-мерный целочисленный вектор, b – целое число) и положительное целое число $K \leqslant |X|$.

ВОПРОС. Существует ли подмножество $X' \in X$, такое, что $|X'| \leqslant K$ и для всех m-мерных векторов $\bar{y}$ с рациональными координатами из условия $\bar{x} \cdot \bar{y} \leqslant b$ для всех $(\bar{x}, b) \in X'$ следует, что $\bar{x} \cdot \bar{y} \leqslant b$ для всех $(\bar{x}, b) \in X$?

Источник [448]. К этой задаче сводится ТП-3.

Комментарий. Задача *NP*-полна в сильном смысле. При $K = |X| - 1$ эквивалентна задаче линейного программирования [448]. В этой же работе можно найти другие *NP*-полные задачи этого вида, в которых стандартная задача линейного программирования дополнена вопросом о выполнении нужного свойства для некоторого подмножества, состоящего из K ограничений.

[МП 8] **НЕСМЕЖНОСТЬ ВЕРШИН МНОГОГРАННИКА ЗАДАЧИ О КОММИВОЯЖЕРЕ**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и два гамильтоновых цикла C и C' в графе G.

ВОПРОС. Верно ли, что C и C' соответствуют несмежным вершинам «многогранника задачи о коммивояжере» для графа G?

Источник [411]. К этой задаче сводится 3-ВЫП.

Комментарий. Результат остаётся справедливым и в «несимметричном» случае, когда G – ориентированный граф, а C и C' – ориентированные гамильтоновы циклы. Аналогичные задачи о несмежности вершин многогранников, соответствующих паросочетанию и КЛИКЕ, могут быть решены за полиномиальное время [74].

[МП 9] **РЮКЗАК**

УСЛОВИЕ. Заданы конечное множество U, размер $s(u) \in Z^+$ и стоимость $v(u) \in Z^+$ каждого $u \in U$, положительные целые числа B и K.

ВОПРОС. Существует ли такое подмножество $U' \in U$, что

$$\sum_{i \in U'} s(u) \leqslant B \text{ и } \sum_{u \in U'} v(u) \geqslant K?$$

Источник [280]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача остаётся *NP*-полной, если $s(u) = v(u)$ для всех $u \in U$ (СУММА РАЗМЕРОВ). С помощью метода динамического программирования задача может быть решена за псевдополиномиальное время (см., например, [98, 324]).

[МП 10] **ЦЕЛОЧИСЛЕННЫЙ РЮКЗАК**

УСЛОВИЕ. Заданы конечное множество U, размер $s(u) \in Z^+$ и стоимость $v(u) \in Z^+$ каждого элемента $u \in U$, а также положительные целые числа B и K.

ВОПРОС. Можно ли так сопоставить каждому элементу $u \in U$ целое число $c(u)$, чтобы выполнялись неравенства

$$\sum_{u \in U} c(u) \cdot s(u) \leqslant B \text{ и } \sum_{u \in U} c(u) \cdot v(u) \geqslant K?$$

Источник [364]. К этой задаче сводится СУММА РАЗМЕРОВ.

Комментарий. Задача остаётся *NP*-полной, если $s(u) = v(u)$ для всех $u \in U$. Методом динамического программирования разрешима за псевдополиномиальное время. Если $|U| = 2$, то задача разрешима за полиномиальное время [210].

[МП 11] **НЕПРЕРЫВНЫЙ РЮКЗАК С КРАТНЫМ ВЫБОРОМ ЭЛЕМЕНТОВ**

УСЛОВИЕ. Заданы конечное множество U, размеры $s(u) \in Z^+$ и стоимость $v(u) \in Z^+$ каждого элемента из множества U, разбиение множества U на непересекающиеся подмножества $U_1, U_2, \ldots, U_m$, положительные целые числа B и K.

ВОПРОС. Можно ли так выбрать по одному элементу u_i из каждого подмножества $U_i, 1 \leqslant i \leqslant m$, и так сопоставить им рациональные числа $r_i, 0 \leqslant r_i \leqslant 1$, что будут выполнены неравенства

$$\sum_{i=1} r_i \cdot s(u_i) \leqslant B \text{ и } \sum_{i=1} r_i \cdot v(u_i) \geqslant K?$$

Источник [245]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача разрешима за псевдополиномиальное время, но остаётся *NP*-полной даже в том случае, если $|U_i| \leqslant 2, 1 \leqslant i \leqslant m$. Если

$|U_i| = 1$ при $1 \leqslant i \leqslant m$ или если потребовать только, чтобы $r_i \geqslant 0$, но не устанавливать для r_i верхней границы, то она разрешима за полиномиальное время «жадными» алгоритмами [246].

[МП 12] **ЧАСТИЧНО УПОРЯДОЧЕННЫЙ РЮКЗАК**

УСЛОВИЕ. Заданы конечное множество U, отношение частичного порядка $\lessdot$ на U, размер $s(u) \in Z^+$ и стоимость $v(u) \in Z^+$ каждого элемента $u \in U$, положительные целые числа B и K.

ВОПРОС. Существует ли такое подмножество $U' \subseteq U$, что если $u \in U'$ и $u' \lessdot u$, то $u' \in U'$ и выполнены неравенства

$$\sum_{u \in U'} s(u) \leqslant B \text{ и } \sum_{u' \in U'} v(u) \geqslant K?$$

Источник [152]. К этой задаче сводится КЛИКА. Рассматриваемая задача обсуждается также в работе [249].

Комментарий. Задача *NP*-полна в сильном смысле, даже если $s(u) = v(u)$ для всех $u \in U$. Общая задача разрешима за псевдополиномиальное время, если $\lessdot$ есть частичное "древовидное"упорядочение.

[МП 13] **ПОКООРДИНАТНЫЕ НЕРАВЕНСТВА МЕЖДУ ВЕКТОРАМИ**

УСЛОВИЕ. Заданы два множества $X = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_k\}$ и $Y = \{\bar{y}_1, \bar{y}_2, \dots, y_l\}$, состоящие из m-мерных целочисленных векторов.

ВОПРОС. Существует ли такой m-мерный целочисленный вектор $\bar{z}$, что число векторов $\bar{x}_i$, для которых выполнено неравенство $\bar{x}_i \geqslant \bar{z}$, не меньше числа таких векторов $\bar{y}_j$ для которых выполнено неравенство $\bar{y}_j \geqslant \bar{z}$ (здесь соотношение $\bar{u} \geqslant \bar{v}$ выполняется тогда и только тогда, когда у вектора $\bar{u}$ нет ни одной координаты, которая была бы меньше соответствующей координаты вектора $\bar{v}$)?

Источник [429]. К этой задаче сводится ПОКОМПОНЕНТНОЕ ВКЛЮЧЕНИЕ (с равными весами).

Комментарий. Задача остаётся *NP*-полной, даже если все компоненты векторов $\bar{x}_i$ или $\bar{y}_j$ принадлежат множеству $\{0, 1\}$.

А7. АЛГЕБРА И ТЕОРИЯ ЧИСЕЛ

А7.1. Задачи о делимости

[АТЧ 1] **КВАДРАТИЧНЫЕ СРАВНЕНИЯ**

УСЛОВИЕ. Заданы положительные целые числа a, b и c.

ВОПРОС. Существует ли положительное целое число $x < c$, такое, что $x^2 \equiv a (mod b)$?

Источник [373]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, даже если в условии даны разложение числа b на простые множители и решение данного сравнения по всем простым модулям, входящим в разложение числа b. Задача решается за полиномиальное время, если $c = \infty$ (т. е. отсутствует ограничение на x сверху) и задано разложение b на простые множители. В предположении истинности расширенной гипотезы Римана при простом b задача решается за полиномиальное время. В общем виде она тривиально разрешима за псевдополиномиальное время.

[АТЧ 2] **СИСТЕМА НЕСРАВНИМОСТЕЙ**

УСЛОВИЕ. Задан набор $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ упорядоченных пар положительных целых чисел, таких, что $a_i \leqslant b_i$ при $1 \leqslant i \leqslant n$.

ВОПРОС. Существует ли такое целое число x, что $x \equiv a_i (mod b_i)$, $1 \leqslant i \leqslant n$?

Источник [503]. К этой задаче сводится 3-ВЫП.

[АТЧ 3] **СОВМЕСТНАЯ ДЕЛИМОСТЬ ЛИНЕЙНЫХ МНОГОЧЛЕНОВ (*)**

УСЛОВИЕ. Заданы векторы $a_i = (a_i[0], \ldots, a_i[m])$ и $b_i = (b_i[0], \ldots, b_i[m])$, $1 \leqslant i \leqslant n$, с положительными целыми координатами.

ВОПРОС. Существуют ли такие положительные целые числа $x_1, x_2, \ldots, x_m$, что $a_i[0] + \sum_{j=1}^{m} a_i[j]x_j$, делит $b_i[0] + \sum_{j=1}^{m} b_i[j]x_j$?

Источник [348, 349]. К этой задаче сводится задача КВАДРАТНЫЕ ДИОФАНТОВЫ УРАВНЕНИЯ.

Комментарий. Не известно, принадлежит ли задача классу *NP*, но при любом фиксированном n она принадлежит *NP*. При любом фиксированном $n \leqslant 5$ задача *NP*-полна. В общем виде она неразрешима при условии, что компоненты векторов x_j могут принимать значения в кольце «целых» вещественного квадратичного расширения поля рациональных чисел. В указанных выше работах приведены другие, близкие результаты о разрешимости и неразрешимости.

[АТЧ 4] **СРАВНИТЕЛЬНАЯ ДЕЛИМОСТЬ**

УСЛОВИЕ. Заданы две последовательности $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_m$ положительных целых чисел.

ВОПРОС. Существует ли положительное целое число c, для которого число таких i, что c делит a_i, больше числа таких j, что c делит b_j?

Источник [429]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, если все b_j различны и все a_i различны [152].

[АТЧ 5] **ДЕЛИМОСТЬ ПОКАЗАТЕЛЬНОГО ВЫРАЖЕНИЯ (*)**

УСЛОВИЕ. Заданы две последовательности $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_m$ положительных целых чисел и целое число q.

ВОПРОС. Делится ли число $\prod_{j=1}^{m}(q^{b-j} - 1)$ на $\prod_{i=1}^{n}(q^{a_i} - 1)$?

Источник [429]. К этой задаче сводится 3-ВЫП.

Комментарий. Не известно, принадежит ли задача классу *NP* или классу $co - NP$, но она решается за псевдополиномиальное время обычным алгоритмом отыскания наибольшего общего делителя. Остаётся *NP*-трудной для любого фиксированного значения $q, |q| > 1$ даже тогда, когда a_i и b_i являются произведениями различных простых чисел.

[АТЧ 6] **НЕДЕЛИМОСТЬ ПРОИЗВЕДЕНИЯ МНОГОЧЛЕНОВ**

УСЛОВИЕ. Заданы последовательности $A_i =< (a_i[1], b_i[1]), \dots, (a_i[K], b_i[K]) >$, $1 \leqslant i \leqslant m$, таких пар целых чисел, что $b_i[j] = 0$, а также целое число N.

ВОПРОС. Верно ли, что многочлен $\prod_{i=1}^{m} \left(\sum_{j=1}^{k} a_i[j] \cdot z^{b_i[j]} \right)$ не делится на многочлен $z^N - 1$?

Источник [430, 431]. К этой задаче сводится 3-ВЫП. Доказательство принадлежности классу *NP* нетривиально и получено во второй из указанных работ.

Комментарий. Близкая задача: «Заданы две последовательности $< a_1, a_2, \dots, a_m >$ и $< b_1, b_2, \dots, b_n >$ положительных целых чисел, требуется выяснить, делится ли $\prod_{j=1}^{n}(z^{b_j} - 1)$ на $\prod_{i=1}^{m}(z^{a_i} - 1)$?» – также *NP*-полна [429].

[ДТЧ 7] **НЕТРИВИАЛЬНЫЙ НАИБОЛЬШИЙ ОБЩИЙ ДЕЛИТЕЛЬ (*)**

УСЛОВИЕ. Заданы последовательности $A_i =< (a_i[1], b_i[1]), \dots, (a_i[K], b_i[K]) >$, $1 \leqslant i \leqslant m$, пар целых чисел, таких, что $b_i[j] \geqslant 0$.

ВОПРОС. Верно ли, что наибольший общий делитель многочленов $\sum_{j=1}^{K} a_i[j] \cdot z^{b_i[j]}$, $1 \leqslant i \leqslant m$, имеет положительную степень?

Источник [430]. К этой задаче сводится 3-ВЫП.

Комментарий. Не известно, принадлежит ли задача *NP* или $co - NP$. Задача остаётся *NP*-трудной, если все $a_i[j]$ равны -1 или +1 [429] или если $m = 2$ [431]. Аналогичная задача, в условии которой дано также положительное целое число K и спрашивается верно ли, что наименьшее общее кратное заданных многочленов имеет степень меньше K при тех же ограничениях, является *NP*-трудной. С помощью стандартных алгоритмов обе задачи могут быть решены за псевдополиномиальное время.

А7.2. Разрешимость уравнений

[АТЧ 8] **КВАДРАТНЫЕ ДИОФАНТОВЫ УРАВНЕНИЯ**

УСЛОВИЕ. Заданы положительные целые числа a, b и c.

ВОПРОС. Существуют ли положительные целые числа x и y, такие, что $ax^2 + by = c$?

Источник [373]. К этой задаче сводится 3-ВЫП.

Комментарий. Диофантовы уравнения вида $ax^k = c$ и $\sum_{i=1}^{k} a_i x_i = c$ разрешимы в целых числах за полиномиальное время для произвольных k. Общая диофантова задача: «Задан полином от k переменных с целыми коэффициентами, имеется ли у него целый корень?» – неразрешима уже при $k = 13$ [377]. Однако данная задача может быть значительно обобщена (на системы уравнений с несколькими переменными), оставаясь при этом в классе *NP*, для этого достаточно, чтобы только одна переменная входила нелинейным образом в уравнение системы [192].

[АТЧ 9] **АЛГЕБРАИЧЕСКИЕ УРАВНЕНИЯ В ПОЛЕ GF[2]**

УСЛОВИЕ. Заданы полиномы $P_i(x_1, x_2, \dots, x_n)$, $1 \leqslant i \leqslant m$, над GF [2] (любой моном любого полинома есть либо 1, либо произведение различных x_i).

ВОПРОС. Существуют ли такие $u_1, u_2, \dots, u_n \in \{0, 1\}$, что $P_i(u_1, u_2, \dots, u_n) = 0$, $1 \leqslant i \leqslant m$, причём арифметические операции выполняются в поле GF [2], т. е. $1 + 1 = 0$, $1 \cdot 1 = 1$ и т. д.?

Источник [129]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной даже в том случае, когда ни в одном из многочленов нет мономов, содержащих более двух переменных [524]. Легко решается за полиномиальное время, если в многочленах нет мономов, содержащих более одной переменной или если $i = 1$. Вариант этой задачи, в котором $U - j$ принимают значения в алгебраическом замыкании GF [2], является *NP*-трудным, даже если в многочленах нет мономов, содержащих более двух переменных [129].

[АТЧ 10] **КОРНИ ПО МОДУЛЮ РАВНЫЕ 1 (*)**

УСЛОВИЕ. Задан набор упорядоченных пар $(a[i], b[i])$, $1 \leqslant i \leqslant n$, целых чисел, причём $b[i] \geqslant 0$.

ВОПРОС. Верно ли, что многочлен $\sum_{i=1}^{n} a[i] \cdot z^{b[i]}$ имеет корень на единичной окружности комплексной плоскости? Иными словами, существует ли комплексное число q, $|q| = 1$, такое, что

$$\sum_{i=1}^{n} a[i] \cdot q^{b[i]} = 0?$$

Источник [431]. К этой задаче сводится 3-ВЫП.

Комментарий. Не известно, принадлежит ли задача *NP* или $co - NP$.

[АТЧ 11] **ЧИСЛО КОРНЕЙ ПРОИЗВЕДЕНИЯ ПОЛИНОМОВ (*)**

УСЛОВИЕ. Заданы последовательности $A_i =< (a_i[1], b_i[1]), \dots, (a_i[k], b_i[k]) >$ пар целых чисел, причём все $b_i[j] \geqslant 0$, а также положительное целое число K.

ВОПРОС. Верно ли, что полином $\prod_{i=1}^{m} \left(\sum_{j=1}^{k} a_i[j] \cdot z^{b_i[j]} \right)$ имеет менее K различных комплексных корней?

Источник [430]. К этой задаче сводится 3-ВЫП.

Комментарий. Не известна принадлежность задачи ни *NP*, ни $co - NP$. Задача остаётся *NP*-трудной, если каждое $a_i[j]$ равно -1 или +1 или если в условии имеется число M и спрашивается, существует ли у произведения многочленов не более K корней кратности M [430].

[АТЧ 12] **ПЕРИОДИЧЕСКОЕ РЕШЕНИЕ РЕКУРРЕНТНОГО УРАВНЕНИЯ (*)**

УСЛОВИЕ. Задана последовательность $(c_i, b_i), 1 \leqslant i \leqslant m$, упорядоченных пар целых чисел с положительными b_i.

ВОПРОС. Существует ли такая последовательность целых чисел $a_0, a_1, \ldots, a_{n-1}, n \geqslant \max\{b_i\}$, что бесконечная последовательность $a_0, a_1, \ldots$, определяемая рекуррентным уравнением

$$a_i = \sum_{j=1}^{m} c_j \cdot a_{i-b_j},$$

для всех $i \geqslant n$ удовлетворяет соотношению $a_i \equiv a_{i(mod n)}$?

Источник [431]. К этой задаче сводится 3-ВЫП.

Комментарий. Не известна принадлежность задачи ни классу *NP*, ни $co - NP$. В указанной работе приведены близкие результаты.

А7.3. Разное

[АТЧ 13] **ВЫЧИСЛЕНИЕ ПЕРМАНЕНТА(*)**

УСЛОВИЕ. Заданы (0-1)-матрица M порядка $n \times n$ и положительное целое $K \leqslant n!$

ВОПРОС. Верно ли, что значение перманента матрицы M равно K?

Источник [522]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача является *NP*-трудной, но не известно, принадлежит ли она классу *NP*. То же самое справедливо для варианта, в котором спрашивается, верно ли, что значение перманента не больше K или не меньше K. Задача вычисления значения перманента матрицы M KP-полна.

[АТЧ 14] **ИНТЕГРАЛ ОТ ПРОИЗВЕДЕНИЯ КОСИНУСОВ**

УСЛОВИЕ. Задана последовательность $(a_1, a_2, \ldots, a_n)$ целых чисел.

ВОПРОС. Верно ли, что $\int_0^{2\pi} \left(\prod_{i=1}^{n} \cos(a_i\theta)\right) d\theta = 0$?

Источник [429]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Задача разрешима за псевдополиномиальное время. В указанной работе приведены близкие результаты о сложности задач интегрирования.

[АТЧ 15] **ТОЧКА РАВНОВЕСИЯ**

УСЛОВИЕ. Заданы набор $\{F_i : 1 \leqslant i \leqslant n\}$ многочленов с целыми коэффициентами от переменных $x_1, x_2, \ldots, x_n$ и конечные области определения переменных $M_i \subseteq Z, 1 \leqslant i \leqslant n$.

ВОПРОС. Существует ли такая последовательность $y_1, y_2, \ldots, y_n, y_i \in M_i, 1 \leqslant i \leqslant n$, что для всех $y \in M_i$ выполнены неравенства

$$F_i(y_1, y_2, \ldots, y_{i-1}, y_i, y_{i+1}, \ldots, y_n) \geqslant F_i(y_1, y_2, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_n)?$$

Источник [463]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, даже если $M_i = \{0, 1\}, 1 \leqslant i \leqslant n$.

[АТЧ 16] **УНИФИКАЦИЯ С КОММУТАТИВНЫМИ ОПЕРАТОРАМИ**

УСЛОВИЕ. Заданы множество переменных V, множество констант C, упорядоченные пары (e_i, f_i) «выражений». (Выражением называется либо переменная из V, либо константа из C, либо $(e+f)$, где e и f – выражения.)

ВОПРОС. Можно ли так сопоставить каждой переменной $v \in V$ выражение $I(v)$, не содержащее переменных, что $I(e_i) \equiv I(f_i)$ при $1 \leqslant i \leqslant n$ (где $I(e)$ – выражение, получающееся заменой каждой переменной v, входящей в e, выражением $I(v)$, а отношение тождества определяется так: $e \equiv f$, если $e = f$ или если $e = (a+b)$, $f = (c+d)$ и либо $a \equiv c, b \equiv d$, либо $a \equiv d, b \equiv c$)?

Источник [479]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, даже если каждое выражение e_i и f_i содержит не более 7 вхождений постоянных и переменных. Вариант задачи, в котором оператор $+$ не коммутативен, т. е. $e \equiv f$ и тогда и только тогда, когда $e = f$, разрешим за полиномиальное время [421].

[АТЧ 17] **УНИФИКАЦИЯ ДЛЯ КОНЕЧНО-ОПРЕДЕЛЁННЫХ АЛГЕБР**

УСЛОВИЕ. Задана алгебра A, конечно-определённая с множеством образующих G, набором операторов O различных конечных арностей и набором Γ определяющих соотношений на правильно построенных выражениях над G и O. Заданы также два правильно построенных выражения e и f над G и O и множество переменных V (детали см. в указанных ниже работах).

ВОПРОС. Можно ли каждой переменной $v \in V$ сопоставить единственный «терм» $I(v)$ над G и O так, что если $I(e)$ и $I(f)$ обозначают выражения, получающиеся заменой каждой переменной в e и f соответствующим термом, то $I(e)$ и $I(f)$ представляют один и тот же элемент A?

Источник [300, 299]. К этой задаче сводится 3-ВЫП. Доказательство того, что задача принадлежит классу *NP*, нетривиально и содержится во второй работе.

Комментарий. Задача остаётся *NP*-полной, если только e или только f содержит символы переменных. В работе [301] изложены кванторные версии рассматриваемой задачи, которые являются полными в классе $P - SPACE$ и в различных уровнях полиномиальной иерархии.

[АТЧ 18] **ПРЕДСТАВИМОСТЬ ЦЕЛОЧИСЛЕННЫМ ВЫРАЖЕНИЕМ**

УСЛОВИЕ. Заданы целочисленное выражение e над операторами U и $+$ (целочисленное выражение определяется рекурсивно: двоичное представление числа $n \in Z^+$ есть целочисленное выражение, представляющее множество $\{n\}$, если f и g есть целочисленные выражения, представляющие множества F и G, то $f \cup g$ есть целочисленное выражение, представляющее множество $F \cup G$, а $f + g$ есть целочисленное выражение, представляющее множества $\{m+n : m \in F$ и $n \in G\}$, а также положительное целое число K.

ВОПРОС. Принадлежит ли K множеству, представляемому выражением e?

Источник [503]. К этой задаче сводится СУММАРНЫЙ ВЕС ПОДМНОЖЕСТВА.

Комментарий. Близкая задача НЕЭКВИВАЛЕНТНОСТЬ ЦЕЛОЧИСЛЕННЫХ ВЫРАЖЕНИЙ: «По двум заданным целочисленным выражениям e и f определить, представляют ли они различные множества?» – является NP-трудной и на самом деле полна в классе $\sum_2^p$ полиномиальной иерархии ([503, 500], см. также разд. 7.2). Если допустить применение оператора «$\rceil$», где $\rceil e$ представляет множество всех целых чисел, не представляемых e, то обе задачи о представимости и неэквивалентности становятся $P-SPACE$-полными, см. [503].

А8. ИГРЫ И ГОЛОВОЛОМКИ

[ИГ 1] **ОБОБЩЁННЫЙ ГЕКС (*)**

УСЛОВИЕ. Задан граф $G = (V, E)$, в нём выделены две вершины $s, t \in V$.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на графе G?

Описание игры. Игроки поочерёдно выбирают вершину из $V \setminus \{s, t\}$; вершины, выбранные игроком 1, «окрашиваются» в синий цвет, а выбранные игроком 2 – в красный. Игра продолжается до тех пор, пока не будут окрашены все выбранные вершины, причём игрок 1 выигрывает тогда и только тогда, когда найдётся путь из s в t в графе G, проходящий только через синие вершины.

Источник [119]. К этой задаче сводятся БУЛЕВСКИЕ ФОРМУЛЫ С КВАНТОРАМИ (БФК) (см. разд. А9.2).

Комментарий. Задача является $P-SPACE$-полной. Версия задачи, в которой игроки поочерёдно выбирают рёбра, а не вершины (известная под названием «переключательная игра Шеннона на рёбрах»), разрешима за полиномиальное время [62]. Если граф G ориентирован и игрок 1 стремится получить синий ориентированный путь из s в t, то обе задачи (т. е. выбор вершин и выбор рёбер) являются $P-SPACE$-полными [119].

[ИГ 2] **ОБОБЩЁННАЯ ГЕОГРАФИЯ (*)**

УСЛОВИЕ. Задан ориентированный граф $G = (V, A)$ и выделенная в нём вершина $v_0 \in V$.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на G?

Описание игры. Игроки поочерёдно выбирают новую дугу из A. Первая дуга должна иметь своим началом вершину v_0, а каждая последующая выбранная дуга должна иметь своим началом конец последней выбранной до этого дуги. Игрок, который первым окажется не в состоянии выбрать новую дугу, проигрывает.

Источник [473]. К этой задаче сводится БУЛЕВСКИЕ ФОРМУЛЫ С КВАНТОРАМИ (БФК) (см. разд. А9.2).

Комментарий. Задача $P-SPACE$-полна, даже если граф двудольный, планарный и имеющий степени захода и выхода, не превосходящие 2, и степени вершин не превосходят 3 (ПЛАНАРНАЯ ГЕОГРАФИЯ) [345]. Игра является обобщением известной игры «география», или «города», в которой игроки поочерёдно называют города, причём название каждого города должно начинаться с той буквы, на которую оканчивается название предыдущего города.

[ИГ 3] **ОБОБЩЁННЫЕ «КЕЙЛИ»(*)**

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на G?

Описание игры. Игроки поочерёдно выбирают вершину в графе, причём эта вершина и все ей смежные выбрасываются из графа. Игрок 1 выигрывает тогда и только тогда, когда игрок 2 оказывается первым игроком, который не может выбрать вершину, так как они уже все выброшены.

Источник [473]. К этой задаче сводится БФК (см. разд. А9.2).

Комментарий. Задача $P-SPACE$-полная. Рассмотрим версию задачи, в которой $G = (V_1 \cup V_2, E)$ – двудольный граф, каждое ребро которого имеет одну вершину в V_1, а другую – в V_2, и, кроме того, игрок i может выбирать вершины из множества V, (причём все смежные вершины, как и раньше, выбрасываются), эта задача также является $P-SPACE$-полной. Описание игры в «кейли», на которой основано приведённое выше обобщение, можно найти в [89].

[ИГ 4] **ПОСЛЕДОВАТЕЛЬНОЕ ПРИСВОЕНИЕ ЗНАЧЕНИЙ ИСТИННОСТИ (*)**

УСЛОВИЕ. Заданы последовательность переменных $U =< u_1, u_2, ..., u_n >$ и семейство C дизъюнкций над U (так же как в задаче ВЫПОЛНИМОСТЬ).

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на U и C?

Описание игры. Игроки поочерёдно присваивают значение истинности переменным из U, причём на i-м ходу игрок 1 присваивает значение переменной u_{2i-1} а игрок 2 – переменной u_{2i}. Игрок 1 выигрывает тогда и только тогда, когда в результате присвоения переменным значений истинности все дизъюнкции из C оказываются выполненными.

Источник [503]. К этой задаче сводится БФК (см. разд. А9.2).

Комментарий. Задача $P-SPACE$-полная, даже если все дизъюнкции в C содержат только три литерала. Она разрешима за полиномиальное время, если каждая дизъюнкция содержит не более двух литералов [474].

[ИГ 5] **ПРИСВОЕНИЕ ИСТИННОСТИ С РАЗБИЕНИЕМ МНОЖЕСТВА ПЕРЕМЕННЫХ (*)**

УСЛОВИЕ. Задано множество переменных U и семейство C дизъюнкций над U.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на U и C?

Описание игры. Игроки поочерёдно выбирают переменные из U до тех пор, пока не будут выбраны все переменные. Игрок 1 выигрывает тогда и только тогда, когда набор значений «истина» у переменных игрока 1 и значений «ложь» у переменных игрока 2 приводит к выполнению всех дизъюнкций из C.

Источник [473]. К этой задаче сводится БФК (см. разд. А9.2).

Комментарий. Задача $P-SPACE$-полная, даже если все дизъюнкции содержат литералы без отрицаний (т. е. не содержат литералов вида $\bar{u}$, если $u \in U$). Аналогичные результаты для нескольких других игр на логических выражениях можно найти в [473].

[ИГ 6] **ФИЛЬТР (*)**

УСЛОВИЕ. Заданы два семейства A и B подмножеств конечного множества X, причём A и B не имеют общих подмножеств.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на A, B и X?

Описание игры. Игроки поочерёдно выбирают элементы из X до тех пор, пока множество X' всех выбранных элементов не будет пересекаться либо с каждым подмножеством из A, либо с каждым подмножеством из B. Игрок 1 выигрывает тогда и только тогда, когда множество X' всех выбранных элементов пересекается с каждым подмножеством из B, причём если последний ход сделан игроком 1, то X' не должно пересекаться с каждым подмножеством из A.

Источник [473]. К этой задаче сводится БФК (см. А9.2).

Комментарий. Задача $P-SPACE$-полная.

[ИГ 7] **НАКРЫВАЮЩЕЕ МНОЖЕСТВО (*)**

УСЛОВИЕ. Задано семейство C подмножеств базисного множества B.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на C и B?

Описание игры. Игроки поочерёдно выбирают новые элементы из B до тех пор, пока для каждого $c \in C$ не окажется выбранным какой-то элемент из c. Игрок, ход которого приводит к этой ситуации, проигрывает.

Источник [473]. К этой задаче сводится БФК (см. разд. А9.2).

Комментарий. Задача $P-SPACE$-полная, даже если все множества в C содержат не более двух элементов (исходная задача НАКРЫВАЮЩЕЕ МНОЖЕСТВО в этом случае полиномиально разрешима). Если поменять местами выигрывающего и проигрывающего, задача будет $P-SPACE$-полной, даже когда все множества в C содержат не более трёх элементов.

[ИГ 8] **МАКСИМАЛЬНОЕ ВЗВЕШЕННОЕ ПАРОСОЧЕТАНИЕ (*)**

УСЛОВИЕ. Заданы граф $G=(V,E)$, веса $w(e) \in Z^+$ для всех $e \in E$ и граница $B \in Z^+$.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на G?

Описание игры. Игроки поочерёдно выбирают новое ребро из E, при этом выбранное ребро не должно иметь общих концов с выбранными ранее рёбрами. Как только сумма весов всех выбранных рёбер превосходит B, игрок 1 выигрывает.

Источник [102]. К этой задаче сводится БФК (см. разд. А9.2).

Комментарий. Задача $P-SPACE$-полная, хотя соответствующая задача о максимальном взвешенном паросочетании разрешима за полиномиальное время [324].

[ИГ 9] **УНИЧТОЖЕНИЕ (*)**

УСЛОВИЕ. Заданы ориентированный ациклический граф $G=(V,A)$, семейство $\{A_i|1\leqslant i\leqslant r\}$ (возможно, пересекающихся) подмножеств множества A, функция f_0, отображающая V в $\{0,1,2,\dots,r\}$, где $f_0(v)=i>0$ означает, что пометка типа i присвоена вершине v, а $f_0(v)=0$ означает, что v не помечена.

ВОПРОС. Имеет ли игрок 1 форсированный выигрыш в описанной ниже игре на G?

Описание игры. Позицией называется $f:V\to\{0,1,\dots,r\}$, причём f_0 – это начальная позиция. Игроки делают ходы по очереди. Ход каждого игрока состоит в том, что он выбирает некоторую вершину $v\in V$ с пометкой $f(v)>0$ и дугу $(v,w)\in A_{f(v)}$ и затем переносит пометку с вершины v на вершину w. Точнее, новая позиция f' имеет тот же вид, что и f, за исключением того, что $f'(v)=0$ и

$$f'(w)=\begin{cases} 0, & \text{если } f(w)>0;\\ f(v), & \text{если } f(w)=0.\end{cases}$$

(Это значит, что если $f(w)>0$, то пометка, перенесённая на w, и пометка, присвоенная ранее вершине w, «взаимно уничтожаются».) Игрок 1 выигрывает тогда и только тогда, когда игрок 2 оказывается неспособным сделать очередной ход.

Источник [129]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ.

Комментарий. Задача *NP*-трудна и принадлежит $P-SPACE$, но не известно, является ли она $P-SPACE$-полной. Она остаётся *NP*-трудной, даже если $r=2$ и $A_1\cap A_2=\varnothing$. Задача разрешима за полиномиальное время, если $r=1$ [128]. Аналогичные результаты, устанавливающие *NP*-трудность для других игр с пометками на ориентированных графах, можно найти в [129].

[ИГ 10] **ШАШКИ НА ДОСКЕ $N\times N$ (*)**

УСЛОВИЕ. Заданы: положительное число N; разбиение чёрных полей на шахматной доске размером $N\times N$ на три группы: (а) пустые поля, (б) поля, занятые чёрными дамками, и (в) поля, занятые красными дамками; информация, чей ход первый (красных или чёрных).

ВОПРОС. Имеют ли чёрные форсированный выигрыш в заданной позиции в игре, играемой по стандартным правилам шашек на расширенной доске размером $N \times N$ (при соответственно увеличенном числе шашек)?

Источник [127]. К этой задаче сводится ПЛАНАРНАЯ ГЕОГРАФИЯ.

Комментарий. Задача $P - SPACE$-трудна. При некоторых правилах она $P - SPACE$-полна. Близкая задача, в которой спрашивается, могут ли чёрные взять все шашки красных за один ход, разрешима за полиномиальное время.

[ИГ 11] **ГО НА ДОСКЕ $N \times N$ (*)**

УСЛОВИЕ. Заданы: положительное число N; разбиение «точек» на доске для игры в «го» размером $N \times N$ на три группы: (а) пустые, (б) занятые чёрными фишками, (в) занятые белыми фишками; информация, кто ходит первым, белые или чёрные.

ВОПРОС. Имеют ли белые форсированный выигрыш в заданной позиции в игре, играемой по стандартным правилам «го» на расширенной доске?

Источник [345]. К этой задаче сводится ПЛАНАРНАЯ ГЕОГРАФИЯ.

Комментарий. Задача $P - SPACE$-трудна.

[ИГ 12] **ЛЕВЫЙ – ПРАВЫЙ ХАКЕНБУШ**

УСЛОВИЕ. Заданы: отдельный предмет из «гарнитура красного дерева», представимый в виде связного графа $G = (V, E)$ с выделенной вершиной $v \in V$, называемой «основанием»; разбиение рёбер графа на множества L и R, где L – множество рёбер, содержащих вершину v и называемых «ножками», а $R = E \setminus L$, причём каждая «ножка» имеет общую вершину не более чем с одним ребром из R, называемым её «стойкой» (не все рёбра в R, однако, являются «стойками»); положительное целое число K.

ВОПРОС. Найдется ли «цена» игры ЛЕВЫЙ-ПРАВЫЙ ХАКЕНБУШ на графе G, меньшая и равная 2^{-K}?. Описание игры и определение «цены игры» см. в [89].

Источник [37]. К этой задаче сводится ПОКРЫТИЕ МНОЖЕСТВ.

Комментарий. Задача остаётся *NP*-полной даже для двудольного графа G. Она разрешима за полиномиальное время, если граф G – дерево. Следствием этого результата является то, что задача выяснения, имеется ли в этой игре выигрыш для игрока 1, является *NP*-трудной.

[ИГ 13] **ОБЛИЦОВКА КВАДРАТА**

УСЛОВИЕ. Заданы: множество красок C; семейство $T \in C^4$ «керамических плиток», где $< a, b, c, d >$ обозначает плитку, у которой верх, правая сторона, низ, левая сторона окрашены соответственно в цвета a, b, c, d; положительное целое число $N \leqslant |C|$.

ВОПРОС. Существует ли покрытие квадрата размером плитками из $N \times N$, т. е. можно ли поставить в соответствие каждой паре $(i, j), 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant N$, некоторую плитку $A = A(i, j) \in T$ так, что, во-первых, если $F(i, j) =< a, b, c, d >$ и $f(i + 1, j) =< a', b', c', d' >$, то $a = c'$, и, во-вторых, если $f(i, j) =< a, b, c, d >$ и $f(i, j + 1) =< a', b', c', d' >$, то $b = d'$?

Источник [154]. К этой задаче сводится ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ПУТЬ.

Комментарий. Вариант задачи, когда спрашивается, можно ли с помощью множества T покрыть плитками всю плоскость $(Z \times Z)$ «периодически» с периодом меньшим, чем N, также является *NP*-полной задачей. В общем случае задача «выяснить, можно ли с помощью некоторого множества плиток покрыть всю плоскость», неразрешима [36], точно так же как и задача «выяснить, можно ли с помощью некоторого множества плиток покрыть плоскость периодически».

[ИГ 14] **СОСТАВЛЕНИЕ КРОССВОРДА**

УСЛОВИЕ. Заданы конечное множество слов $W \in \Sigma^*$ и матрица A из нулей и единиц размером $n \times n$.

ВОПРОС. Можно ли составить кроссворд размером $n \times n$ из слов множества W так, чтобы слова записывались в клетки, соответствующие нулям в A? Другими словами, если E – это множество пар (i, j), таких, что $A_{ij} = 0$, то существует ли отображение $f : E \to \Sigma$, такое, что буквы, приписываемые любой горизонтальной или вертикальной последовательности (максимальной длины) элементов из E, образуют слово из W?

Источник [342]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной, даже если все элементы матрицы A есть нули.

[ИГ 15] **ОБОБЩЁННАЯ ИГРА «ИНСТАНТ ИНСАНИТИ»**

УСЛОВИЕ. Заданы конечное множество C «красок» и множество Q кубиков, $|Q| = |C|$ и все грани кубиков из Q окрашены в заданные цвета из C.

ВОПРОС. Можно ли кубики Q выстроить в один вертикальный столбик так, чтобы каждый цвет из C появлялся ровно один раз на каждой из четырёх сторон столбика?

Источник [452]. К этой задаче сводится ТОЧНОЕ ПОКРЫТИЕ.

Комментарий. Имеется соответствующая игра двух лиц, в которой игроки по очереди помещают очередной кубик на верх столбика, причём игрок 1 пытается построить столбик, удовлетворяющий сформулированному выше условию, а игрок 2 препятствует этому. Задача «выяснить, имеет ли игрок 1 форсированный выигрыш в этой игре», является $P - SPACE$-полной. Название ИНСТАНТ ИНСАНИТИ (INSTANT INSANITY) – торговый знак, принадлежащий фирме Parker Brothers, Inc.

A9. ЛОГИКА

A9.1. Пропозициональная логика

[ЛОГ 1] **ВЫПОЛНИМОСТЬ**

УСЛОВИЕ. Заданы множество переменных U и набор C дизъюнкций над U (определение см. в разд. 2.6).

ВОПРОС. Существует ли для набора дизъюнкций C выполняющий набор истинностных значений?

Источник [91]. *NP*-полнота задачи доказана в теореме Кука.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если $|c| = 3$ для всех $c \in C$ (3-ВЫП) или если $|c| \leqslant 3$ для всех $c \in C$, а также если для каждой переменной $u \in U$ имеется не более трёх дизъюнкций, содержащих u или $\bar{u}$. Задача *NP*-полна и тогда, когда $|c| \leqslant 3$ для всех $c \in C$ и двудольный граф $G = (V, E)$ планарен (ПЛАНАРНАЯ 3-ВЫП) (здесь $V = U \cup C$, а E состоит из таких пар $\{u, c\}$, что u или $\bar{u}$ принадлежит дизъюнкции c). Общая задача разрешима за полиномиальное время, если $|c| \leqslant 2$ для всех $c \in C$ (см., например, [114]).

[ЛОГ 2] **3-ВЫПОЛНИМОСТЬ (3-ВЫП)**

УСЛОВИЕ. Заданы множество переменных U и такой набор C дизъюнкций над U, что $|c| = 3$ для каждой дизъюнкции $c \in C$.

ВОПРОС. Существует ли для набора дизъюнкций C выполняющий набор истинностных значений?

Источник [91]. К этой задаче сводится ВЫПОЛНИМОСТЬ.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если каждая дизъюнкция содержит только сами переменные или только их отрицания (МОНОТОННАЯ 3-ВЫП) [174] или если для каждой переменной не более 5 дизъюнкций из C содержат или литерал u или $\bar{u}$.

[ЛОГ 3] **3 ВЫП ПРИ РАЗЛИЧНЫХ ЛИТЕРАЛАХ**

УСЛОВИЕ. Заданы множество переменных U и такой набор C дизъюнкций над U, что $|c| = 3$ для каждой дизъюнкции $c \in C$.

ВОПРОС. Существует ли для набора C такой выполняющий набор истинностных значений, что в каждой дизъюнкции из C найдётся по крайней мере один истинный и один ложный литерал?

Источник [474]. К этой задаче сводится 3-ВЫП.

[ЛОГ 4] **3-ВЫП ПРИ ОДНОМ ИСТИННОМ ЛИТЕРАЛЕ**

УСЛОВИЕ. Заданы множество переменных U и такой набор C дизъюнкций над U, что $|c| - 3$ для каждой $c \in C$.

ВОПРОС. Существует ли для U такой выполняющий набор истинностных значений, что для каждой дизъюнкции в C имеется в точности один истинный литерал?

Источник [474]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если ни одна $c \in C$ не содержит отрицаний переменных.

[ЛОГ 5] **МАКСИМАЛЬНАЯ 2-ВЫПОЛНИМОСТЬ**

УСЛОВИЕ. Заданы множество переменных U; такой набор C дизъюнкций над U, что $|c| = 2$ для всех $c \in C$; положительное целое число $K \leqslant C$.

ВОПРОС. Существует ли для U набор истинностных значений, выполняющий не менее K дизъюнкций из набора C?

Источник [157]. К этой задаче сводится 3-ВЫП.

Комментарий. Разрешима за полиномиальное время при $K = |C|$ (см., например, [114]).

[ЛОГ 6] **ОБОБЩЁННАЯ ВЫПОЛНИМОСТЬ**

УСЛОВИЕ. Заданы положительные целые числа $k_1, k_2, \dots, k_m$, последовательность $S =< R_1, R_2, \dots, R_m >$ подмножеств $R_i \subseteq \{T, F\}^{k_i}$, множество переменных U и для всех i, $1 \leqslant i \leqslant m$, множества C_i, состоящие из наборов по k_i переменных из множества U.

ВОПРОС. Существует ли такой набор истинностных значений $t : U \to \{T, F\}$, что для всех i, $1 \leqslant i \leqslant m$, и для всех наборов $(u[1], u[2], \dots, u[k_i]) \in C_i$ имеет место

$$(t(u[1]), t(u[2]), \dots, t(u[k_i])) \in R_i?$$

Источник [474]. К этой задаче сводится 3-ВЫП.

Комментарий. Если для заданной последовательности S не выполняется одна из шести перечисленных ниже возможностей, то задача *NP*-полна, в противном случае она разрешима за полиномиальное время. Упомянутые возможности таковы:

(1) каждое R_i содержит $\{T\}^{k_i}$;

(2) каждое R_i содержит $\{F\}^{k_i}$;

(3) каждое R_i логически «эквивалентно» некоторому выражению, имеющему конъюнктивную нормальную форму и содержащему в каждой дизъюнкции по крайней мере одно отрицание;

(4) каждое R_i логически «эквивалентно» некоторому выражению, имеющему конъюнктивную нормальную форму и содержащему в каждой дизъюнкции по крайней мере один литерал без отрицания;

(5) каждое R_i логически «эквивалентно» некоторому выражению, имеющему конъюнктивную нормальную форму и содержащему в каждой дизъюнкции не более двух литералов;

(6) каждое R_i есть «множество решений» некоторой системы уравнений над GF [2].

NP-полнота задач 3-ВЫП, 3-ВЫП ПРИ РАЗЛИЧНЫХ ЛИТЕРАЛАХ и 3-ВЫП ПРИ ОДНОМ ИСТИННОМ ЛИТЕРАЛЕ вытекает из приведённой классификации. Если допустить принадлежность k_i-мерных наборов из множества C_i множеству $(U \cap)\{T, F\}^{k_i})$ («формулы с константами»), то задача *NP*-полна даже в том случае, если выполнены условия (1) или (2), но по-прежнему разрешима за полиномиальное время, если выполняется одно из условий (3) – (6). Квантифицированная версия задачи «с константами», в которой дополнительно задана последовательность $Q_1, Q_2, \dots, Q_n$ кванторов $\forall$ и $\exists$ и вопрос заключается в истинности формулы

$$(Q_1, u_1)(Q_2, u_2) \dots (Q_n u_n)\ [c \in R_i \text{ для всех } c \in C_i, 1 \leqslant i \leqslant m],$$

является $P{-}SPACE$-полной, даже если зафиксировать последовательность S, не удовлетворяющую свойствам (3)–(6). Если последовательность S удовлетворяет одному из свойств (3)–(6), то последняя задача разрешима за полиномиальное время.

[ЛОГ 7] **ВЫПОЛНИМОСТЬ БУЛЕВСКИХ ВЫРАЖЕНИЙ**

УСЛОВИЕ. Заданы множество переменных U и подмножество B множества всех 16 возможных двуместных булевских связок, а также правильно построенное булевское выражение E над U и B.

ВОПРОС. Существует ли набор истинностных значений переменных множества U, при котором выражение E выполнено?

Источник [91]. Общая сводимость[8].

Комментарий. Задача остаётся *NP*-полной, если B состоит только из четырёх булевских связок $\{\wedge, \vee, \rightarrow, \neg\}$ или любого другого полного набора булевских связок. Задача *NP*-полна также для любого неполного множества булевских связок, содержащего в качестве подмножества одно из следующих множеств: $\{\not\rightarrow\}, \{\not\leftarrow\}, \{\not\equiv, \vee\}$ или $\{\not\equiv, \wedge\}$, см. [341]. Задача разрешима за полиномиальное время для любого неполного набора булевских связок, не включающего ни одного из четырёх перечисленных выше множеств.

[ЛОГ 8] **ОТСУТСТВИЕ ТАВТОЛОГИИ**

УСЛОВИЕ. Задано булевское выражение E на множестве переменных U, в котором используются булевские связки $\neg$ (не), «$\vee$» (или), «$\wedge$» (и) и «$\rightarrow$» (следует).

ВОПРОС. Верно ли, что E не тавтология, т. е. существует ли такой набор истинностных значений переменных множества U, что E принимает значение «ложь»?

Источник [91]. К этой задаче сводится ВЫПОЛНИМОСТЬ.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда E имеет «дизъюнктивную нормальную форму», каждый дизъюнктивный член которой содержит не более трёх литералов.

[ЛОГ 9] **МИНИМАЛЬНАЯ ДИЗЪЮНКТИВНАЯ НОРМАЛЬНАЯ ФОРМА**

УСЛОВИЕ. Заданы множество переменных $U = \{u_1, u_2, \dots, u_n\}$, множество $A \subseteq \{T, F\}^n$ «наборов истинностных значений» и положительное целое число K.

ВОПРОС. Существует ли над множеством переменных U такая дизъюнктивная нормальная форма E, содержащая не более K дизъюнктивных членов, что E истинна только для наборов истинностных значений из множества A?

Источник [173]. К этой задаче сводится МИНИМАЛЬНОЕ ПОКРЫТИЕ.

Комментарий. Вариант задачи, в котором условие содержит полную таблицу истинностных значений, т. е. два непересекающихся подмножества A и B, $A, B \subseteq \{T, F\}^n$, $A \cup B = \{T, F\}^n$, и спрашивается, будет ли E истинно для наборов истинностных значений из множества A и ложно для наборов истинностных значений из множества B, также *NP*-полон, хотя в последнем случае длина записи условия может быть гораздо больше [374].

---

[8] Имеется в виду сводимость, описанная при доказательстве теоремы Кука, см. стр. 57. – *Прим. ред.*

[ЛОГ 10] **ПОЛНОТА НАБОРА СВЯЗОК**

УСЛОВИЕ. Заданы множество переменных U и набор C правильно построенных булевских выражений над U.

ВОПРОС. Верно ли, что набор C полон? Набор C называется полным, если существует такое полное множество логических связок (т. е. унарных и бинарных операторов) $D = \{\theta_1, \theta_2, \ldots, \theta_k\}$, что для каждого оператора $\theta_i \in D$ найдётся выражение $E \in C$ и отображение $s : U \to \{a, b\}$, для которых $s(E) \equiv a\theta_i b$ или $s(E) \equiv \theta_i a$ (в зависимости от унарности или бинарности θ_i).

А9.2. Разное

[ЛОГ 11] **БУЛЕВСКИЕ ФОРМУЛЫ С КВАНТОРАМИ БФК (*)**

УСЛОВИЕ. Заданы множество переменных $U = \{u_1, u_2, \ldots, u_n\}$ и правильно построенная булевская формула с кванторами $F = (Q_1 u_1)(Q_2 u_2) \ldots (Q_n u_n)E$, где E – булевское выражение, а Q есть либо $\forall$, либо $\exists$.

ВОПРОС. Верно ли, что F истинна?

Источник [503]. Общая сводимость.

Комментарий. Задача $P - SPACE$-полна даже в том случае, когда E – дизъюнктивная нормальная форма и каждая дизъюнкция содержит не более трёх литералов (3-ВЫП С КВАНТОРАМИ). Но если в каждой дизъюнкции имеется не более двух литералов, то задача разрешима за полиномиальное время [474]. Если ограничиться только формулами, содержащими не более k перемен кванторов (т. е. такими формулами F, что имеется не более k таких индексов i, что $Q_i \neq Q_{i+1}$), то задача полна в некотором зависящем от k и Q_1 классе полиномиальной иерархии (см. разд. 7.2).

[ЛОГ 12] **ТЕОРИЯ ПЕРВОГО ПОРЯДКА С РАВЕНСТВОМ (*)**

УСЛОВИЕ. Заданы конечное множество переменных $U = \{u_1, u_2, \ldots, u_n\}$, предложение S над U в теории первого порядка с равенством. (Предложения этой теории определяютсяиндуктивно следующим образом: «выражением» называется формула вида «$u = v$», где $u, v \in U$, либо формула вида «$\neg E$», «$(E \vee F)$», «$(E \wedge F)$» или $(E \Rightarrow F)$», где E и F – выражения. Предложением называется формула вида $(Q_1 u_1)(Q_2 u_2) \ldots (Q_n u_n)E$, где E – выражение, а Q_I – один из кванторов $\forall$ или $\exists$.)

ВОПРОС. Верно ли, что предложение S истинно во всех моделях теории?

Источник [503]. Общая сводимость.

Комментарий. Задача $P - SPACE$-полна. Аналогичная задача $P - SPACE$-трудна для любой теории первого порядка, имеющей модель, в которой какой-нибудь предикатный символ интерпретирован в виде не тождественно истинного отношения [233].

[ЛОГ 13] **$S5$-ВЫПОЛНИМОСТЬ МОДАЛЬНОЙ ЛОГИКИ**

УСЛОВИЕ. Задана правильно построенная модальная формула A над конечным множеством переменных U. Модальная формула – это либо переменная $u \in U$, либо формула вида «$(A \wedge B)$», «$\neg A$» или «$\Box A$», где A и B – модальные формулы.

ВОПРОС. Верно ли, что формула A является «$S5$-выполнимой»? Иначе говоря, существует ли модель (W, R, V), где W – множество, R – рефлексивное, транзитивное и симметричное бинарное отношение на W, а V – такое отображение множества $U \times W$ в множество $\{T, F\}$, что для некоторого $w \in W$ выполняется равенство $V(A, w) = T$? Отображение V продолжается на формулы согласно следующим правилам: $V(A \wedge B, w) = T$ тогда и только тогда, когда $V(A, w) = V(B, w) = T$; $V(\neg A, w) = T$ тогда и только тогда, когда $V(A, w) = F$; $V(\Box A, w) = T$ тогда и только тогда, когда $V(A, w') = T$ для всех $w' \in W$, таких, что $(w, w') \in R$.

Источник [314]. К этой задаче сводится 3-ВЫП. Доказательство принадлежности задачи к классу *NP*-нетривиально.

[ЛОГ 14] **ДОКАЗУЕМОСТЬ В МОДАЛЬНОЙ ЛОГИКЕ (*)**

УСЛОВИЕ. Заданы правильно построенная модальная формула A, модальная система $S \in \{K, T, S4\}$. (Детали см. в [314] или [229].)

ВОПРОС. Доказуема ли формула A в системе S?

Источник [314]. К этой задаче сводится БФК.

Комментарий. Задача $P - SPACE$-полна для фиксированной системы $S \in \{K, T, S4\}$ или для любой другой фиксированной модальной системы, в которой можно доказать всё доказуемое в K и нельзя доказать ничего недоказуемого в $S4$.

[ЛОГ 15] **ЛОГИКА ПРЕДИКАТОВ БЕЗ ОТРИЦАНИЯ**

УСЛОВИЕ. Заданы множество переменных $U = \{u_1, u_2, \ldots, u_n\}$, множество символов функций $F = \{f_1^{m_1}, f_2^{m_2}, \ldots, f_k^{m_k}\}$, множество символов отношений $R = \{R_1^{r_1}, R_2^{r_2}, \ldots, R_J^{r_j}\}$($m_i \geqslant 0$ и $r_i \geqslant 0$ – размерности соответствующих функций и отношений), правильно построенное логическое предложение A без отрицаний над U, F и R. (Такие предложения определяются индуктивно: терм есть переменная $u \in U$ или выражение вида $f_i^{m_i}(t_1, t_2, \ldots, t_m)$, где t_j – терм. Формула есть выражение вида $t_1 = t_2$, где t_1 и t_2 – термы, или выражение вида $R_i^{r_i}(t_1, t_2, \ldots, t_{r_i})$, где t_i – терм, или выражение вида $(A \wedge B), (A \vee B), \forall u_i(A), \exists u_i(A)$, где A и B – формулы, $u_i \in U$. Предложением называется формула, на каждую переменную которой навешен квантор, предшествующий её вхождению в формулу.)

ВОПРОС. Верно ли, что предложение A истинно при всех интерпретациях F и R?

Источник [302]. К этой задаче сводится 3-ВЫП. Доказательство того, что задача принадлежит классу *NP*, нетривиально.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда отсутствуют кванторы общности, символы отношений и имеются только две функции размерности 0 (т. е. константы).

[ЛОГ 16] **ВЫПОЛНИМОСТЬ КОНЪЮНКЦИЙ ВЫРАЖЕНИЙ, ВКЛЮЧАЮЩИХ ФУНКЦИИ И НЕРАВЕНСТВА**

УСЛОВИЕ. Заданы множество переменных U, множество F одноместных символов функций и набор C выражений вида $U * V$, где $*$ –один из символов «$\leqslant$», «$>$», «$=$» или «$\neq$», а U и V представляют собой один из символов «0», «1», «u», «$f(0)$», «$f(1)$» или «$f(u)$» для некоторых $f \in F$ и $u \in F$.

ВОПРОС. Можно ли так присвоить всем переменным из $u \in U$ и всем $f(u)$ (где $u \in U$ и $f \in F$) целочисленные значения, что все выражения из C будут истинны при стандартной интерпретации отношений $\leqslant, >, =$ и $\neq$?

Источник [435]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда отношения $=$ и $\neq$ не используются. Задача разрешима за полиномиальное время, если не используются отношения $\leqslant$ и $>$ [398] или если не используются отношения $=$ и $\neq$ и отсутствуют символы функций [356]. Вариант задачи, в котором W и V имеют вид «u» или «$u + c$» (где $u \in U$, $c \in Z$), является *NP*-полным, если разрешается использовать все четыре отношения, но разрешим за полиномиальное время, если можно пользоваться одной из пар отношений $\leqslant$ и $>$ или $=$ и $\neq$ [66].

[ЛОГ 17] **МИНИМАЛЬНОЕ МНОЖЕСТВО АКСИОМ**

УСЛОВИЕ. Заданы конечное множество «предложений» S, подмножество $T \subseteq S$ «истинных предложений», «отношение следования» R, состоящее из пар (A, s), где $A \subseteq S$ и $s \in S$, а также положительное целое число $K \leqslant |S|$.

ВОПРОС. Существуют ли подмножество $S_0 \subseteq T, |S_0| \leqslant K$ и положительное целое число n, такие, что $S_n = T$? (Здесь S_i, $1 \leqslant i \leqslant n$, – множество, состоящее в точности из таких предложений $s \in S$, что либо $s \in S_i$, либо существует $U \subseteq S_{i-1}$ и для которого $(U, s) \in R$.)

Источник [437]. К этой задаче сводится ТП-3.

Комментарий. Задача *NP*-полна даже тогда, когда $T = S$.

[ЛОГ 18] **ПОСЫЛКА ПЕРВОГО ПОРЯДКА**

УСЛОВИЕ. Заданы конечное множество «символов переменных» U, конечное множество «символов функции» C, набор $E = \{E_1, E_2, \ldots, E_m\}$ выражений над $U \cup C$, набор $F = \{F_1, F_2, \ldots, F_m\}$ выражений над C.

ВОПРОС. Существует ли подстановка S, отображающая каждую переменную $u \in U$ в такое выражение $s(u)$ над C, что $\{s(E_1), s(E_2), \ldots, s(E_m)\}$ есть подмножество множества $\{F_1, F_2, \ldots, F_n\}$? (Здесь $s(E_i)$ обозначает результат подстановки вместо каждого вхождения в E_i, переменной $u \in U$ соответствующего выражения $s(u)$.)

Источник [32, 33]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной для любого фиксированного $n \geqslant 3$, но при любом фиксированном m разрешима за полиномиальное время.

[ЛОГ 19] **ПОДТВЕРЖДЕНИЕ ВТОРОГО ПОРЯДКА**

УСЛОВИЕ. Заданы два «выражения логики второго порядка» E_1 и E_2, причём второе выражение не содержит переменных (в выражении второго порядка функции могут быть переменными, детали см. в [32]).

ВОПРОС. Существует ли такой набор значений переменных[9] из E_1, что в результате подстановки получается выражение, идентичное E_2?

Источник [32]. К этой задаче сводится 3-ВЫП. Доказательство принадлежности задачи классу *NP* нетривиально.

Комментарий. Разрешимость более общей задачи – УНИФИКАЦИЯ ВТОРОГО ПОРЯДКА, в которой и E_2, и E_1 могут содержать переменные и спрашивается, существует ли такой набор значений переменных, что в результате соответствующих подстановок E_1 и E_2 становятся идентичными, не известна. Задача УНИФИКАЦИЯ ТРЕТЬЕГО ПОРЯДКА неразрешима [228], а задача УНИФИКАЦИЯ ПЕРВОГО ПОРЯДКА разрешима за полиномиальное время [32, 421].

А10. ТЕОРИЯ АВТОМАТОВ И ЯЗЫКОВ

А10.1. Теория автоматов

[ТАЯ 1] **НЕЭКВИВАЛЕНТНОСТЬ КОНЕЧНЫХ АВТОМАТОВ (*)**

УСЛОВИЕ. Заданы два недетерминированных конечных автомата A_1 и A_2 с общим входным алфавитом Σ. (Недетерминированный конечный автомат $A = (Q, \Sigma, \delta, q_0, F)$ состоит из конечного множества состояний Q, входного алфавита Σ, функции перехода δ, отображающей $Q \times \Sigma$ в подмножества множества Q, начального состояния q_0 и множества допускающих состояний $F, F \subseteq K$ (см., например, [215]).)

ВОПРОС. Верно ли, что A_1 и A_2 распознают различные языки?

Источник [292]. К этой задаче сводится НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ.

Комментарий. Задача $P-SPACE$-полна даже в том случае, когда $|\Sigma| = 2$ и A_2 – тривиальный автомат, распознающий Σ^*. Общая задача *NP*-полна, если $|\Sigma| = 1$ или если оба автомата A_1 и A_2 распознают конечные языки (последнее свойство может быть проверено за полиномиальное время [215]). Задача разрешима за полиномиальное время, если A_1 и A_2 – детерминированные конечные автоматы (см., например, [215]).

[ТАЯ 2] **НЕТРИВИАЛЬНОСТЬ ДВУСТОРОННЕГО КОНЕЧНОГО АВТОМАТА (*)**

УСЛОВИЕ. Задан двусторонний недетерминированный конечный автомат $A = (Q, \Sigma, \delta, q_0, F)$, где Q, Σ, δ, q_o и F обозначают то же самое, что и для одностороннего недетерминированного конечного автомата, а функция перехода δ отображает $Q \times \Sigma$ в подмножества множества $Q \times \{-1, 0, 1\}$ (см., например, [215]).

ВОПРОС. Существует ли слово $x \in \Sigma^*$, допускаемое автоматом A?

Источник [231]. К этой задаче сводится ДОПУСКАЕМОСТЬ ЛИНЕЙНО ОГРАНИЧЕННЫМ АВТОМАТОМ.

---

[9] В данном случае значением переменной может быть любой терм. – *Прим. ред.*

Комментарий. Задача $P - SPACE$-полна даже тогда, когда $|\Sigma| = 2$ и A – детерминированный автомат. Если $|\Sigma| = 1$, то общая задача *NP*-полна [136]. Если $|\Sigma|$ – односторонний недетерминированный конечный автомат, то общая задача может быть решена за полиномиальное время (см., например, [215]). Аналогичные результаты о конечности языка, распознаваемого автоматом A, можно найти в указанных выше работах.

[ТАЯ 3] **ДОПУСКАЕМОСТЬ КОНЕЧНЫМ АВТОМАТОМ (*)**

УСЛОВИЕ. Заданы «линейно ограниченный автомат» A с входным алфавитом Σ (определение см. в книге [215]) и слово $x \in \Sigma^*$.

ВОПРОС. Верно ли, что автомат A допускает слово x?

Источник [280]. Общая сводимость.

Комментарий. Задача $P - SPACE$-полна даже в том случае, если A – детерминированный автомат (см. задачу ДОПУСКАЕМОСТЬ В ЛИНЕЙНОЕ ВРЕМЯ из разд. 7.4). Кроме того, существуют конкретные детерминированные линейно ограниченные автоматы, для которых задача $P - SPACE$-полна.

[ТАЯ 4] **ДОПУСКАЕМОСТЬ АВТОМАТОМ КВАЗИРЕАЛЬНОГО ВРЕМЕНИ**

УСЛОВИЕ. Заданы многоленточная машина Тьюринга M (в нашей терминологии – программа для машины Тьюринга), у которой читающая головка входной ленты на каждом такте работы сдвигается вправо, причём если читающая головка видит пустой символ, то машина обязана остановиться, а также слово x, принадлежащее входному алфавиту Σ машины M. (Более полное описание машин этого типа и их эквивалентное задание можно найти в работе [50].)

ВОПРОС. Верно ли, что M допускает x?

Источник [46]. Общая сводимость.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если кроме входной ленты M имеет только одну рабочую ленту. См. также задачу ПРИНАДЛЕЖНОСТЬ ЯЗЫКУ КВАЗИРЕАЛЬНОГО ВРЕМЕНИ (языки, допускаемые автоматами квазиреального времени, совпадают с квазиреальными по времени языками, определёнными в [ТАЯ 18]).

[ТАЯ 5] **ДОПУСКАЕМОСТЬ НЕСТИРАЮЩИМ СТЕКОВЫМ АВТОМАТОМ (*)**

УСЛОВИЕ. Заданы «односторонний нестирающий стековый автомат A (1НСС-автомат) с входным алфавитом Σ (определение см. в книге [215]) и слово $x \in \Sigma^*$.

ВОПРОС. Верно ли, что автомат A допускает слово x?

Источник [136, 214]. К этой задаче сводится ДОПУСКАЕМОСТЬ ЛИНЕЙНО ОГРАНИЧЕННЫМ АВТОМАТОМ. Во второй работе доказано, что задача принадлежит классу $P - SPACE$.

Комментарий. Задача $P - SPACE$-полна даже в том случае, если слово $x \in \Sigma^*$ фиксировано и A – «допускающий стековый автомат» (определение см. в работе [188]). Если x – пустое слово и на A наложить дополнительное ограничение, состоящее в том, что A –допускающий стековый автомат с одним стековым символом, то задача становится *NP*-полной .[136]. Если же x меняется, а A фиксирован, то задача принадлежит классу *NP* для любого 1НСС-автомата, то же самое верно, если A – произвольный «вложенный стековый автомат» [460]. Существуют конкретные 1НСС-автоматы, для которых упомянутая задачэ *NP*-полна [460], эти 1НСС-автоматы можно взять из класса допускающих стековых автоматов [482], которые одновременно являются «читающими магазинными автоматами» [232]. Однако если A – «односторонний недетерминированный магазинный автомат», то задача разрешима за полиномиальное время (даже если A может меняться); последнее утверждение верно и для «двусторонних недетерминированных магазинных автоматов» [8].

[ТАЯ 6] **ПЕРЕСЕЧЕНИЕ ДЛЯ КОНЕЧНЫХ АВТОМАТОВ (*)**

УСЛОВИЕ. Задана последовательность $A_1, A_2, \ldots, A_n$ детерминированных конечных автоматов, имеющих общий входной алфавит Σ.

ВОПРОС. Существует ли слово $x \in \Sigma^*$, допускаемое всеми автоматами $A_i, 1 \leqslant i \leqslant n$?

Источник [300, 301]. К этой задаче сводится ДОПУСКАЕМОСТЬ С ЛИНЕЙНОЙ ПАМЯТЬЮ.

Комментарий. Задача $P-SPACE$-полна. Для любого фиксированного n она разрешима за полиномиальное время [215].

[ТАЯ 7] **РЕДУКЦИЯ НЕ ПОЛНОСТЬЮ ОПРЕДЕЛЕННОГО АВТОМАТА**

УСЛОВИЕ. Заданы не полностью определённый детерминированный конечный автомат $A = (Q, \Sigma, \delta, q_0, F)$ (где Q – множество состояний, Σ – входной алфавит, δ – «частичная» функция перехода, отображающая некоторое подмножество множества $Q \times \Sigma$ в Q, начальное состояние $q_0 \in Q$, $F \subseteq Q$ – множество «допускающих состояний») и положительное целое число K.

ВОПРОС. Можно ли функцию перехода δ так продолжить до всюду определённой функции, отображающей $Q \times \Sigma$ в Q, что у получающегося в результате полностью определённого автомата имеется эквивалентный «редуцированный автомат», число состояний которого не превосходит K?

Источник [426]. К этой задаче сводится РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА.

Комментарий. Задача остаётся *NP*-полной при любом фиксированном $K \geqslant 6$. Близкая задача, в которой допускается «расщепление состояний» (как, например, в работе [423]), также *NP*-полна для любого фиксированного $K \geqslant 6$ (см. [426]). Если наряду с «расщеплением состояний» допускается также «расщепление символов», как, например, в работе [187], то аналогичная задача, когда у приведённого автомата сумма числа состояний и числа символов не должна превосходить K, также *NP*-полна [427]. Задача отыскания детерминированного конечного автомата с минимальным числом состояний, эквивалентного заданному полностью определённому автомату, разрешима за полиномиальное время (см., например, [211] или [14]). Аналогичная задача для полностью определённого недетерминированного конечного автомата $P - SPACE$-полна (см. задачу НЕЭКВИВАЛЕНТНОСТЬ КОНЕЧНЫХ АВТОМАТОВ).

[ТАЯ 8] **МИНИМАЛЬНЫЙ РАЗДЕЛЯЮЩИЙ КОНЕЧНЫЙ АВТОМАТ**

УСЛОВИЕ. Заданы конечный алфавит Σ, два конечных подмножества $S, T \in \Sigma^*$ и положительное целое число K.

ВОПРОС. Существует ли такой детерминированный конечный автомат A, имеющий K состояний, что распознаваемый им язык $L \subseteq \Sigma^*$ обладает свойством, согласно которому $S \subseteq L, T \subseteq \Sigma^* \setminus L$?

Источник [174]. К этой задаче сводится МОНОТОННАЯ 3-ВЫП.

Комментарий. Если для некоторого n справедливо $S \cup T = \Sigma^{(n)}$ (где $\Sigma^{(n)}$ – множество всех слов над Σ, длины, не превосходящей n), то задача разрешима за полиномиальное время [514]. Однако для любого фиксированного $\varepsilon > 0$ задача *NP*-полна даже в том случае, если ограничиться индивидуальными задачами, в которых $(S \cup T) \subseteq \Sigma^{(n)}$ и $|\Sigma^{(n)} - (S \cup T)| \leqslant |\Sigma^{(n)}|^{\varepsilon}$ [17].

A10.2. Формальные языки

[ТАЯ 9] **НЕЭКВИВАЛЕНТНОСТЬ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ (*)**

УСЛОВИЕ. Заданы два регулярных выражения E_1 и E_2 с операторами $\{\cup, \cdot, *\}$ и алфавит Σ (определение см. в разд. 7.4).

ВОПРОС. Верно ли, что E_1 и E_2 представляют различные языки?

Источник [503,497]. Общая сводимость. Во второй работе доказано, что задача лежит в классе $P - SPACE$.

Комментарий. Задача $P - SPACE$-полна, даже если $\Sigma = 2$ и $E_2 = \Sigma^*$ (НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ, см. разд. 7.4). В действительности задача $P - SPACE$-полна, если E_2 – любое фиксированное выражение, представляющее «неограниченный» язык [237]. Для фиксированного E_2, представляющего некоторый бесконечный «ограниченный» язык, задача *NP*-полна, но разрешима за полиномиальное время для любого фиксированного выражения E_2, представляющего любой конечный язык. Общая задача остаётся $P - SPACE$-полной, если E_1 и E_2 имеют фиксированную «высоту относительно *», равную k, $k \geqslant 1$ [503,230]. Задача остаётся *NP*-полной, если E_1 или E_1 и E_2 представляют ограниченные языки (последнее свойство проверяемо за полиномиальное время) [237] или если $|\Sigma| = 1$ [503]. В цитированных работах указаны другие близкие результаты и труднорешаемые обобщения (см. также [231] и [236]).

[ТАЯ Ю] **МИНИМАЛЬНОЕ РАЗДЕЛЯЮЩЕЕ РЕГУЛЯРНОЕ ВЫРАЖЕНИЕ**

УСЛОВИЕ. Заданы конечный алфавит Σ, два конечных подмножества $S, T \subseteq \Sigma^*$ и положительное целое число K.

ВОПРОС. Существует ли такое регулярное выражение E над Σ, содержащее не более K символов из Σ, что язык L, определяемый выражением E, обладает свойством: $S \subseteq L$ и $T \subseteq \Sigma \setminus L$?

Источник [17]. К данной задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, даже если ограничиться только выражениями E, не содержащими оператора «$\cup$» или оператора «$*$» [16].

[ТАЯ И] ПОКРЫТИЕ РЕИНОЛЬДСА ДЛЯ БЕСКОНТЕКСТНЫХ ГРАММАТИК

УСЛОВИЕ. Заданы две бесконтекстные грамматики $G_1 = \{N_1, \Sigma, \Pi_1, S_1)$ и $G_2 = (N_2, \Sigma, \Pi_2, S_2)$, где Σ – конечное множество терминальных символов, N_i – конечное множество нетерминальных символов, $S_i \in N_i$ – начальный символ и Π_i – множество «продукций» вида «$A \to w$», где $A \in N_i$ и $w \in (N_i \cup \Sigma)^*$.

ВОПРОС. Верно ли, что G_2 – «покрытие Рейнольдса» для грамматики G_1? Иными словами, существует ли такая функция f, отображающая $N_1 \cup \Sigma$ в $N_2 \cup \Sigma$, что $f(x) = x$ для всех $x \in \Sigma$, $f(A) \in N_2$ для всех $A \in N_1$, $f(S_1) = S_2$ и для каждой продукции вида $A \to x_1 x_2 \ldots x_n$ из Π_1 её образ $f(A) \to f(x_1)f(x_2) \ldots f(x_n)$ есть продукция из Π_2?

Источник [235]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной, если G_1 и G_2 – «регулярные»[10] грамматики. Аналогичные результаты верны для близких вопросов: «Верно ли, что G_2 – «слабое покрытие Рейнольдса для G_1?» и «Верно ли, что G_2 – «гомоморфный образ» грамматики G_1?» Задача: «Задана грамматика G, существует ли такая $LL(k)$ -бесконтекстная грамматика H, что H – покрытие Рейнольдса для грамматики G?» – разрешима за полиномиальное время. То же самое верно для задач, получающихся заменой класса $LL(k)$ на $LR(k)$ или на один из нескольких других классов грамматик (см. [235]).

[ТАЯ 12] ПОКРЫТИЕ ЛИНЕЙНЫХ ГРАММАТИК (*)

УСЛОВИЕ. Заданы две линейные бесконтекстные грамматики $G_1 = (N_1, \Sigma, \Pi_1, S_1)$ и $G_2 = (N_2, \Sigma, \Pi_2, S_2)$ (в таких грамматиках не допускаются продукции, содержащие в правой части более одного нетерминального символа).

ВОПРОС. Существует ли такая функция $h : P_1 \to P_2 \cup \lambda$ (где λ обозначает пустую продукцию), что при отображении h грамматика G_1 покрывает G_2? (Говорят, что при отображении h грамматика G_1 покрывает грамматику G_2, если для всех слов $w \in \Sigma^*$ выполнены два свойства: (1) если w выводимо из S_1 посредством некоторой последовательности продукций $p_1, p_2, \ldots, p_n$, то w выводимо из S_2 последовательностью продукций $h(p_1), h(p_2), \ldots, h(p_n)$; (2) если w выводимо из S_2 посредством продукций $q_1, q_2, \ldots, q_n$ из Π_2, то существует такая последовательность продукций $p_1, p_2, \ldots, p_m$, выводящая w в грамматике G_1, что $h(p_1), h(p_2), \ldots, h(p_m)$ совпадает с $q_1, q_2, \ldots, q_n$.)

---

[10] Регулярной называется бесконтекстная грамматика, каждая продукция которой имеет вид $A \to aB$ или $A \to a$, где $A, B \in N$ и $a \in \Sigma$. – *Прим. ред.*

Источник [237,238]. К этой задаче сводится НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ. Во второй работе доказана принадлежность задачи классу $P - SPACE$.

Комментарий. Задача $P-SPACE$-полна даже для «регулярных» грамматик. Для произвольных бесконтекстных грамматик задача неразрешима. В работе [235] приведены близкие результаты.

[ТАЯ 13] **СТРУКТУРНАЯ НЕЭКВИВАЛЕНТНОСТЬ ЛИНЕЙНЫХ ГРАММАТИК (*)**

УСЛОВИЕ. Заданы две бесконтекстные грамматики $G_1 = (N_1, \Sigma, \Pi_1, S_1)$ и $G_2 = (N_2, \Sigma, \Pi_2, S_2)$.

ВОПРОС. Верно ли, что G_1 и G_2 – «структурно неэквивалентны»? (Грамматики G_1 и G_2 называются структурно неэквивалентными, если грамматики со скобками, получаемые из G_1 и G_2 заменой каждой продукции вида $A \to w$ продукцией вида $A \to (w)$ (где «(» и «)» – новые терминальные символы), порождают различные языки.)

Источник [237]. К этой задаче сводится НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ.

Комментарий. Задача $P - SPACE$-полна, если G_1 и G_2 регулярны и $|\Sigma| = 2$. Она *NP*-полна, если $|\Sigma| = 1$. Для произвольных бесконтекстных языков задача разрешима, но принадлежность её классу $P - SPACE$ не установлена.

[ТАЯ 14] **НЕЭКВИВАЛЕНТНОСТЬ РЕГУЛЯРНЫХ ГРАММАТИК (*)**

УСЛОВИЕ. Заданы две регулярные грамматики $G_1 = (N_1, \Sigma, \Pi_1, S_1)$ и $G_2 = (N, \Sigma, \Pi_2, S_2)$. Регулярной грамматикой называется бесконтекстная грамматика, в которой каждая продукция имеет вид $A \to aB$ или $A \to a$, где $A, B \in N, a \in \Sigma$.

ВОПРОС. Верно ли, что G_1 и G_2 порождают различные языки?

Источник [70]. К этой задаче сводится НЕЭКВИВАЛЕНТНОСТЬ КОНЕЧНЫХ АВТОМАТОВ.

Комментарий. Задача $P-SPACE$-полна даже в том случае, если $|\Sigma| = 2$ и G_2 – фиксированная грамматика, порождающая Σ^* (задача НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОЙ ГРАММАТИКИ). Общая задача *NP*-полна, когда $|\Sigma| = 1$ или когда обе грамматики порождают конечные языки (последнее свойство проверяется за полиномиальное время; см., например, [215]). Если G_1 – произвольная линейная грамматика, а G_2 – фиксированная грамматика, порождающая Σ^* (задача НЕУНИВЕРСАЛЬНОСТЬ ЛИНЕЙНОЙ ГРАММАТИКИ), то задача становится неразрешимой [237].

[ТАЯ 15] **НЕ $LR(K)$-БЕСКОНТЕКСТНАЯ ГРАММАТИКА**

УСЛОВИЕ. Заданы бесконтекстная грамматика G и положительное целое число K, представленное в унарной записи.

ВОПРОС. Верно ли, что G не является $LR(K)$-грамматикой (определение см. в [242])?

Источник [242]. Общая сводимость.

Комментарий. При любом фиксированном K задача разрешима за полиномиальное время. Если K представлено в двоичной записи (что и требуется при нашем стандартном кодировании), то задача полна в классе $NEXP-TIME$ и, следовательно, труднорешаема. Вопрос о существовании такого K, что G есть $LR(K)$-грамматика, неразрешим [239]. Аналогичные результаты имеют место и тогда, когда свойство «$LR(K)$» заменяется свойством $<< LL(K), << LC(K) >>, << SLR(K) >>$ или любым другим из большего числа свойств (см. указанные выше работы). Однако в случае свойства «$LL(K)$», если известно, что для некоторого K' грамматика G обладает свойством $LR(K')$, то за полиномиальное время можно выяснить вопрос о существовании такого K, что грамматика G обладает свойством «$LL(K)$» [241].

[ТАЯ 16] **НЕПУСТОТА ETOL-ГРАММАТИКИ (*)**

УСЛОВИЕ. Задана ETOL-грамматика $G = (N, \Sigma, \{\delta_1, \delta_2, \dots, \delta_n\}, S)$, где N – конечное множество «нетерминальных» символов, Σ – конечное множество «терминальных» символов, $S \in N$ – начальный символ, каждое δ_i представляет собой «таблицу» продукций, переводящих символы множества $N \cup \Sigma$ в слова языка $(N \cup \Sigma)^*$ (на каждом шаге вывода каждый символ имеющегося в наличии слова заменяется посредством некоторой продукции из заранее выбранной таблицы (см., например, [207])).

ВОПРОС. Верно ли, что язык, порождённый грамматикой G, не пуст?

Источник [275]. К этой задаче сводится НЕУНИВЕРСАЛЬНОСТЬ РЕГУЛЯРНОГО ВЫРАЖЕНИЯ.

Комментарий. Задача $P-SPACE$-полна, даже если G является «EDTOL-грамматикой без ε». Задача *NP*-полна и в том случае, если G содержит ровно одну таблицу и для каждого символа ровно одну продукцию (т. е. если G есть «EDOL-грамматика»). Для вопроса о том, порождает ли G бесконечный язык, верны аналогичные результаты. Для бесконтекстных грамматик эти задачи разрешимы за полиномиальное время, а для контекстных грамматик – неразрешимы (см., например, [215]).

[ТАЯ 17] **ПРИНАДЛЕЖНОСТЬ БЕСКОНТЕКСТНОМУ ЯЗЫКУ С УПРАВЛЯЕМЫМ ВЫВОДОМ**

УСЛОВИЕ. Заданы бесконтекстная грамматика с управляемым выводом $G = (N, \Sigma, \Pi, S)$ без ε и слово $x \in \Sigma^*$. (В таких грамматиках все продукции из Π имеют вид $A \to w(T)(F)$, где $A \in N, w \in (N \cup \Sigma^*) \setminus \varepsilon, T$ и F – подмножества множества Π, указывающие, в зависимости от применимости или неприменимости предыдущей продукции, подмножество в Π, из которого должна быть выбрана следующая продукция [456].)

ВОПРОС. Верно ли, что x принадлежит языку, порождённому грамматикой G?

Источник [482]. К этой задаче сводится 3-ВЫП. Отсутствие ε обеспечивает принадлежность классу *NP*.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если потребовать, чтобы все продукции $A \to w(T)(F)$ из Π имели $T = F$ [525].

Если для всех продукций из Π $T = F = \Pi$ и если допускается применение продукций к пустому слову ε, то получается задача принадлежности контекстно-свободным языкам, которая разрешима за полиномиальное время (см., например, [215]).

[ТАЯ 18] **ПРИНАДЛЕЖНОСТЬ ЯЗЫКУ КВАЗИРЕАЛЬНОГО ВРЕМЕНИ**

УСЛОВИЕ. Заданы бесконтекстные грамматики G_1, G_2 и G_3, имеющие общий терминальный алфавит Σ, второй конечный алфавит Γ, функция $h : \Sigma \to \Gamma$ и слово $w \in \Gamma^*$.

ВОПРОС. Верно ли, что w принадлежит «языку квазиреального времени», определяемому грамматиками G_1, G_2, G_3 и функцией h, т. е. языку $L = h(L_1 \cap L_2 \cap l_3)$, состоящему из всех слов из Γ^* вида $h(x_1)h(x_2) \dots h(x_k)$, таких, что $x_1, x_2, \dots, x_k \in L_1 \cap L_2 \cap L_3$, где L_1, L_2 и L_3 – языки, порождаемые грамматиками G_1, G_2 и G_3 соответственно?

Источник [230, 188]. К этой задаче сводится 3-ВЫП.

Комментарий. Если h – взаимно-однозначная функция, то задача решается за полиномиальное время (стандартным методом разбора бесконтекстных грамматик; см., например, [215]). Задача остаётся *NP*-полной и тогда, когда L_3 совпадает с Σ^*, т.е. $L = h(L_1 \cap L_2)$ [188], но разрешима за полиномиальное время, если L_2 и L_3 совпадают с Σ^*. т. е. $L = h(L_1)$.

[ТАЯ 19] **ПРИНАДЛЕЖНОСТЬ ETOL-ЯЗЫКУ (*)**

УСЛОВИЕ. Заданы ETOL-грамматика $G = (N, \Sigma, \{\delta_1, \delta_2, \dots, \delta_n\}, S)$ и слово $w \in \Sigma^*$.

ВОПРОС. Принадлежит ли слово w языку, порождаемому грамматикой G?

Источник [525]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача $P - SPACE$-полна, даже если G есть «ETOL-грамматика без ε» [275]. Если G фиксирована, то задача принадлежит классу *NP*, в этом случае существуют конкретные грамматики, для которых она *NP*-полна. Последний факт имеет место даже тогда, когда G есть «TOL-грамматика» (т. е. не имеет нетерминальных символов) и не содержит е [525]. Задача разрешима за полиномиальное время для фиксированной G, если G есть «EDTOL-грамматика» [276] либо если G есть «EOL-грамматика» (т. е. имеет только одну таблицу) [404].

[ТАЯ 20] **ПРИНАДЛЕЖНОСТЬ КОНТЕКСТНОМУ ЯЗЫКУ (*)**

УСЛОВИЯ- Заданы контекстная грамматика $G = (N, \Sigma, \Pi, S)$ и слово $w \in \Sigma^*$ (в контекстной грамматике каждая продукция имеет вид $x \to y$, где x и y – непустые слова над алфавитом $N \cup \Sigma$ и $|y| \geqslant |x|$).

ВОПРОС. Принадлежит ли w языку, порождённому грамматикой G?

Источник [310]. К этой задаче сводится ДОПУСКАЕМОСТЬ ЛИНЕЙНО ОГРАНИЧЕННЫМ АВТОМАТОМ.

Комментарий. Задача $P - SPACE$-полна даже для детерминированных контекстных грамматик. Более того, существуют конкретные контекстные грамматики, для которых задача *NP*-полна. Существует также конкретная «линейная по времени» контекстная грамматика, для которой задача *NP*-полна [49]. (Для любой конкретной контекстной грамматики задача принадлежит классу *NP*.)

[ТАЯ 21] **ПРИНАДЛЕЖНОСТЬ ЯЗЫКУ, ОПРЕДЕЛЯЕМОМУ ДРЕВОВИДНЫМ ПРЕОБРАЗОВАТЕЛЕМ (*)**

УСЛОВИЕ. Заданы «преобразователь сверху-вниз с древовидной структурой» M и его выходной алфавит Γ, бесконтекстная грамматика G и слово $w \in \Gamma^*$ (детали см. в указанных ниже работах).

ВОПРОС. Принадлежит ли w «результату» «поверхностного множества», определяемого преобразователем M и грамматикой G?

Источник [446]. Общая сводимость.

Комментарий. Задача $P-SPACE$-полна. Для фиксированных M и G она принадлежит классу *NP*. Существуют конкретные M и G, для которых задача *NP*-полна [460]. Если потребовать линейности преобразователя M, то общая задача разрешима за полиномиальное время. Если M – «детерминированный» преобразователь, то для фиксированного M задача разрешима за полиномиальное время [447].

А11. ОПТИМИЗАЦИЯ ПРОГРАММ

А11.1. Генерация кода программы

[ОП 1] **ДОСТАТОЧНОСТЬ ЧИСЛА РЕГИСТРОВ**

УСЛОВИЕ. Заданы ориентированный ациклический граф $G = (V, A)$, положительное целое число K.

ВОПРОС. Существует ли вычисление для G, использующее не более K регистров? Иными словами, существуют ли упорядочение $v_1, v_2, \ldots, v_n$ вершин множества V, где $n = |V|$, и последовательность $S_0, S_1, \ldots, S_n$ подмножеств множества V, такие, что: (1) $|S_i| \leqslant K($ для всех i; (2) S_0 – пусто, S_n содержит все вершины из графа G со степенью захода 0; (3) для всех $i, 1 \leqslant i \leqslant n, v_i \in S_i, S_i \setminus \{v_i\} \subseteq S_{i-1}$ и S_{i-1} содержит все вершины u, такие, что $(v_i, u) \in A$?

Источник [477]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если степени выхода всех вершин графа G не превосходят 2. Вариант этой задачи, в котором допускается «перевычисление» (т. е. ищутся последовательности $v_1, v_2, \ldots, v_m$ и $S_0, S_1, \ldots, S_m$ без априорной оценки числа m и вершины в первой последовательности могут повторяться, но все остальные сформулированные выше условия выполняются), является *NP*-трудным, и его принадлежность классу *NP* не установлена.

[ОП 2] **ДОПУСТИМОСТЬ НАЗНАЧЕНИЯ РЕГИСТРОВ**

УСЛОВИЕ. Заданы ориентированный ациклический граф $G = (V, A)$, положительное целое число K и назначение регистров $f : V \to \{R_1, R_2, \ldots, R_k\}$.

ВОПРОС. Существует ли вычисление для G, использующее указанное назначение регистров? Иными словами, существуют ли упорядочение $v_1, v_2, \ldots, v_n$ вершин из множества V и последовательность $S_1, S_2, \ldots, S_n$ подмножеств множества V, удовлетворяющие всем условиям, сформулированным в задаче ДОСТАТОЧНОСТЬ ЧИСЛА РЕГИСТРОВ, и дополнительному условию, заключающемуся в том, что для всех i и $j (1 \leqslant j \leqslant K, 1 \leqslant i \leqslant n)$ существует не более одной вершины $u \in S_i$, такой, что $f(u) = R_j$?

Источник [477]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если степени выхода всех вершин графа G не превосходят 2.

[ОП 3] ДОСТАТОЧНОСТЬ ЧИСЛА РЕГИСТРОВ ДЛЯ РЕАЛИЗАЦИИ ЦИКЛОВ

УСЛОВИЕ. Заданы множество V параметров циклов, длина цикла $N \in Z^+$, для каждого параметра v начальное время $s(v) \in Z_0^+$ и продолжительность $l(v) \in Z^+$, целое положительное число K.

ВОПРОС. Достаточно ли K регистров для запоминания параметров циклов? Иными словами, существует ли такое назначение регистров $f : V \to \{1, 2, \ldots, K\}$, что если $f(u) = f(v)$ для некоторых $u, v \in V, u \neq v$, то из неравенства $s(u) \leqslant s(v)$ следует, что $s(u)+l(u) \leqslant^s (v)$ и $s(v)+l(v)(modN) \leqslant s(u)$?

Источник [153]. К этой задаче сводятся ОБРАЗУЮЩИЕ ПЕРЕСТАНОВКИ.

Комментарий. Для любого фиксированного K задача разрешима за полиномиальное время.

[ОП 4] ГЕНЕРАЦИЯ КОДА НА МАШИНЕ С ОДНИМ РЕГИСТРОМ

УСЛОВИЕ. Заданы ациклический ориентированный граф $G = (V, A)$, у которого степени выхода всех вершин не превосходят Σ, и положительное целое число K.

ВОПРОС. Существует ли программа в кодах ЭВМ, содержащая не более K команд, вычисляющая на машине с одним регистром все корневые (т. е. со степенью захода равной 0) вершины графа G, причём в начальный моменте в памяти находятся все «листья» графа G (т. е. вершины со степенью выхода равной 0) и в программе используются только команды «загрузить», «запомнить» и «выполнить»? (Команда «загрузить» передает в регистр определённую вершину. Команда «запомнить» передает вершину из регистра в память. Новая вершина v может быть вычислена с помощью команды «выполнить» при условии, что в регистре находится такая вершина u, что $(v, u) \in A$, а любая другая вершина u', для которой $(v, u') \in A$, находится в памяти. В результате исполнения команды «выполнить» вершина u, находящаяся в регистре, заменяется вершиной v. Вычисление новой вершины не закончено, пока она не помещена в память с помощью операции «запомнить».)

Источник [61]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если все вершины, имеющие степень захода больше 1, соединены рёбрами только с листьями графа G [10]. Задача разрешима за полиномиальное время, если G – ориентированный лес [480].

[ОП 5] **ГЕНЕРАЦИЯ КОДА С НЕОГРАНИЧЕННЫМ ЧИСЛОМ РЕГИСТРОВ**

УСЛОВИЕ. Заданы ориентированный ациклический граф $G = (V, A)$, не имеющий вершин со степенью выхода большей Σ, разбиение множества A на такие два непересекающихся множества L и R, что любые две дуги, исходящие из одной вершины, принадлежат различным множествам, положительное целое число K.

ВОПРОС. Существует ли программа вычисления всех корневых вершин графа G, отвечающая следующим требованиям: (1) программа состоит не более чем из K команд, (2) программа начинает вычисление с листьев графа G, занесённых в память, (3) в программе используются только команды вида «$r_i \to r_j$» или «$r_i \leftarrow r_i op r_j$», $i, j \in Z^+$ (где вершина v со степенью выхода, равной 2, и исходящими дугами $(v, u) \in L$ и $(v, w) \in R$ может быть вычислена только с помощью команды $r_i \leftarrow r_i op r_j$, в которой r_i содержит u, а r_j содержит v)?

Источник [10]. К этой задаче сводится МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если только листья графа G имеют степень захода, превосходящую 1. «Коммутативный» вариант задачи, в котором допускаются также команды вида «$r_i \leftarrow r_j op r_i$», является *NP*-полным [11]. Если G – лес или если допускаются трёхадресные команды вида «$r_i \leftarrow r_j op r_k$, то обе задачи могут быть решены за полиномиальное время [10].

[ОП 6] **ГЕНЕРАЦИЯ КОДА ДЛЯ ПАРАЛЛЕЛЬНЫХ ПРИСВАИВАНИЙ**

УСЛОВИЕ.Заданы множество переменных $V = \{v_1, v_2, \dots, v_n\}$, множество операторов присваивания $A = \{A_1, A_2, \dots, A_n\}$, где каждое A_i имеет вид «$v_i \leftarrow op(B_i)$» (B_i – некоторое подмножество множества V), и положительное целое число K.

ВОПРОС. Существует ли такое упорядочение $v_{\pi(1)}, v_{\pi(2)}, \dots, v_{\pi(n)}$ множества V, для которого имеется не более K значений $i, 1 \leqslant i \leqslant n$, таких, что для некоторого $j > i$ имеет место принадлежность $v_{\pi(i)} \in B_{\pi(j)}$?

Источник [476]. К этой задаче сводится МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ.

Комментарий. Задача остаётся *NP*-полной и в том случае, если $|B_i| \leqslant 2$ для всех B_i.

[ОП 7] **ГЕНЕРАЦИЯ КОДА С АДРЕСНЫМИ ВЫРАЖЕНИЯМИ**

УСЛОВИЕ. Заданы последовательность команд $I = (I_1, I_2, \dots, I_n)$, для каждой команды $I_i \in I$ выражение $g(I_i)$ вида «I_j», «$I_j + k$», «$I_j - k$» или «k», где $I_j \in I$ и $k \in Z^+$, а также положительные целые числа B, C и M.

ВОПРОС. Можно ли команды из последовательности I запомнить в виде одно- и двухбайтовых команд, так чтобы общая память, используемая при этом, не превзошла M? Иными словами: определим функцию $h(I_j)$, полагая её равной $f(I_j), f(I_j) \pm k$ или k, в зависимости от того, равняется ли $g(I_j)$ соответственно $I_j, I_j \pm k$ или k; существует ли такое взаимнооднозначное отображение $f : I \to \{1, 2, \ldots, M\}$, что $f(I_i) < f(I_j)$ при $i < j$ и либо $-C < f(I_i) - h(I_i) < B$, либо $f(I_i) + 1$ не принадлежит области значений f?

Источник [511]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной для некоторых конкретных значений B и C, например равных соответственно 128 и 127 (можно взять также и гораздо меньшие значения). Задача разрешима за полиномиальное время, если в условии нет «паталогических» выражений (детали см. в указанной выше работе).

[ОП 8] **ГЕНЕРАЦИЯ КОДА ПРОГРАММЫ С НЕФИКСИРОВАННЫМИ АДРЕСАМИ ПЕРЕМЕННЫХ**

УСЛОВИЕ. Заданы последовательность команд $I = (I_1, I_2, \ldots, I_n)$, конечное множество переменных V, назначение $g : I \to I \cup V$ и положительные целые числа B и M.

ВОПРОС. Можно ли команды из I, рассматриваемые как одно- и двухбайтовые слова, с переменными между ними так расположить в памяти, что используемый при этом объём памяти не превзойдет M? Иными словами, существует ли такое взаимнооднозначное отображение $f : I \cup V \to \{1, 2, \ldots, M\}$, что (1) при $i < j f(I_i) < f(I_j)$ и (2) для всех $i, 1 \leqslant i \leqslant n$, либо $|f(I_i) - f(g(I_i))| < B$, либо $f(I_i) + 1$ не принадлежит области значений функции f?

Источник [450]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже для некоторых фиксированных B, например для $B = 31$. Если V пусто, то задача разрешима за полиномиальное время.

[ОП 9] **ВЫЧИСЛЕНИЕ СЕМЕЙСТВА**

УСЛОВИЕ. Заданы набор C подмножеств конечного множества A и положительное целое число J.

ВОПРОС. Существует ли последовательность

$$S = \{z_1 \leftarrow x_1 \cup y_1, z_2 \leftarrow x_2 \cup y_2, \ldots, z_i \leftarrow x_j \cup y_j\}, j \leqslant J,$$

операторов «объединения», где каждое из x_i и y_i есть либо $\{a\}$ для некоторого $a \in A$, либо z_k для некоторого $k < i$, причём для всех $i, 1 \leqslant i \leqslant j, x_i$ и y_i не пересекаются и для каждого подмножества $c \in C$ найдётся такое $z_i, 1 \leqslant i \leqslant j$, что соответствующее множество совпадает с c?

Источник [152]. К этой задаче сводится ВЕРШИННОЕ ПОКРЫТИЕ (см. разд. 3.2.2).

Комментарий. Задача остаётся *NP*-полной даже в том случае, если $|c| \leqslant 3$ для всех $c \in C$. Аналогичная задача, в которой не требуется, чтобы x_i и y_i не пересекались, также *NP*-полна при тех же ограничениях.

[ОП 10] **ОПТИМИЗАЦИЯ ЧИСЛА БИТ МИКРОКОДА**

УСЛОВИЕ. Заданы конечное множество «микрокоманд» A, набор $C = \{c_1, c_2, \dots, c_n\}$ подмножеств множества A (подмножество множества микрокоманд называется микроинструкцией) и положительное целое число K.

ВОПРОС. Существует ли для данного набора микроинструкций K-битовый формат? Иными словами, существует ли такое разбиение множества A на непересекающиеся подмножества $A_1, A_2, \dots, A_n$, что ни одна пара подмножеств A_i и C_j не содержит более одного общего элемента и

$$\sum_{i=1}^{n} \lceil \log_2(|A_i| + 1) \rceil \leqslant K?$$

Источник [451]. К этой задаче сводится 3-С.

А11.2. Программы и схемы

[ОП 11] НЕЭКВИВАЛЕНТНОСТЬ ПРОГРАММ С МАССИВАМИ

УСЛОВИЕ. Заданы конечные множества X, Θ и R переменных, операторов и переменных массивов соответственно, две программы P_1 и P_2, составленные из команд следующего вида: $x_0 \theta x_1 x_2 \dots x_r$ («выполнить»), $\alpha[x_i] \leftarrow x_j$ («изменить»), $x_i \leftarrow \alpha x[j]$ («выбрать»), где $x_i \in X, \theta \in Theta, r$ – число аргументов оператора $\theta, \alpha \in R$, а также конечное множество значений V и интерпретация каждого оператора $\theta \in Theta$ как функции из V^r в V.

ВОПРОС. Можно ли придать переменным из X такие начальные значения из V, что для некоторой переменной из X программы дадут различные конечные значения? (Детали о работе таких программ см. в [104].)

Источник [104]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если в программах отсутствуют команды «выполнить» и имеется только одна переменная массива. Если отсутствуют команды «изменить» или команды «выбрать» или отсутствуют переменные массивов, то задача разрешима за полиномиальное время.

[ОП 12] НЕЭКВИВАЛЕНТНОСТЬ ПРОГРАММ С ПРИСВАИВАНИЯМИ

УСЛОВИЕ. Заданы конечное множество переменных X, две программы P_1 и P_2, представляющие собой последовательности команд присваивания, имеющих вид «$x_0 \leftarrow$, если $x_1 = x_2$ то x_3 иначе x_4», где x_i – переменные из X, и конечное множество значений V.

ВОПРОС. Можно ли всем переменным из X придать такие начальные значения из V, что для некоторой переменной из X программы дадут различные конечные значения? (Детали о работе таких программ см. в [104].)

Источник [104]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если $V = \{0, 1\}$. Она содержится в качестве частного случая во многих задачах о неэквивалентности простых программ и, следовательно, доказывает, что они *NP*-трудные [104, 528].

[ОП 13] **НЕЭКВИВАЛЕНТНОСТЬ ПРОГРАММ С КОНЕЧНОЙ ПАМЯТЬЮ (*)**

УСЛОВИЕ. Заданы конечное множество переменных X, конечный алфавит Σ, две программы P_1 и P_2, каждая из которых представляет собой последовательность команд $I_1, I_2, \dots, I_m$ (P_1 и P_2 не обязательно имеют одинаковую длину m). Команды имеют вид «прочитать x_i», «записать v_j», «$x_i \leftarrow v_j$», «если $v_i = v_k$, перейти к I_i», «принять», «стоп», где $x_i \in X, v_j, v_k \in X \cup \Sigma \cup \{\$\}$ и I_m – команда вида «стоп» или «принять».

ВОПРОС. Существует ли такое слово $w \in \Sigma^*$, что, начав работать на этом слове, программы P_1 и P_2 дадут различные результаты? (Детали о работе программ такого типа см. в [274].)

Источник [274]. К этой задаче сводится ДОПУСКАЕМОСТЬ ЛИНЕЙНО ОГРАНИЧЕННЫМ АВТОМАТОМ.

Комментарий. Задача $P - SPASE$-полна даже тогда, когда P_2 – фиксированная программа, не содержащая команд типа записать и, следовательно, не дающая результата. В указанной выше работе приведено много специальных случаев этой задачи и различных её вариантов, которые или $P - SPACE$-полны или ещё труднее.

[ОП 14] **НЕЭКВИВАЛЕНТНОСТЬ ПРОГРАММ БЕЗ ВЛОЖЕННЫХ ЦИКЛОВ**

УСЛОВИЕ. Заданы конечное множество переменных X, подмножество входных переменных $Y \in X$, выделенная выходная переменная x_0. Две программы с циклами, но без вложенных циклов P_1 и P_2 (т. е. последовательности команд вида «$x \leftarrow y$», «$x \leftarrow x + 1$», «$x \leftarrow 0$», «*цикл* x» и «*конец*», где $x, y \in X$, причём вслед за командой *цикл*, команда *конец* должна предшествовать следующей команде *цикл*).

ВОПРОС. Существует ли такое отображение $f : Y \to Z^+$, сопоставляющее входным переменным начальные значения, что в результате работы обеих программ выходная переменная x_0 получает разные значения? (Подробности о программах этого типа см. в указанных ниже работах.)

Источник [88, 516]. К этой задаче сводится 3-ВЫП. Во второй работе доказана принадлежность задачи классу *NP*.

Комментарий. Если допустить вложенные циклы (даже глубины 2), то задача становится неразрешимой [382]. Если не допускаются циклические команды, то задача разрешима за полиномиальное время [516]. Основной результат обобщен в работе [233].

[ОП 15] **НЕЭКВИВАЛЕНТНОСТЬ ПРОСТЫХ ФУНКЦИЙ**

УСЛОВИЕ. Заданы множество X переменных, два выражения f и g над X. Каждое выражение есть композиция функций из следующего набора: «$s(x) = x+1$», «$p(x) = \max\{x-1, 0\}$», «$plus(x, y) = x+y$», «$div(x, i) = \lceil x/i \rceil$», «$mod(x, t) = x - t \cdot \lceil x/t \rceil$», «$w(x, y) =$ если $y = 0$, то x иначе 0» и «выбрать $(x_1, x_2, \dots, x_n)_i^n = x_i$» где $x, y, x_i \in X; i, n, t \in Z^*$ и $i \leqslant n$.

ВОПРОС. Существуют ли такие неотрицательные целые значения переменных из X, для которых выражения f и g различны?

Источник [516]. К этой задаче сводится НЕЭКВИВАЛЕНТНОСТЬ ПРОГРАММ БЕЗ ВЛОЖЕННЫХ ЦИКЛОВ.

Комментарий. Задача остаётся *NP*-полной, если f и g определяются только в терминах функции $w(x, y)$, или в терминах функций $plus$ и mod, или в терминах функций $plus$ и p [346]. *NP*-полны также варианты этой задачи, в которых f и g определяются только в терминах функций $plus$ и «$sub1(x) = \max\{0, 1 - x\}$» или только в терминах функций «$minus(x, y) = \max\{0, x - y\}$» (где $x, y \in X \cup Z^+$) [88].

[ОП 16] **СИЛЬНАЯ НЕЭКВИВАЛЕНТНОСТЬ СХЕМ ЯНОВА**

УСЛОВИЕ. Заданы конечные множества F и P, состоящие соответственно из символов функций и символов предикатов, одна переменная x и две схемы Янова над F, P и x. Схемой Янова называется последовательность $I_1, I_2, \ldots, I_m$ команд вида «$x \leftarrow f(x)$», «если $p(x)$ то перейти к I_j иначе перейти к I_k» и «стоп», где $f \in F$ и $p \in P$.

ВОПРОС. Верно ли, что две заданные схемы Янова не являются строго эквивалентными? Иными словами, существуют ли область определения D, для каждой функции $f \in F$ интерпретация в виде отображения $f : D \to D$, для каждого предиката $p \in P$ интерпретация в виде отображения $p : D \to \{T, F\}$, начальное значение $x_0 \in D$ переменной x, такие что либо обе заданные схемы останавливаются с различными конечными значениями переменной x, либо одна останавливается, а другая – нет?

Источник [88,461]. К этой задаче сводится 3-ВЫП. Принадлежность классу *NP* доказана во второй работе.

Комментарий. Задача остаётся *NP*-полной даже в том случае, если ни одна из программ не содержит циклов, а P_2 – тривиальная программа, оставляющая значение x неизменным. Задача о сильной эквивалентности схем Янова с двумя переменными неразрешима, даже если предположить, что $|F| = |P| = 1$ [363]. Аналогичные результаты о других свойствах, таких, как «слабая эквивалентность», «дивергенция», «остановка» и т. д., см. в работах [234] и [240]. За полиномиальное время может быть проверена сильная эквивалентность «сильно свободных» схем Янова, т. е. таких схем, в которых между двумя последовательными проверками предикатов применяется хотя бы одна функция (см. [88]). Задача о сильной эквивалентности «свободных» схем Янова в настоящее время открыта.

[ОП 17] **СИЛЬНАЯ НЕЭКВИВАЛЕНТНОСТЬ МОНАДНЫХ РЕКУРСИВНЫХ СХЕМ**

УСЛОВИЕ. Заданы конечные множества F и P, состоящие из символов функций и символов предикатов соответственно, множество G «определённых» символов функций (не пересекающееся с F), выделенный символ $f_0 \in G$ и две линейные монадные рекурсивные схемы S_1 и S_2 (такая схема для каждой $f \in G$ представляет собой набор утверждений вида «$fx =$ если px то αx иначе βx, где $p \in P, \alpha, \beta \in (F \cup G)^*$, причём α и β содержат по крайней мере одно вхождение символа из множества G).

ВОПРОС. Существуют ли множество определения D, интерпретация каждого $f \in F$ в виде отображения $f : D \to D$, интерпретация каждого

$p \in P$ в виде отображения $p : D \to \{T, F\}$ и начальное значение $x_0 \in D$, такие, что два значения $f_0(x_0)$, определяемые согласно рекурсивным схемам S_1 и S_2, либо различны, либо одно из них определено, а другое нет.

Источник [88]. К этой задаче сводится СИЛЬНАЯ НЕЭКВИВАЛЕНТНОСТЬ СХЕМ ЯНОВА. Доказательство принадлежности задачи классу *NP* нетривиально.

Комментарий. Задача остаётся *NP*-полной и в том случае, если одна из схем тривиально полагает $f_0(x) = x$. а другая «линейна справа», т. е. α и β содержат определённый символ (из множества G) в виде крайней оправа буквы. В указанной выше работе приведены другие результаты об *NP*-полноте и *NP*-трудности задач о линейных монадных рекурсивных схемах.

[ОП 18] **ОТНОШЕНИЕ НЕВКЛЮЧЕНИЯ СВОБОДНЫХ B-СХЕМ**

УСЛОВИЕ. Заданы две свободные B-схемы S_1 и S_2. (Свободной B-схемой называется корневой ориентированный ациклический граф $G = (V, A)$, часть вершин которого имеют степень выхода, равную 0 (листья), а остальные (тестовые) – степень выхода 2, причём дуги, исходящие из тестовой вершины, помечены метками L и R.) Кроме того, зафиксировано: (1) множество B символов булевских переменных и каждой тестовой вершине v сопоставлена такая метка $l(v) \in B$, что никакие две тестовые вершины графа, расположенные на общем ориентированном пути, не имеют одинаковых меток; (2) множество F-символов функций, причём каждому листу $v \in V$ сопоставлена метка $l(v) \in F \cup \{\Omega\}$.

ВОПРОС. Верно ли, что схема S_1 не «содержится» в схеме S_2? Иными словами, существует ли такое отображение $t : B_1 \cup B_2 \to \{L, R\}$, что если два пути из корневых вершин графов G_1 и G_2 в листья этих графов, образованные согласно правилу «из тестовой вершины выход по дуге с меткой $t(l(v))$», – заканчиваются в листьях, помеченных символами f_1 и f_2 соответственно, то $f_1 \neq f_2$ и $f_1 \neq \Omega$?

Источник [126]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача о «сильной неэквивалентности свободных B-схем (т. е. неэквивалентности в том же смысле, что и выше, но с одним условием – $f_1 \neq f_2$) остаётся открытой. Если же одна из схем S_1 или S_2 есть упорядоченная B-схема, то последняя задача разрешима за полиномиальное время. Открытый вариант задачи эквивалентен по Тьюрингу задаче о сильной неэквивалентности свободных схем Янова (см. задачу СИЛЬНАЯ НЕЭКВИВАЛЕНТНОСТЬ СХЕМ ЯНОВА).

[ОП 19] **НЕСВОБОДНОСТЬ СХЕМ ПРОГРАММ БЕЗ ЦИКЛОВ**

УСЛОВИЕ. Заданы конечные множества F и P, состоящие из символов функций и символов предикатов соответственно, множество переменных X, монадная без циклов схема программ S над F, P и X. Такая схема состоит из последовательности команд $I_1, I_2, \dots, I_m$ вида «$x \leftarrow f(y)$», если $p(x)$ то перейти к I_j иначе перейти к I_k и «стоп», где $x \in X, f \in F$ и $p \in P$, и в соответствующем «графе переходов» не должно быть ориентированных циклов.

ВОПРОС. Верно ли, что схема S несвободна? Иными словами, существует ли в графе переходов, соответствующем схеме S, ориентированный путь, вдоль которого не может пройти ни одно вычисление по программе S, вне зависимости от того, какая интерпретация придается функциям из F, предикатам из P и какие начальные значения придаются переменным из X?

Источник [88]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной для $|X| = 2$. Если $|X| = 1$, то она разрешима за полиномиальное время. Если допускаются циклы, а $|X|$ произвольно, то задача неразрешима [420].

[ОП 20] **ПРОГРАММЫ С ФОРМАЛЬНО РЕКУРСИВНЫМИ ПРОЦЕДУРАМИ**

УСЛОВИЕ. Заданы конечное множество A идентификаторов процедур, алголоподобная программа P, содержащая описание процедур из A, и обращение к ним (подробности см. в работе, указанной ниже).

ВОПРОС. Верно ли, что некоторая процедура из A формально рекурсивна в программе P (в смысле работы [320])?

Источник [533]. К этой задаче сводится 3-ВЫП.

Комментарий. В указанной работе приведены близкие результаты о задачах, в которых спрашивается: «Обладает ли программа P свойствами «формально самый последний», «корректность формальных параметров», «формальное макросвойство и др?»

А12. РАЗНОЕ

[РАЗ 1] **СООТНОШЕНИЕ «МЕЖДУ»**

УСЛОВИЕ. Заданы конечное множество A и набор упорядоченных троек (a, b, c) различных элементов из множества A.

ВОПРОС. Существует ли такая взаимно-однозначная функция $f : A \to \{1, 2, \ldots, |A|\}$, что для каждой тройки (a, b, c) выполнено одно из свойств: либо $f(a) < f(b) < f(c)$, либо $f(c) < f(b) < f(a)$?

Источник [403]. К этой задаче сводится РАСЩЕПЛЕНИЕ МНОЖЕСТВА.

[РАЗ 2] **ЦИКЛИЧЕСКОЕ УПОРЯДОЧЕНИЕ**

УСЛОВИЕ. Заданы конечное множество A, набор C упорядоченных троек (a, b, c) различных элементов из A.

ВОПРОС. Существует ли такая взаимно-однозначная функция $f : A \to \{1, 2, \ldots, |A|\}$, что для каждой тройки $(a, b, c) \in C$ выполнено одно из свойств: либо $f(a) < f(b) < f(c)$, либо $f(b) < f(c) < f(a)$, либо $f(c) < f(a) < f(b)$?

Источник [138]. К этой задаче сводится 3-ВЫП.

[РАЗ 3] **НЕВЫЖИВАЕМОСТЬ СЕТЕЙ ПЕТРИ СО СВОБОДОЙ ВЫБОРА**

УСЛОВИЕ. Задана сеть Петри $P = (n, M_0, T)$, где $n \in Z^+$, M_0 есть n-мерный вектор с целыми неотрицательными координатами, T – множество переходов вида $< a, b >$, где a и b – это n-мерные $(0-1)$-векторы, причём предполагается выполненным свойство «свободы выбора», состоящее в том, что для каждой пары $< a, b > \in T$ либо у a имеется ровно одна отличная от нуля координата, либо во всех остальных переходах $< c, d > \in T$ вектор c имеет 0 во всех тех позициях, в которых a имеет 1.

ВОПРОС. Верно ли, что P не «выживает»? Иными словами, существует ли такой переход $t \in T$ и такая последовательность σ переходов из T, что для любой последовательности τ переходов из T последовательность $\sigma\tau t$ не «уничтожающая» в M_0? (Последовательность $< a_1, b_1 >, < a_2, b_2 >, \ldots, < a_m, b_m >$ называется уничтожающей в M_0 тогда и только тогда, когда последовательность $M_0, M_1, \ldots, M_{2m}$, в которой $M_{2i+1} = M_{2i} - a_i$ и $M_{2i+2} = M_{2i+1} + b_i, 0 \leqslant i < m$, не содержит векторов с отрицательными координатами.)

Источник [272]. К этой задаче сводится 3-ВЫП. Доказательство принадлежности классу *NP* нетривиально и основано на результатах [193].

[РАЗ 4] **ДОСТИЖИМОСТЬ ДЛЯ 1-КОНСЕРВАТИВНЫХ СЕТЕЙ ПЕТРИ (*)**

УСЛОВИЕ. Заданы 1-консервативнаясеть Петри $P = (n, M_0, T)$, т. е. такая сеть, что для каждого перехода $< a, b > \in T$ векторы a и b имеют одинаковое число единиц, и n-мерный вектор M с неотрицательными координатами.

ВОПРОС. Верно ли, что в сети P вектор M достижим из M_0? Иными словами, существует ли такая последовательность $< a_1, b_1 >, < a_2, b_2 >, \ldots, < a_m, b_m >$ переходов из T, что последовательность $M_0, M_1, \ldots, M_{2m}$, получаемая так же, как в предыдущей задаче, не содержит векторов с отрицательными координатами и $M_{2m} = M$?

Источник [272]. К этой задаче сводится ДОПУСКАЕМОСТЬ ЛИНЕЙНО ОГРАНИЧЕННЫМ АВТОМАТОМ.

Комментарий. Даже в том случае, когда рассмотренная сеть P является сетью Петри со свободой выбора, задача $P - SPACE$-полна. Для произвольных сетей Петри разрешимость задачи не установлена, но известно, что для её решения необходима по крайней мере экспоненциальная память [353]. Аналогичный результат имеет место для задачи о «доминировании»: «Существует ли достижимый из M_0 вектор M', у которого каждая координата не меньше соответствующей координаты вектора M? Близкая задача о «K-ограниченности» (заданы сеть P и целое число K, верно ли, что не существует вектора, достижимого из M_0, у которого все координаты превосходят K?) $P - SPACE$-полна как для произвольных сетей Петри, так и для 1-консервативных сетей со свободой выбора. Дополнительные детали и близкие результаты см. [272, 233].

[РАЗ 5] **ПРЕДСТАВИМОСТЬ КОНЕЧНОЙ ФУНКЦИИ (*)**

УСЛОВИЕ. Заданы конечное множество A, набор F функций $f : A \to A$ и выделенная функция $h : A \to A$.

ВОПРОС. Может ли функция h быть представлена в виде композиции функций из семейства F?

Источник [303]. К этой задаче сводится ПЕРЕСЕЧЕНИЕ КОНЕЧНЫХ АВТОМАТОВ.

Комментарий. Задача $P-SPACE$-полна.

[РАЗ 6] **РАЗЛОЖЕНИЕ ПОДСТАНОВКИ**

УСЛОВИЕ. Заданы подстановка σ множества $\{1,2,\dots,N\}$ и последовательность $S_1,S_2,\dots,S_m$ подмножеств множества $\{1,2,\dots,N\}$.

ВОПРОС. Можно ли представить подстановку σ в виде композиции подстановок: $\sigma=\sigma_1,\sigma_2,\dots,\sigma_m$ (где для каждого i, $1\leqslant i\leqslant m$, подстановка σ_i оставляет неподвижными элементы множества $\{1,2,\dots,N\}\setminus S_i$)?

Источник [153]. К этой задаче сводится ТП-3.

Комментарий. При любом фиксированном N задача разрешима за полиномиальное время.

[РАЗ 7] **ДЕКОДИРОВАНИЕ ЛИНЕЙНЫХ КОДОВ**

УСЛОВИЕ. Заданы $(0-1)$-матрица $A=(a_{ij})$ порядка $m\times n$, $(0-1)$-вектор $\bar y=(y_1,y_2,\dots,y_m)$ и положительное целое число K.

ВОПРОС. Существует ли такой $(0-1)$-вектор $\bar x=(x_1,x_2,\dots,x_n)$, имеющий не более K отличных от нуля координат, что

$$\sum_{i=1}^{n} x_i a_{ij}\equiv y_i(mod 2) \text{ для всех } j,\ 1\leqslant j\leqslant m?$$

Источник [38]. К этой задаче сводится 3-С.

Комментарий. Если вектор $\bar y$ нулевой и, следовательно, ищется «кодирующее слово», у которого вес по Хеммингу не превосходит K, то задача открыта. Вариант задачи, в котором ищется вектор $\bar x$, содержащий ровно K единиц, является *NP*-полным даже тогда, когда $\bar y$ фиксирован и равен 0.

[РАЗ 8] **КОЭФФИЦИЕНТ ИЗБИРАТЕЛЯ ПО ШЕПЛИ-ШУБИКУ**

УСЛОВИЕ. Заданы упорядоченное множество избирателей $V=\{v_1,v_2,\dots,v_n\}$, число голосов $w_i\in Z^+$ для каждого избирателя $v_i\in V$ и квота $q\in Z^+$.

ВОПРОС. Верно ли, что избиратель v_1 имеет коэффициент по Шепли–Шубику, не равный 0? (Коэффициентом $p(v)$ избирателя $v\in V$ называется величина $(1/n!)$, умноженная на число таких подстановок n множества $\{1,2,\dots,n)$, что

$$\sum_{i=1}^{j-1} w_{\pi(i)}<q,\ \sum_{i=1}^{j} w_{\pi(i)}\geqslant q \text{ и } v=v_{\pi(j)}.)$$

Источник [152]. К этой задаче сводится РАЗБИЕНИЕ. Определение коэффициента избирателя взято из работы Шепли и Шубика [485].

Комментарий. Задача вычисления коэффициента заданного избирателя по [485] является KP-полной, однако методом динамического программирования она может быть решена за псевдополиномиальное время.

[РАЗ 9] **СКОПЛЕНИЯ**

УСЛОВИЕ. Заданы конечное множество X, расстояния $d(x, y) \in Z_0^+$ для каждой пары x, y элементов из X и два положительных целых числа K и B.

ВОПРОС. Существует ли такое разбиение множества X на не пересекающиеся множества $X_1, X_2, \ldots, X_k$, что для всех $x, y \in X_i d(x, y) \leqslant B$ при $1 \leqslant i \leqslant k$?

Источник [57]. К этой задаче сводится РАСКРАШИВАЕМОСТЬ ГРАФА В 3 ЦВЕТА.

Комментарий. Задача *NP*-полна, даже если $K = 3$, а все расстояния лежат в интервале [0,1]. При $K = 2$ она разрешима за полиномиальное время. Варианты этой задачи, в которых ищутся такие разбиения, что сумма по всем X_i величин $\max\{d(x, y) : x, y \in X_i\}$ не превосходит B или $\sum_{x,y \in X} d(x, y) \leqslant B$ для всех X_i, также являются *NP*-полными (причём последняя задача *NP*-полна даже при $K = 2$).

[РАЗ 10] **ТЕСТ СЛУЧАЙНОСТИ ДЛЯ ПАР (*)**

УСЛОВИЕ. Заданы последовательность $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ упорядоченных пар целых чисел, неотрицательное целое число K.

ВОПРОС. Существуют ли не менее K подмножеств $S \subseteq \{1, 2, \ldots, n)$, для каждого из которых

$$\sum_{i \in S} |x_i - y_i| \leqslant \sum_{x_i > y_i} (x_i - y_i)?$$

Источник [483]. К этой задаче сводится РАЗБИЕНИЕ.

Комментарий. Принадлежность классу *NP* не установлена. Соответствующая задача перечисления KP-полна, но методом динамического программирования решается за псевдополиномиальное время.

[РАЗ 11] **РАНЖИРОВАНИЕ ПО МАКСИМУМУ ПРАВДОПОДОБИЯ**

УСЛОВИЕ. Заданы матрица $A = (a_{ij})$ размером $n \times n$ с целыми элементами, которые удовлетворяют соотношению $a_{ij} + a_{ji} = 0$ для всех $i, j \in \{1, 2, \ldots, n\}$, и положительное целое число C.

ВОПРОС. Существует ли матрица $B = (b_{ij})$, получаемая из A одновременной перестановкой строк и столбцов, такая, что

$$\sum_{1 \leqslant i < j \leqslant n} min\{b_{ij}, 0\} \geqslant -C?$$

Источник [442]. К этой задаче сводится МНОЖЕСТВО ДУГ, РАЗРЕЗАЮЩИХ КОНТУРЫ.

Комментарий. Задача *NP*-полна в сильном смысле.

[РАЗ 12] **ДОМИНИРУЮЩЕЕ МНОЖЕСТВО МАТРИЦЫ**

УСЛОВИЕ. Заданы $(0-1)$-матрица M размером $n \times n$ и положительное целое число K.

ВОПРОС. Существует ли в M доминирующее множество, состоящее не более чем из K ненулевых элементов? Иными словами, существует ли такое подмножество $C \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ мощности не более K, что $M_{ij} = 1$ для всех $(i, j) \in C$ и если $M_{ij} = 1$, то существует элемент $(i', j') \in C$, для которого или $i = i'$, или $j = j'$?

Источник [540]. К этой задаче сводится МИНИМАЛЬНОЕ ПО МОЩНОСТИ МАКСИМАЛЬНОЕ ПАРОСОЧЕТАНИЕ.

Комментарий. Задача остаётся *NP*-полной даже в том случае, когда M – верхнетреугольная матрица.

[РАЗ 13] **ПОКРЫТИЕ МАТРИЦЫ**

УСЛОВИЕ. Заданы матрица $A = (a_{ij})$ размером $n \times n$ с неотрицательными целыми элементами и целое число K.

ВОПРОС. Существует ли такое отображение $f : \{1, 2, \ldots, n\} \to \{-1, +1\}$, что

$$\sum_{1 \leqslant i,j \leqslant n} a_{ij} f(i) f(j) \geqslant K?$$

Источник [152]. К этой задаче сводится МАКСИМАЛЬНЫЙ РАЗРЕЗ.

Комментарий. Задача *NP*-полна в сильном смысле и остаётся таковой даже в том случае, если требовать положительной определённости матрицы A.

[РАЗ 14] **ПРОСТО ОТКЛОНЯЕМЫЕ ДИЗЪЮНКЦИИ**

УСЛОВИЕ. Задан набор M m-мерных векторов $(M_i[1], M_i[2], \ldots, M_i[m])$, $1 \leqslant i \leqslant n$, координаты которых принимают значения 0, 1 или x.

ВОПРОС. Существует ли такое разбиение множества $\{1, 2, \ldots, m\}$ на непересекающиеся подмножества I, J и такое отображение $f : \{1, 2, \ldots, m\} \to \{0, 1\}$, что если Φ обозначает формулу $\vee_j \in I(M[j] = f(j))$, а Ψ – формулу $\vee_{j \in J}(M[j] = f(j))$, то Φ и Ψ просто отклоняемы в M? (Дизъюнкции Φ и Ψ называются просто отклоняемыми в M, если произведение числа таких $M_i \in M$, что Φ и Ψ истинны для M_i, с числом таких $M_i \in M$, что Φ и Ψ ложны для M_i, больше произведения числа таких $M_i \in M$, что Φ истинна, а Ψ ложна для M_i, с числом таких $M_i \in M$, что Φ ложна, а Ψ истинна для M_i.) Определение понятия «простой отклоняемости» взято из работы [205].

Источник [438]. К этой задаче сводится МАКСИМАЛЬНЫЙ РАЗРЕЗ.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $f(j) = 1$ при $1 \leqslant i \leqslant m$. Если все $M_i[j]$ принимают значения 0 или 1, то задача разрешима за полиномиальное время. Другие близкие результаты можно найти во второй из указанных выше работ.

[РАЗ 15] **ДЕРЕВО РЕШЕНИЙ**

УСЛОВИЕ. Заданы конечное множество объектов X, набор $T = \{T_1, T_2, \ldots, T_m\}$ бинарных тестов $T_i : X \to \{0, 1\}$, положительное целое число K.

ВОПРОС. Существует ли для X дерево решений с использованием тестов T_i, обладающее внешним путем длины, не превосходящей K? (Деревом решений называется бинарное дерево, в котором каждая «невисячая вершина» (не лист) помечена каким-либо тестом из T, а каждая «висячая вершина» помечена каким-либо объектом из X. При этом предполагается также, что ребро, ведущее от «невисячей вершины» к её левому потомку, помечено 0, а к правому – 1, а также если $T_{i_1}, O_{i_1}, T_{i_2}, O_{i_2}, \ldots, T_{i_k}, O_{i_k}$ – последовательность меток вершин и рёбер на некотором пути, ведущем из корня к некоторой «висячей вершине», помеченной объектом $x \in X$, то x – единственный объект, для которого при всех j, $1 \leqslant j \leqslant k$, выполнено соотношение $T_{i_j}(x) = O_{i_j}$. Общей длиной внешнего пути такого дерева называется сумма по всем листьям числа рёбер в пути от корня до этого листа.)

Источник [243]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной даже в том случае, когда для каждого теста $T_i \in T$ существует не более трёх объектов $x \in X$, таких, что $T_i(x) = 1$.

[РАЗ 16] ЛОГИЧЕСКИЙ ПОДГРАФ

УСЛОВИЕ. Задан ориентированный ациклический граф $G = (V, A)$, имеющий ровно одну вершину $s \in V$ со степенью захода, равной нулю; для каждой вершины $v \in V$, степень выхода которой больше нуля, задано значение $f(v) \in \{\text{и, или}\}$, заданы также вес $w(a) \in Z^+$ каждой дуги и целое положительное число K.

ВОПРОС. Существует ли в графе G такой подграф $G' = (V', A')$, что (1) $s \in V'$; (2) если $v \in V'$ и $f(w) = \text{и}$, то все дуги (из множества A), исходящие из v, содержатся в A'; (3) если $v \in V'$ и $f(v) = \text{или}$, то по крайней мере одна из дуг (множества A), исходящих из v, принадлежит A'; (4) сумма весов дуг из множества A' не превосходит K?

Источник [463]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной и тогда, когда $w(a) = 1$ для всех $a \in A$ [152]. Для корневых деревьев задача решается за полиномиальное время методом динамического программирования.

[РАЗ 17] ВЫЯВЛЕНИЕ НЕИСПРАВНОСТИ В ЛОГИЧЕСКОЙ СЕТИ

УСЛОВИЕ. Заданы ориентированный ациклический граф $G = (V, A)$ с единственной вершиной $v^* \in V$, имеющей степень выхода 0; отображение $f : (V \setminus \{v^*\}) \to \{I, \text{и, или, не}\}$, такое, что из $f(v) = I$ следует, что степень захода вершины v равна 0, из $f(v) =$не следует, что степень захода вершины v равна 1, из $f(v) =$и (или $f(v) =$или) следует, что степень захода вершины v равна 2; задано также подмножество $V' \subseteq V$.

ВОПРОС. Можно ли любую одиночную неисправность среди вершин из множества V выявить путем экспериментов со входами и выходами? Иными словами, рассмотрим G как логическую сеть со входными вершинами I, выходом v^* и логическими элементами для реализации функций **и, или, не**, помещёнными в соответствующие вершины. Можно ли для каждой вершины $v \in V'$ и каждого $x \in \{T, F\}$ сопоставить каждой вершине из I такое значение из множества $\{T, E\}$, что выход сети на этом входе отличается от выхода той же сети, в которой логический элемент в вершине v «застопорился» в состоянии x?

Источник [250]. К этой задаче сводится 3-ВЫП.

Комментарий. Задача остаётся *NP*-полной даже тогда, когда $V' = V$ или если V' содержит ровно одну вершину, для которой $f(v) = I$.

[РАЗ 18] **ВЫЯВЛЕНИЕ НЕИСПРАВНОСТИ В ОРИЕНТИРОВАННОМ ГРАФЕ**

УСЛОВИЕ. Заданы ориентированный ациклический граф $G = (V, A)$ (где выделены множество вершин $I \subseteq V$ со степенью захода 0 и $O \subseteq V$ – множество вершин со степенью выхода 0), а также положительное целое число K.

ВОПРОС. Существует ли «множество тестов» мощности, не превосходящей K, позволяющее определить любую «одиночную неисправность» в G? Иными словами, существует ли такое подмножество $T \subseteq T \times O$, что $|T| \leqslant K$ и для каждой вершины $v \in V$ существует пара $\{u_1, u_2) \in T$, для которой вершина v находится на одном из ориентированных путей из u_1 в u_2?

Источник [247]. К этой задаче сводится ТП-3.

Комментарий. Задача остаётся *NP*-полной, даже если $|O| = 1$. Вариант этой задачи, в котором ищется множество T, позволяющее «локализовать» одиночную неисправность (т. е. такое множество тестов T, что для каждой пары вершин $v, v' \in V$ существует тест $(u_1, u_2) \in T$, при котором v находится на одном из ориентированных путей из u_1 в u_2, а v' не находится), также является *NP*-полным для $|O| = 1$. Если $K \geqslant |I| \cdot |O|$, то обе задачи решаются за полиномиальное время.

[РАЗ 19] **ВЫЯВЛЕНИЕ НЕИСПРАВНОСТИ С ПОМОЩЬЮ ТЕСТОВЫХ ТОЧЕК**

УСЛОВИЕ. Заданы ориентированный ациклический граф $G = (V, A)$ (имеющий ровно одну вершину $s \in V$ со степенью захода, равной 0, и ровно одну вершину $t \in V$ со степенью выхода, равной 0), а также положительное целое число K.

ВОПРОС. Можно ли любую «одиночную неисправность» в G локализовать с помощью «подсоединения» к дугам из A не более K «тестовых точек» (т. е. существует ли такое подмножество $A' \subseteq A$, что $|A'| \leqslant K$ и множество тестов

$$T = (\{s\} cup \{u_1 : (u_1, u_2) \in A'\}) \times (\{t\} \cup \{u_1, u_2) \in A'\})$$

обладает тем свойством, что для каждой пары вершин $v, v' \in V \setminus \{s, t\}$ существует такой тест $(u_1, u_2) \in T$, что v находится на одном из ориентированных путей, ведущих от u_1 к u_2, а v' не находится)?

Источник [247]. К этой задаче сводится ТП-3.

Комментарий. Варианты этой задачи, в которых требуется локализовать все одиночные неисправности с использованием не более K «тестовых соединений» или «блокирующих элементов», также являются *NP*-полными. То же самое верно для задачи отыскания при наличии «тестовых точек», «тестовых соединений» или «блокирующих элементов» такого множества тестов T, что $|T| \leqslant K$. (Дальнейшие уточнения см. в указанной выше работе.)

А13. ОТКРЫТЫЕ ЗАДАЧИ

[ОЗ 1] **ИЗОМОРФИЗМ ГРАФОВ**

УСЛОВИЕ. Заданы два графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$.

ВОПРОС. Верно ли, что графы G_1 и G_2 изоморфны? Иначе говоря, существует ли такая взаимно-однозначная функция $f : V_1 \to V_2$, что $\{u, v\} \in E_1$ тогда и только тогда, когда $\{f(u), f(v)\} \in E_2$?

Комментарий[11].

Задача остаётся открытой и в том случае, если G_1 и G_2 принадлежат к следующим типам графов: регулярные, двудольные, графы линий, графы сравнимостей, графы хорд, графы неориентированных путей (т. е. графы пересечений для множества путей в неориентированном дереве), см. [209, 23, 52, 386]. Задача разрешима за полиномиальное время для планарных графов (см., например, [216]) и для графов интервалов.

Для «транзитивных по дугам» кубических графов задача лежит в классе $NP \cap co-NP$ [386]. Задаче ИЗОМОРФИЗМ ГРАФОВ полиномиально эквивалентны следующие задачи: изоморфизм ориентированных графов, изоморфизм бесконтекстных грамматик [235], изоморфизм конечно-определённых алгебр [300], изоморфизм полугрупп [52], изоморфизм конъюнктивных запросов [67], изоморфизм графа своему дополнению [86], изоморфизм и задача о вычислении числа различных изоморфизмов между G_1 и G_2 [24,375]. Частный вариант задачи КЛИКА, полиномиально эквивалентный задаче ИЗОМОРФИЗМ ГРАФОВ, указан в работе [304]. Задачи об изоморфизме групп и латинских квадратов могут быть решены за время $O(n^{\log n})$ [387] и, видимо, проще задачи ИЗОМОРФИЗМ ГРАФОВ.

[ОЗ 2] **ПОДГРАФ, ГОМЕОМОРФНЫЙ ФИКСИРОВАННОМУ ГРАФУ** H

УСЛОВИЕ. Задан граф $G = (V, E)$.

ВОПРОС. Содержит ли G подграф, гомеоморфный графу H (т. е. такой подграф $G' = (V', E')$, который может быть преобразован в граф, изоморфный графу H последовательностью замен вершин степени 2 на ребро, соединяющее её соседей)?

Комментарий[12]. Если включить H в качестве дополнительной переменной в условие задачи, то задача станет *NP*-полной, поскольку в качестве

---

[11] Ряд частных случаев задачи ИЗОМОРФИЗМ ГРАФОВ, разрешимых за полиномиальное время, можно найти в работах [7*, 10*, 1*, 4*, 9*]. – *Дополнение авторов*

[12] Для ориентированных графов аналогичная задача решена в работе [2*]. Была получена полная характеризация графов H, для которых эта задача *NP*-полна или разрешима за полиномиальное время. – *Дополнение авторов.*

частного случая она будет содержать задачу ГАМИЛЬТОНОВ ЦИКЛ. Для некоторых фиксированных графов H (например, если H – треугольник) задача разрешима за полиномиальное время. Существует ли некоторый фиксированный граф H, для которого задача *NP*-полна? Если такого графа не существует, то можно ли указать конкретный граф $H = (U, F)$, для которого *NP*-полна следующая, близкая по смыслу задача: «Заданы граф $G = (V, E)$ и взаимно-однозначное отображение $f : U \to V$. Существует ли подграф $G' = (V', E')$, который указанным выше способом можно преобразовать в граф, изоморфный H, причём так, чтобы отображение f осуществляло искомый изоморфизм?» Известно, что эта последняя задача также является *NP*-полной, если граф H не фиксировать, а включить в условие задачи (поскольку эта задача содержит в качестве частного случая задачу НЕПЕРЕСЕКАЮЩИЕСЯ СОЕДИНЯЮЩИЕ ПУТИ). Для некоторых конкретных графов H, например «треугольник» и «два различных ребра», найдено несколько сложных полиномиальных алгоритмов [321, 487]. Не известно, существует ли конкретное целое число K, такое, что для графа H, состоящего из K непересекающихся рёбер, задача *NP*-полна.

[03 3] **РОД ГРАФА**

УСЛОВИЕ. Заданы граф $G = (V, E)$, положительное целое число K.

ВОПРОС. Можно ли граф G так вложить в поверхность рода K, что никакие два ребра не пересекутся?

Комментарий. При $K = 0$ (в этом случае вопрос заключается в планарности графа C?) задача разрешима за полиномиальное время (см., например, [213]). Полиномиальный алгоритм для кубических графов и $K = 1$ анонсирован в работе [123], а в работе [443] предложен полиномиальный алгоритм для произвольных графов и любого фиксированного значения K. Кроме того, для некоторых специальных классов графов, в том числе для клик, кубов и полных двудольных графов, получены простые замкнутые формулы для рода (см., например, [196]). Хотя для произвольных G и K рассматриваемая задача в настоящее время является открытой, тесно связанная с ней задача ПРОДОЛЖЕНИЕ ВЛОЖЕНИЯ (в которой по заданным G, K и вложению подграфа графа G в поверхности рода K требуется выяснить, можно ли продолжить заданное вложение на весь граф)–*NP*-полна [443]. К числу открытых задач, обобщающих задачу о планарности графа, относятся следующие задачи: «Верно ли, что граф G имеет не более K пересечений, т. е. можно ли G так вложить в плоскость, что будут пересекаться не более K пар рёбер?», «Вернр ли, что толщина графа G не превосходит K, т. е. можно ли множество E так разбить на непересекающиеся множества $E_1, E_2, \ldots, E_k$, $k \leqslant K$, чтобы каждый из подграфов $G = (V, E_i)$ был планарен?» К числу близких *NP*-полных задач относятся задачи ПЛАНАРНЫЙ ПОДГРАФ и ИНДУЦИРОВАННЫЙ ПЛАНАРНЫЙ ПОДГРАФ (см. задачу ИНДУЦИРОВАННЫЙ ПОДГРАФ СО СВОЙСТВОМ П).

[03 4] **ПОПОЛНЕНИЕ ДО ТРИАНГУЛИРОВАННОГО ГРАФА**

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число K.

ВОПРОС. Существует ли множество E' неупорядоченных пар вершин из V, содержащее множество E и удовлетворяющее условию $|E - E'| \leqslant K$, такое, что граф $G' = (V, E')$ – триангулированный, т. е. такой, что для любого простого цикла, содержащего более трёх вершин, существует ребро, не принадлежащее циклу, но соединяющее некоторые две его вершины?

Комментарий. Эта задача эквивалентна неориентированному варианту задачи [ТГ 46] и соответствует минимизации «заполненности» матрицы при применении исключения по Гауссу к симметрической матрице (см., например, [455]). Другая характеризация транзитивных графов дана в работе [166].

[03 5] **ХРОМАТИЧЕСКИЙ ИНДЕКС**[13]

УСЛОВИЕ. Заданы граф $G = (V, E)$ и положительное целое число K.

ВОПРОС. Верно ли, что хроматический индекс графа G не превосходит K? Иначе говоря, существует ли такое разбиение множества E на непересекающиеся подмножества $E_1, E_2, \dots, E_k$, где $k \leqslant K$, что для всех i, $1 \leqslant i \leqslant k$, никакие два ребра из E_i не имеют в G общего конца?

Комментарий. В силу теоремы Визинга [35], хроматический индекс графа G равен либо h, либо $h+1$, где h – максимум степеней вершин в графе G. Поэтому задача может быть переформулирована так: «Задан граф G. Верно ли, что хроматический индекс графа G равен максимуму степеней вершин?» Для двудольных графов ответ всегда положителен [35], причём существуют полиномиальные алгоритмы для того, чтобы в этом случае построить искомое разбиение (см., например, [133]). Частный класс графов, для которого задача открыта, – кубические графы (т. е. регулярные графы степени 3). В последнем случае задачу можно переформулировать следующим образом: «Задан граф G. Можно ли вершины графа G покрыть простыми непересекающимися циклами, имеющими чётное число вершин?» Последняя задача – одна из многих открытых задач, связанных с паритетом. Другая задача этого типа формулируется так: «Задана система C подмножеств конечного множества X. Существует ли такая непустая подсистема $C' \in C$, что каждый элемент $x \in X$ принадлежит чётному числу (может быть и не одному) подмножеств из C'?» Последняя задача эквивалентна открытой задаче, упомянутой в комментарии к задаче ДЕКОДИРОВАНИЕ ЛИНЕЙНЫХ КОДОВ.

[03 6] **ОСТОВНОЕ ДЕРЕВО С ПАРИТЕТОМ**[14]

УСЛОВИЕ. Заданы граф $G = (V, E)$ и разбиение множества E на непересекающиеся двухэлементные множества $E_1, E_2, \dots, E_m$.

ВОПРОС. Существует ли в G такое остовное дерево $T = (V, E')$, что для всех i, $1 \leqslant i \leqslant m$, выполнено условие: или $E_i \subseteq E'$ или $E_i \cap E' = \varnothing$?

[13] В работе [5*] к этой задаче была сведена задача 3-ВЫП, что доказывает *NP*-полноту задачи ХРОМАТИЧЕСКИЙ ИНДЕКС. Эта задача остаётся *NP*-полной даже для кубических графов. – *дополнение авторов.*

[14] В работе [8*] было доказано, что эта задача, как и общая задача о паритете для «представимых» матроидов, разрешима за полиномиальное время. – *Дополнение авторов.*

Комментарий. Эта задача представляет собой типичный частный случай задачи «матроид с паритетом» (см., например, [324]). Последняя задача сама есть обобщение задач о паросочетании в графе и о пересечении двух матроидов, которые разрешимы за полиномиальное время (в случае матроида необходимо предположение о существовании полиномиального алгоритма, распознающего в матроиде независимые множества). Близкая задача об «остовном дереве со многими выборами», в которой ищется такое остовное дерево $T = (V, E')$, что в E' содержится, самое большее, один элемент из каждого множества E_i, представляет собой частный случай задачи о пересечении двух матроидов и, следовательно, может быть решена за полиномиальное время (см. задачу ВЕТВЛЕНИЕ С МНОГИМИ ВЫБОРАМИ).

[ОЗ 7] **РАЗМЕРНОСТЬ ЧАСТИЧНОГО УПОРЯДОЧЕНИЯ**

УСЛОВИЕ. Заданы ориентированный ациклический транзитивный граф $G = (V, A)$ (граф G называется транзитивным, если из того, что $(u, v) \in A$ и $(v, w) \in A$, следует, что $(u, w) \in A$), а также положительное целое число $K \leqslant |V|^2$.

ВОПРОС. Существует ли набор, включающий не более K линейных упорядочений множества V, такой, что $(u, v) \in A$ тогда и только тогда, когда u меньше v в смысле каждого из заданных упорядочений?

Комментарий. При $K = 2$ задача решается за полиномиальное время [327]. Для произвольного K и для фиксированного $K \geqslant 3$ в настоящее время она является открытой.

[ОЗ 8] **РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ ДЛЯ ТРЁХ ПРОЦЕССОРОВ**

УСЛОВИЕ. Заданы множество T заданий единичной длительности, частичный порядок $\lessdot$ на множестве T и директивный срок $D \in Z^+$.

ВОПРОС. Можно ли составить такое расписание выполнения заданий из множества T на 3 процессора, что выполняются все условия предшествования и директивный срок D не будет нарушен? Иными словами, существует ли такое расписание $\sigma : T \to \{0, 1, \ldots, D-1\}$, что соотношение $t \lessdot t'$ влечёт неравенство $\sigma(t) < \sigma(t')$ и для любого $i, 0 \leqslant i \leqslant D-1$, имеется не более трёх заданий $t \in T$, таких, что $\sigma(t) = i$?

Комментарий. Аналогичная задача для двух процессоров решается за полиномиальное время [131,85], даже когда каждое задание имеет индивидуальный директивный срок и момент поступления [147]. Если число процессоров входит в условие как параметр, то задача является *NP*-полной [518]. Дополнительные подробности указаны в комментарии к задаче РАСПИСАНИЕ С ОТНОШЕНИЕМ ПРЕДШЕСТВОВАНИЯ. Не известно, существует ли конкретное значение K, для которого вариант этой задачи с K процессорами *NP*-полон.

[ОЗ 9] **ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ**[15]

[15]) Л. Г. Хачиян [6*] доказал, что эта задача лежит в классе Р. Примененный при этом метод эллипсоидов получил широкое освещение даже в популярных изданиях. Однако практическое значение этого метода ещё предстоит выяснить [3*]. – *Дополнение авторов.*

УСЛОВИЕ. Заданы целочисленные векторы $V_i = (v_i[1], v_i[2], \dots, v_i[n])$, $1 \leqslant i \leqslant m$, $D = (d_1, d_2, \dots, d_m)$, $C = (c_1, c_2, \dots, c_n)$ и целое число B.

ВОПРОС. Существует ли такой вектор $X = (x_1, x_2, \dots, x_n)$ с рациональными координатами, что для всех i, $1 \leqslant i \leqslant m$, выполнены неравенства $V_i \cdot X \leqslant d_i$ и $C \cdot X \geqslant B$?

Комментарий. Задача лежит в пересечении $NP \cap co-NP$ (принадлежность классу $co-NP$ следует из фундаментальной теоремы двойственности линейного программирования). Для любого фиксированного m она разрешима за полиномиальное время. Имеется много вариантов задачи линейного программирования, полиномиально эквивалентных ей (см., например, [448]). Один из них получается, если из условия изъять вектор C, а из вопроса – неравенство $CX \geqslant B$ (см. также [412]). Примеры задач о потоках в сетях, полиномиально эквивалентных задаче ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ, упомянуты в комментариях к задачам НЕОРИЕНТИРОВАННЫЙ ПОТОК С НИЖНЕЙ ОЦЕНКОЙ, ПОТОК В СЕТИ С ОГРАНИЧЕНИЯМИ НА ПУТИ и ДВУХПРОДУКТОВЫЙ ЦЕЛОЧИСЛЕННЫЙ ПОТОК. Обобщение задачи ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ – задача «линейная дополнительность» хотя и лежит в классе NP, но в настоящее время остаётся открытой[16]) [397].

[03 10] **АБСОЛЮТНАЯ УНИМОДУЛЯРНОСТЬ**[17]

УСЛОВИЕ. Задана матрица M размером $m \times n$ с элементами из множества $\{-1, 0, 1\}$.

ВОПРОС. Верно ли, что M – не вполне унимодулярная матрица? Иными словами, существует ли в M квадратная подматрица, определитель которой не принадлежит множеству $\{-1, 0, 1\}$?

Комментарий. Задача открыта и в том случае, когда все элементы матрицы M принадлежат множеству $\{0, 1\}$. Значение вполне унимодулярных матриц для целочисленного программирования обсуждается, например, в работе [324] и книге [161][18].

[ОЗ 11] **СОСТАВНОЕ ЧИСЛО**

УСЛОВИЕ. Задано положительно целое число N.

ВОПРОС. Существуют ли такие целые числа m и $n (m, n > 1)$, что $N = m \cdot n$?

Комментарий. Задача лежит в пересечении $NP \cap co-NP$ [434]. Несмотря на то, что для её решения не известно полиномиального алгоритма, известен алгоритм, работающий полиномиальное время в предположении истинности «расширенной гипотезы Римана»[385]. Такой алгоритм, однако, не известен для отыскания простых делителей числа N. Последняя задача

[16]Задача линейная дополнительность NP-полна. На этот факт нам указали несколько специалистов. – *Дополнение авторов.*

[17]8) В работе [11*] дана элементная теорема о характеризации абсолютно унимодулярных матриц и на её основе построен полиномиальный алгоритм решения задачи АБСОЛЮТНАЯ УНИМОДУЛЯРНОСТЬ. – *Дополнение авторов.*

[18]См. также [226]. – *Прим. ред.*

может оказаться труднее соответствующей задачи распознавания. Конечно, все эти задачи легко решаются за псевдополиномиальное время.

[ОЗ 12] **ТРИАНГУЛЯЦИЯ МИНИМАЛЬНОЙ ДЛИНЫ**

УСЛОВИЕ. Заданы набор $C = \{(a_i, b_i) : 1 \leqslant i \leqslant n\}$ пар целых чисел (определяющих координаты n точек плоскости) и положительное целое число B.

ВОПРОС. Существует ли триангуляция множества точек, определяемого набором C, общая «дискретно-евклидова» длина которой не превосходит B? (Триангуляцией называется набор непересекающихся прямолинейных отрезков, соединяющих по две точки из C каждый, который разбивает внутренность выпуклой оболочки множества C на треугольники. Дискретно-евклидова длина отрезка, соединяющего точки (a_i, b_i) и (a_j, b_j), определяется формулой $\lceil((a_i - a_j)^2_{(}b_i - B_j)^2)^{1/2}\rceil$, а общая длина триангуляции есть сумма длин, образующих её отрезков.)

Комментарий. Аналогичная задача для метрики типа L_1 также открыта. В работе [359] приведены контрпримеры, опровергающие несколько предположений о полиномиальности ряда алгоритмов решения этой задачи, и показано, что близкая задача ОГРАНИЧЕННАЯ ТРИАНГУЛЯЦИЯ является NP-полной.

Приложение В

ЛИТЕРАТУРА

В1. Книги, указанные авторами

1. ABDEL-WAHAB, H. M. [1976], Scheduling with Applications to Register Allocation and Deadlock Problems, Doctoral Thesis, Dept. of Electrical Engineering, University of Waterloo, Waterloo, Ontario. (A5.1)

2. ABDEL-WAHAB, H. M. AND T. KAMEDA [1978], «Scheduling to minimize maxmum cumulative cost subject to series-parallel precedence constraints,» Operations Res. 26, 141-158. (A5.1)

3. ADLEMAN, L., AND K, MANDERS [1977], «Reducibility, randomness, and iniraclability (Abstract),» Proc. 9th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 151-163». (7,1)

ADLEMAN, L. See also MANDERS, K.

4. ADOLPHSON, D. [1977], «Single machine job sequencing with precedence constraints,» SIAM J. Comput. 6, 40-54. (A5.1)

5. ADOLPHSON, D., AND T. C. Hu [1973], «Optimal linear ordering,» SIAM J. Appl. Math. 25, 403-423. (A1.3; A5.1)

6. AHO, A. V., M. R. GAREY, AND F. K. HWANG [1977], «Rectilinear Steiner trees: efficient special case algorithms,» Networks 7, 37-58. (A2.1)

7. AHO, A.V., M. R. GAREY, AND J. D. ULLMAN [1972], «The transitive reduction of a directed graph, SIAM J. Comput. 1, 131-137. (A1.2)

8. AHO, A. V., J. E. HOPCROFT, AND J. D. ULLMAN [1968], «Time and tape complexity of pushdown automaton languages,» Information and Control 13, 186-206. (A 10.1)

9. AHO, A. V., J. E. HOPCROFT, AND J. D. ULLMAN [1974], The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA. (1.3; 2.3; 4.0; 6.1; 7.4)

10. AHO, A. V., S. C JOHNSON, AND J. D. ULLMAN [1977a], «Code generation for expressions with common subexpressions,» J. Assoc. Comput. Mach. 24, 146-160, (A 11.1)

11. AHO, A. V., S. C. JOHNSON, AND J. D. ULLMAN [1977b], private communication, (A11.1)

12. AHO, A. V., Y. SAGIV, AND J. D. ULLMAN [1978], «Equivalences among relational expressions,» unpublished manuscript. (A4.3)

13. AHO, A. V., AND K. SETHI [1977], private communication. (A1.4)

14. AHO, A.V., AND J. D. ULLMAN-[1972], The Theory of Parsing, Translation, and Compiling - Volume 1: Parsing, Prenfice-Hall, Inc., Englewood Gliffs, NJ. (A10.1)

15. AHO, A.V., AND J. D. ULLMAN [1977], private communication. (A4.2)

16. ANGLUIN, D. [1976], An Application of the Theory of Computational Complexity to the Study of Inductive Inference, Doctoral Thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA. (A10.2)

17. ANGLUIN, D. [1977], «On the complexity of minimum inference of regular sets,» unpublished manuscript. (A10.1; A10.2)

18. ANGLUIN, D., AND L. G. VALIANT [1977], «Fast probabilistic algorithms for Hamiltonian circuits and matchings,» Proc. 9th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 30-41. (6.3)

19. APPEL, K., AND W. HAKEN [1977a], «Every planar map is 4-coIorable - 1: Discharging,» Ill. J. Math. 21, 429-490. (4.1)

20. APPEL, K., AND W. HAKEN [1977b], «Every planar map is 4-colorable - 2: Reducibility,» Ill. J. Math. 21, 491-567. (4.1)

21. ARAKI, T., У. SUGIYAMA, T. KASAMI, AND J. OKUI [1977], «Complexity of the deadlock avoidance problem,» Proc. 2nd IBM Symp. on Mathematical Foundations of Computer Science, IBM Japan, Tokyo, 229-252. (A5.3)

ARAKI, T. See also SUGIYAMA, Y.

22. ARJOMANDI, E. [1977], private communication. (A1.1)

23. BABAI, L. [1976], private communication. (7.1; A13)

24. BABAI, L. [1977], private communication. (A13)

25. BAKER, T. J. GILL, AND R. SOLOVAY [1975], «Relativizations of the P=?NP question,» SIAM J. Comput.A, 431-442. (7.6)

26. BAKER, T.P., AND A. U SELMAN [1976], «A second step toward the polynomial hierarchy,» Proc. 17th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 71-75. (7.6)

27. BALL, M. O. [1977a], Network Reliability and Analysis: Algorithms and Complexity, Doctoral Thesis, Operations Research Dept., Cornell University, Ithaca, NY.1 (A2.2)

28. BALL, M. 0. [1977b], private communication. (A2.2)

29. BARROW, H. G., AND R. M. BURSTALL [1976], «Subgraph isomorphism, matching relational structures and maximal cliques,» Information Processing Lett. 4, 83-84. (A 1.4)

30. BARTHOLDI, J. J., Ш, J. B. OKLIN; AND H. D. RATLIFF [1977], "Circular ones and cyclic staffing,» Report No. 21, Dept. of Operations Research, Stanford University, Stanford, CA. (A5.4)

BARZDIN, Y., M. See TRAKHTENBROT, B. A.

31. BAUER,M., D. BRAND, M. FISCHER, A. MEYERJ AND M. PATERSON [1973], «A note on disjunctive form tautologies,"SIGACT News 5:2, 17-20. (7.4)

32. BAXTER, L.D. [1976], The Complexity of Unification, Doctoral Thesis, Dept. of Computer Science, University of Waterloo, Waterloo, Ontario. (A9.2)

33. BAXTER, L.D. [1977], «The NP-compIeteness of subsumption,» unpublished manuscript. (A9.2)

34. BEERI, C, AND P. A. BERNSTEIN [1978], «Computational problems related to the design of normal form relational schemes,» unpublished manuscript. (A4.3)

BEERI, C See also BERNSTEIN, P. A.; SHAMIR, E.

35. BERGE, C. [1973], Graphs and Hypergraphs, North-Holland, Amsterdam. (A13)

36. BERGER, R. [1966], The Undecidability of the Domino Problem (Mem. Amer. Math. Sot, No. 66), American Mathematical Society, Providence, RI. (1.4; A8)

37. BERLEKAMP, E. R. [1976], private communication. (A2.1; A8)

38. BERLEKAMP, E. R., R. J. MCELIECE, AND H. C. A. VAN TILBORG [1978], «On the inherent intractability of certain coding problems,» IEEE Trans. Information Theory (to appear). (A 12)

39. BERMAN, Lю, AND J. HARTMANIS [1977], «On isomorphisms and density of NP and other complete sets,» SIAM J. Comput. 6, 305-322. (7.1)

BERMAN, L. See also HARTMANIS, J.

40. BERNSTEIN, P. A., AND C. BEERI [1976], «An algorithmic approach lo normalization 1 of relational database schemas,» Report CSRG-73, Computer Systems Research Group, University of Toronto, Canada. (A4.3)

BERNSTEIN, P. A. See also BEERI, C; PAPADIMITRIOU, C. H.

BLATTNER, W. O. See DANTZIG, G. B.

41. BLAZEWICZ, J. [1976], «Scheduling dependent tasks with different arrival times to meet deadlines,» in H. Beilner and E. Gelenbe (eds.), Modelling and Performance Evaluation of Computer Systems, North Holland, Amsterdam, 57-65. (A5.1)

42. BLAZEWICZ, J. [1977a], «Mean flow lime scheduling under resource constraints,» Report No. PR-19/77, Institute of Control Engineering, Technical University of Poznan, Poland. (A5.2)

43. BLAZEWicz, J. (1977b], «Scheduling with deadlines and resource constraints,» Report PR-25/77, Institute of Control Engineering, Technical University of Poznan, Poland. (A5.2)

44. BLAZEWICS, J. [1978], «Deadline scheduling of tasks with ready times and resource constraints,» unpublished manuscript. (A5.2)

45. BOESCH, F. T., S. CHEN, AND J. A. M. MCHUGH [1974], «On covering the points of a graph with point disjoint paths,» in Graphs and Combinatorics (Proc. Capitol Conf. on Graph Theory and Combinatorics), Lecture Notes in Math., Vol. 46, Springer, Berlin, 201-212. (A1.2)

4б. BOOK, R. V. [1972], «On languages accepted in polynomial time,» SIAM J. Comput. 1, 281-287. (7.5; A10.1)

47. BOOK, R. V. [1974], «Comparing complexity classes,» J. Comput. System Set 9, 213-229. (7.5)

48. BOOK, R. V. [1976], «Translational lemmas, polynomial time, and $(log_n)^j$-space,» Theor. Comput. Sci. 1, 215-226. (7.5)

49. BOOK, R.V. [1978], «On the complexity of formal grammars,» Acta Informat. 9, 171-182. (A10.2)

50. BOOK, R. V., AND S. GREIBACH [1970], «Quasi-realtime languages,» Math. Systems Theory 4, 97-111. (A10.1)

51. BOOTH, K. S. [1975], PQ Tree Algorithms, Doctoral Thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA. (A4.2)

52. BOOTH, K. S. [1978], «Isomorphism testing for graphs, semigroups, and finite automata are polynomial equivalent problems,» SIAM J. Comput. 7, 273-279. (7.1; A13)

53. BOOTH, K. S., AND.G. S. LUEKER [1975], «Linear algorithms to recognize interval graphs and test for the consecutive ones property,» Proc. 7th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 255-265. (A13)

54. BOOTH, K.S., AND G.S. LUEKER [1976], «Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms,» J. Comput. System Set. 13, 335-379. (A1.2)

55. BOROSH, I., AND L. B. TREYBIG [1976], «Bounds on positive integral solutions of linear Diophantine equations,» Proc. Amer. Math. Soc. 55, 299-304. (A6)

BRAND, D. See BAUER, M.

56. BROOKS, R. L. [1941], «On coloring the nodes of a network,» Proc. Cambridge Philos. Soc. 37, 194-197. (4.1; A1.1)

57. BRUCKER, P. [1978], «On the complexity of clustering problems,» in R. Henn, B. Korle, and W. Oletti (eds.), Optimierung und Operations Research, Lecture Notes in Economics and Mathematical Systems, Springer, Berlin (to appear). (A12)

58. BRUCKER, P., M. R. GAREY, AND D. S. JOHNSON [1977], «Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness,» J. Math. Oper. Res. 2, 275-284. (A5.2)

BRUCKER, P. See also LENSTRA, J. K.

59. BRUNO, J., E: G. COFFMAN, JR, AND R. SETHI [1974], «Scheduling independent tasks to reduce mean finishing time,» Comm. ACM 171 382-387. (A5.2)

60. BRUNO, J., AND P. DOWNEY [1978], «Complexity of task scheduling with deadlines, set-up times and changeover costs,» SIAM J; Comput. (to appear). (A 5.1)

61. BRUNO, J., AND R. SETHI [1976], «Code generation for a one-register

machine,» J. Assoc. Comput. Mach. 23, 502-510; (A11.1)

62. BRUNO, J., AND L. WEINBERG [1970], «A constructive graph-theoretic solution of the Shannon switching game,» IEEE Trans. Circuit Theory CT-17, 74-8l. (A8)

BURKHARD, W. A. See WALSH, A. M.

63. BURR, S. [1976], private communication. (A1.1)

64. BURR, S., P.ERDOS, AND L. LOVASZ [1976]; «On graphs of Ramsey type,» ArS Combinatorica 1, 167-190. (A1.1)

BURSTALL, R, M. See BARROW, H. G.

65. CARLIER, J. [1978], «Probleme a une machine,» Report No. 78.05, Institut de Programmation, Universite de Pierre et Marie Curie, Paris, France, (A5.1)

66. CHAN, T. [1977], «An algorithm for checking PL/CV arithmetic inferences,» Report No. 77-326, Dept. of Computer Science, Cornell University, Ithaca, NY. (A 9.2)

67. CHANDRA, A. K. AND P. M. MERLIN [1977], «Optimal implementation of conjunctive queries in relational data bases,» Proc. 9th Ann. ACM Symp. on Theory of Computing, Associalion for Computing Machinery, New York, 77-90. (A4.3; A13)

68. CHANDRA, A. K., AND L. J, STOCKMYER [1976], «Alternation,» Proc. 17th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach; CA, 98-108. (7.6)

CHANDRA, A. K. See also STOCKMEYER, L. G. CHANDY, K. M.

See VAN SICKLE, L. CHEN, S. See BOESCH, F. T.

69. CHO, Y., AND S. SAHNI [1978], «Preemptive scheduling of independent jobs with release and due times on open, flow, and job shops,» unpublished manuscript, (A 5.3)

CHO, Y. See also SAHNI, S.

70. CHOMSKY, N., AND G. A. MILLER [1958], «Finite state languages,» Information and Control 1, 91-112. (A10.2)

71. CHRISTOFIDES, N. [1976], «Worst-case analysis of a new heuristic for the travelling salesman problem,»Technical Report, Graduate School of .Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA. (6.1)

72. GIUNC, F. R. K., AND R. L. GRAHAM [1977], private communication.

(A 1.2)

73. CHVATAL, V. [1973], «On the computational complexity of finding a kernel,» Report No. CRM-300, Centre de Recherches Mathematiques, Universite de Montreal. (A1.5)

74. CHVATAL, V. [1975], «On certain polytopes associated with graphs,» J. Combinatorial Theory Ser. B 18, 138-154. (A6)

75. CHVATAL, V. [1976], private communication. (A1.2; A 1,3)

76. CHVATAL, V. [1977], «Determining the stability number of a graph,» SIAM J. Comput. 6, 643-662. (7.6)

77. CHVATAL, V. [1978], private communication. (A3.1)

78. CHVATAL, V., AND P. L. MAMMER [1975], «Aggregation of inequalities in integer programming,» Report No. STAN-CS-75-518, Computer Science Dept., Stanford University, Sianford, CA. (A1.5)

79. CHVATAL, V., AND G. THOMASSEN [1978], «Distances in orientations of graphs,» J. Combinatorial Theory Ser. B 24, 61-75. (A1.5)

80. Cobham, A. [1964], «The intrinsic computational difficulty of functions,» in V. Bar-Hillel (ed.), Proc. 1964 International Congress for Logic Methodology and Philosophy of Science, North Holland, Amsterdam, 24-30. (1.3; 5.2)

81. COCKAYNE, E., S. GOODMAN, AND S. HEDETNIEMI [1975], «A linear algorithm for the domination number of a tree,» Information Processing Lett. 4, 41-44. (A 1.1)

82. COCKAYNE, E. J., AND S, T. HEDETNIEMI 11975], «Optimal domination in graphs,» IEEE Trans, Circuits and Systems CAS-22, 855-857. (A1.1)

83. COCKAYNE, B. J., S. T. HEDETNIEMI, AND P. J. SLATER [1978], private communication. (A2.5)

84. CODY, R. A., AND E. G. COFFMAN, JR [1976], «Record allocation for minimizing expected retrieval costs on drum-like storage devices,» J. Assoc. Comput. Mach.23, 103-115. (A4.1)

85. COFFMAN, E. G., JR, AND R. L. GRAHAM (I972I, «Optimal scheduling for two-processor systems,» Acta Informat. 1, 200-213. (A5.2; A13)

COFFMAN, E. G., JR. See also BRUNO, J.; CODY, R. A.; MUNTZ, R. R.

86. COLBOURN, M. J., AND C. J. COLBOURN [1978], «Graph isomorphism and self-complementary graphs,» SIGACT News 10:1, 25-29. (A13)

COLBOURNE, C. J. See COLBOURNE, M. J.

87. COMER, D., AND R. SETHI [1976], «Complexity of Trie index construction (Extended abstract),» Proc. 17th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 197-207. (A4.1)

88. CONSTABLE, R. L., H. B. HUNT, III, AND S. SAHNI [1974], «On the computational-complexity of scheme equivalence,» Report No. 74-201, Dept. of Computer Science, Cornell University, Ithaca, NY. (Extended abstract appeared in Proc. 8th Am Princeton Conf. on Information Sciences and Systems, Dept. of Electrical Engineering, Princeton University, Princeton, NJ, 15-20). (A4.2; A11.2)

89. CONWAY, J. H, [1976], On Numbers and Games, Academic Press, New York. (A8)

90. CONWAY, R. W., W. L. MAXWELL, AND L. W. MILLER [1967], Theory of Scheduling; Addison-Wesley, Reading, MA. (A5.2)

91. COOK, S. A. [1971a], «The complexity of theorem-proving procedures,» Proc. 3rd Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 151-158. (1.5; 2.6; 3.1.1; 5.2; A 1.4; A9.1)

92. COOK, S. A. [1971b], «Characterizations of pushdown machines in terms of time-bounded computers,» J. Assoc. Comput. Mach.18, 4-18. (7.5).

93. COOK, S. A. [1973], «A hierarchy for nondeterministic time complexity,» J. Comput. System Set. 7, 343-353. (7.6)

94. COOK, S. A. [1974], «An observation on time-storage trade off,» J. Comput. System, Sci. 9, 308-316. (7.5)

95. COOK, S., AND R. SETHI [1976], «Storage requirements for deterministic polynomial lime recognizable languages,» J. Comput. System Sci. 13, 25-37 (7.5).

96. CORNUEJOLS, G., M. L. FISHER, AND G. L. NEMHAUSER [1977], «Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms,» Management Sci. 23, 789-810. (6.1)

97. CORNUEJOLs, G., AND G. L. NEMHAUSER [1978], «Tight bounds for Christofides' traveling salesman heuristic,» Math. Programming 14; 116-121. (6.1)

CRESSWELL, M. J. See HUGHES, G, E.

CULIK, K., II See OPATRNY, J.

98. DANTZIG, G. B. [1957], «Discrete-variable extremum problems,» Operations Res. 5, 266-277. (4.2.2; A6)

99. DANTZIG, G.B. [1960], «On the significance of solving linear program-

ming problems with some integer variables,» Econometrika 282 30-44, (1.5)

100. DANTZIG, G.B., W. O. BLATTNER, AND M. R. RAO [1967], «All shortest routes from a fixed origin in a graph,» in Theory of Graphs: International Symposium, Gordon and Breach, NY, 85-90. (1.5)

101. DATE, C. J. [1975], An Introduction to Database Systems, Addison-Wesley, Reading, MA. (A4.3)

DEMERS, A. See JOHNSON, D. S.

DEO, N. See KRISHNAMOORTHY, M. S.; REINGOLD, E. M.

102. DOBKIN, D., AND R. E. LADNER [1978], private communication. (A 8)

103. DOBKIN, D., R. LIPTON, AND S. RE1SS [1976], «Linear programming is P-complete,» in (same authors), «Excursions into geometry,» Report No. 71, Dept. of Computer Science, Yale University, New Haven, CT. (7.5)

DOBKIN, D. See also REISS, S. P.

DORIMAN, Y. G. See ORLOVA, G. I.

104. DOWNEY, P, J., AND R. SETHI [1976], «Assignment commands and array structures,» Proc. 17th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 57-66. (All.2)

DOWNEY, P. J. See also BRUNO. J.

EDELBERG, M. See HlRSCHRERG, D.

105. EDMONDS, J. [1962], «Covers and packings in a family of sets,» Bull. Amer. Math. Soc. 68, 494-499. (1.5)

106. EDMONDS, J. [1965a], «Paths, trees, and flowers,» Canad. J. Math. 17, 449-467, (1.3; 5.2)

107. EDMONDS, J. [1965b], «Minimum partition of a matroid into independent subsets» J. Res. Nat. Bur. Standards Sect. B 69, 67-72. (5.2)

108. EDMONDS, J., AND E. L. JOHNSON [1970], «Matching: a well-solved class of integer linear programs,» in Combinatorial Structures and their Applications, Gordon and Breach, New York, 89-92. (All; A1.2)

109. EDMONDS, J., AND E. L. JOHNSON [1973], «Matching, Euler tours, and the Chinese postman,» Math. Programming 5, 88-124. (A2.3)

110. EDMONDS, J., AND R. M. KARP [1972], «Theoretical improvements in algorithmic efficiency for network flow problems,» J. Assoc. Comput. Mach.

19, 248-264. (A2.1)

111.EDMONDS, J., AND D. W. MATULA [1975], private communication. (4.2.2; A1.4)

112. EHRLICH, G., S. EVEN, AND R. E. TARJAN [1976], «Intersection graphs of curves in the plane,» J. Combinatorial Theory Ser. B 21, 8-20. (A 1.1)

ERDOS, P. See BURR, S.

113. EsWAREN, K. P., AND R. E. TARJAN [1976], «Augmentation problems,» SlAM J. Comput. 5, 653-665. (A2.2)

114. EVEN, S., A. ITAI, AND A. SHAMIR [1976], «On the complexity of timetable and multicommodity flow problems,» SIAM J. Comput. 5, 691-703. (3.1.1; 3.2.3; A2.4; A5.3; A9.1)

115. EVEN, S., AND D, S. JOHNSON [1977], unpublished results. (A2.1; A2.4)

116. EVEN, S., D. I, LICHTENSTEIN, AND Y SHILOACH [1977], «Remarks on Zeigler's method for matrix compression,» unpublished manuscript. (A4.2)

117. EVEN, S., A. PNUELI, AND A. LEMPEL [1972], «Permutation graphs and transitive graphs,» J. Assoc. Comput. Mach. 19, 400-410. (A1.1; A1.2)

118. EveN, S., AND Y. SHILOACH [1975], «NP-completeness of several arrangement problems,» Report No. 43, Dept. of Computer Science, Technion, Haifa, Israel. (A 1.3)

119. EVEN, S., AND R. E. TARJAN [1976], «A combinatorial problem which is complete in polynomial space,» J. Assoc. Comput. Mach. 23, 710-719. (7.4; A8)

EVEN, S. See also EHRLICH, G.

120. FAGIN, R. [1974], «Generalized first-order spectra and polynomial time recognizable sets,» in R. M. Karp (ed.). Complexity of Computation, American Mathematical Society, Providence, Rl, 43-73. (A4.2)

121. FARLEY, A., S. HEDETNIEMI, S. MITCHELL, AND A. PROSKUROWSKI [1977], «Minimum broadcast graphs,» Report No. CS-TR-77-2, Dept. of Computer Science, University of Oregon, Eugene, OR. (A2.5)

122. FiscHER, M. J., AND M. O. RABIN [1974], «Super-exponential complexity of Pres-burger arithmetic,» in R. M. Karp (ed.). Complexity of Computation, American Mathematical Sociely, Providence, RI, 27-41. (1.4; 7.6)

FISCHER, M. J. See also BAUER, M.; SEIFERAS, J. I.; WAGNER, R. A.

FISHER, M. L. See CORNUEJOLS, G.; NEMUAUSER, G. L.

123. FILOTTI, I. S. [1978], «An efficient algorithm for determining whether a cubic graph is toroidal,» Proc. 1Oth Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 133-142. (A 13)

124. FLORIAN, M., AND M. KLEIN [1971], «Deterministic production planning with concave costs and capacity constraints,» Management Sci. 18, 12-20. (A5.4)

FLORIAN, M. See also LENSTRA, J. K.

125. FORD, L. R., AND D. R. FULKERSON [1962], Flows in Networks, Princeton University Press, Princeton, NJ. (A2.4)

126. FORTUNE, S., J. E. HOPCROFT, AND E. M. SCHMIDT [1977], «The complexity of equivalence and containment for free single variable program schemes,» Report No. TR77-310, Dept. of Computer Science, Cornell University, Ithaca, NY. (A 11.2)

127. FRAENKEL, A. S., M. R. GAREY, D. S. JOHNSON, T. SCHAEFER, and Y. YESHA [1978], «The complexity of Checkers on an $N \times N$ board - Preliminary report,» Proc. 19th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 55-64. (7.4; A8)

128. FRAENKEL, A. S., and Y. YESHA [1976], «Theory of annihilation games,» Bull. Amer. Math. Soc. 82, 775-777. (A8)

129. FRAENKEL, A. S., and Y. YESHA [1977], «Complexity of problems in games, graphs, and algebraic equations,» unpublished manuscript. (A1.4; A 1.5; A7.2;A8)

130. FREDERICKSON, G. N., M. S. HECHT, and C. E. KIM [1978], «Approximation algorithms for some routing problems,» SIAM J. Comput. 7, 178-193. (A2.3)

131. FUJU, M., T. KASAMI, and K. NINOMIYA 11969], «Optimal sequencing of two equivalent processors,» SIAM J. Appl. Math. 17, 784-789. Erratum [1971], SIAM J. Appl. Math. 20, 141. (A13)

132. FULKERSON, D. R., AND D. A. GROSS [1965], «Incidence matrices and interval graphs,» Pacific J. Math. 15, 835-855. (A1.2)

FULKERSON, D. R. See also FORD, L. R.

FULLER, S. H. See STONE, H. S.

133. GABOW, H. N. [1976], «Using Euler partitions to edge color bipartite muifigraphs,» Internal. J. Comput. Information Sci. 5, 345-355. (A 13)

134. GABOW, H. N., S. N. MAHESHWARI, and L. OSTERWEIL [1976], «On two problems in the generation of program test paths,» IEEE Trans. Software Engrg. SE-2, 227-231. (A1.5)

135. GALIL, Z, [1974], «On some direct encodings of nondeterminislic Turing machines operating in polynomial time into P-complete problems,» SIGACT News 6:1, 19-24. (7.3)

136. GALIL, Z. [1976], «Hierarchies of complete problems,» Acta Informat. 6, 77-88. (7.5;A10.1)

137. GALIL, Z. [1977], «On resolution with clauses of bounded size,» SIAM J. Cornput. 6, 444-459. (7.6)

138. GALIL, Z., and N. Megiddo [1977], «Cyclic ordering is NP-complete,» Theor. Comput. Sci. 5, 179-182. (A12)

139. GAREY, M. R. [1973], «Optimal task sequencing with precedence constraints,» Discrete Math. 4, 37-56. (A5.1)

140. GAREY, M. R., F. GAVRIL, and D. S. JOHNSON [1977], unpublished results. (A1.2)

141. GAREY, M. R., R. L. GRAHAM, and D. S. JOHNSON [1976], «Some NP-complete geometric problems,» Proc. 8th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 10-22. (A2.3)

142. GAREY, M. R., R.L. GRAHAM, and D. S. JOHNSON [1977], «The complexity of computing Steiner minimal trees,» SIAM J. Appl. Math. 32, 835-859. (A2.1)

143. GAREY, M. R., R. L. GRAHAM, D. S. JOHNSON, and D.E. KNUTH [1978], «Complexity results for bandwidth minimization,» SIAM J. Appl. Math. 34, 477-495, (3.2.2; A1.3)

144. GAREY, M. R., R. L. GRAHAM, D. S. JOHNSON, and A. C., YAO [1976], «Resource constrained scheduling as generalized bin packing,» J. Combinatorial Theory Ser. A 21,257-298. (6.1)

145. GAREY, M.R., and D.S. JOHNSON [1975], «Complexity results for multiprocessor scheduling under resource constraints,» SIAM J. Comput. 4, 397-411. (A3.2; A 5.2)

146. GAREY, M.R., and D.S. JOHNSON [1976a], «The complexify of near-optimal graph coloring,» J. Assoc. Comput. Mach. 23, 43-49. (6.2)

147. GAREY, M. R., and D, S. JOHNSON [i976bj, «Approximation algorithms for combinatorial problems: an annotated bibliography,» in J. F. Traub (ed.), Algorithms and Complexity: New Directions and Recent Results, Academic

Press, New York, 41-52. (6.2)

148. GAREY M. R., and D. S. JOHNSON [1976c], «Scheduling tasks with nonuniform deadlines on two processors,» J. Assoc. Conmput. Mach. 23, 461-467.
(A5.1;A5.2)

149. GAREY, M. R., and D. S. JOHNSON [1977a], 'The rectilinear Steiner tree problem is NP-complete,» SIAM J. Appl. Math. 32, 826-834. (AH; A1.2; A2.1)

150. GAREY, M. R., and D, S. JOHNSON [1977b], «Two-processor scheduling with starttimes and deadlines,» SIAM J. Comput. 6, 416-426. (A5.1; A5.2; A13)

151. GAREY, M. R., and D. S. JOHNSON [1978], «Strong NP-completeness results: motivation, examples, and implications,» J. Assoc. Comput. Mach. 25, 499-508. (5.2; 6.2)

152. GAREY, M. R., and D. S. JOHNSON [—], unpublished results. (A1.1; A1.2; A1.4; A1.5;A2.1; A2.2; A2.5; A3.1; A3.2; ASA; A6; A7.1; A11.1; A12)

153. GAREY, M. R., D. S. JOHNSON, G. L. Miller, and C. H. Papadimitriou [1978], unpublished results. (A1.1; A11.1; A12)

154. GAREY, M. R., D. S. JOHNSON, and C. H. PAPADIMITRIOU [1977], unpublished results. (A1.3;A8)

155. GAREY, M. R., D. S. JOHNSON, and R. SETHI [1976], «The complexity of flowshop and jobshop scheduling,» Math., Oper. Res. 1, 117-129. (3.2.2; A5.3)

156. GAREY, M. R., D. S. JOHNSON, B. B. SIMONS, and R. E. TARJAN [1978], «Scheduling unit time tasks with arbitrary release times and deadlines,» unpublished manuscript. (A5.1)

157. GAREY, M. R., D. S. JOHNSON, and L. Stockmeyer [1976], «Some simplified NP-compleie graph problems,» Theory Comput. Sci. 1, 237-267. (3.2.2; 4.1; A1.1; A1.2; A1.3; A2.2; A9.1)

158. GAREY, M. R., D. S. JOHNSON, and R. E. Tarjan [1976a], «The planar Hamiltonian circuit problem is NP-complete,» SIAM J. Comput. 5, 704-714. (3.2.3; A1.3)

159. GAREY, M. R., D. S. JOHNSON, and R. E. Tarjan [1976b], unpublished results. (A1.1)

GAREY, M. R. See also Aho, A. V.; Brucker, P.; Fraenkel, A. S.; JOHNSON, D. S.

160. Garfinkel, R. S. [1977], «Minimizing wallpaper waste. Part 1: a class of traveling salesman problems,» Operations Res. 25, 741-751. (A2.3)

161. GARFINKEL, R. S., and G. L. Nemhauser [1972], Integer Programming, John Wiley & Sons, New York. (6.0; A2.3; A13)

162. GAVETT, J. [1965], «Three heuristic rules for sequencing jobs to a single production facility,» Management Sci. 11, B166-B176. (6.1)

163. Gavril, F. [1972], «Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph,» SIAM J. Comput. 1, 180-187. (A1.1; A1.2; A4.1)

164. Gavril, F. [1973], «Algorithms for a maximum clique and a maximum independent set of a circle graph,» Networks 3, 261-273. (A1.2)

165. GavriL, F. [1974a], «Algorithms on circular-arc graphs,» Networks 4, 357-369. (A1.1; A1.2)

166. Gavril, F. [1974b], «The interseclion graphs of subtrees in trees are exactly the chordal graphs,» Combinatorial Theory Ser. B 16, 47-56. (A13)

167. Gavril, F. [1974c], private communication. (6.1)

168. GAVRIL, F. [1977a), «Some NP-complete problems on graphs,» Proc. 11th Conf. on Information Sciences and Systems, Johns Hopkins University, Baltimore, MD, 91-95. (A1.1; A1.2; A1.3; A2.5; A4.1)

169. Gavril, F.; [1977b], private communication. (A1.2)

Gavril, F. See also GAREY, M. R.; Yannakakis, M.

Geldmacher, R. C. See Liu, P. C.

170. Geoffrion, A. M. [1974], «Lagrangian relaxation and its uses in integer programming, 1» Math. Prog, Study 2, 82-114. (6.0)

171. GlLL, J. T., III [1977], «Computational complexity of probabilistic Turing machines,» SIAM J. Comput. 6, 675-695. (7.3)

Gill, J. T., III. See also Baker, T.

172. GILMORE, P. C, AND R.E. Gomory [1964], «Sequencing a one state-variable machine: a solvable case of .the traveling salesman problem,» Operations Res. 12, 655-679. (A2.3;A5.3)

173. GIMPEL, J. F. [1965], «A method of producing a Boolean function having an arbitrarily prescribed prime implicant table,» IEEE Trans. Computers 14, 485-488. (1.5;A9.1)

174. GOLD, E.M. [1974], «Complexity of automaton identification from given data,» unpublished manuscript. (A9.1; A10.1)

175. Gold, E. M. [1978], «Deadlock protection: easy and difficult cases,» SIAM J. Comput. 7, 320-336. (A5.3)

176. Gol'dberg, M. K., and I. A. Klipker [1976], «Minimal placing of trees on a line,» Technical Report, Physico-Technical Institute of Low Temeperatures, Academy of Sciences of Ukranian SSR, USSR (in Russian). (A1.3)

177. Goldschlager, L. M. [1977], «The monotone and planar circuit value problems are log space complete for P,» SIGACT News 9:2, 25-29. (7.5)

178. Golumbic, M. C. [1977], «The complexity of comparability graph recognition and coloring;» Computing 18, 199-208. (A1.1; A1.2)

Gomory, R. E. See Gilmore, P. C.

179. Gonzalez, T. [1977], «Optimal mean finish time preemptive schedules,» Report No. 220, Computer Science Dept., Pennsylvania State University, University Park, PA. (A5.2)

180. Gonzalez, T., E. L. Lawler, and S. Sahni [1978], «Optimal preemptive scheduling of a fixed number of unrelated processors in polynomial time,» unpublished manuscript. (A5.2)

181. GONZALEZ, T., and S. Sahni [1976], «Open shop scheduling to minimize finish time,» J. Assoc. Comput Much; 23, 665-679. (A5.3)

182. Gonzalez, T., and S. Sahni [1978a], «Flowshop and jobshop schedules: complexity and approximation,» Operations Res. 26, 36-52. (A5.3)

183. Gonzalez, T., and S. Sahni [I978b], «Preemptive scheduling of uniform processor systems,» J. Assoc. Comput. Mach. 25;92-101. (A5.2)

Gonzalez, T. See also Sahni, S.

Goodman, S. See Cockayne, E.; Morrow, C.

184. Goyal, D. K. [1976), «Scheduling processor bound systems,» Report No. CS-76-036, Computer Science Department, Washington State University, Pullman, WA. (A5.2)

185. GRAHAM, R. L. [1966], «Bounds for certain multiprocessing anomalies,» Bell Syst. Tech. 3. 45, 1563-1581. (6.2)

186. GRAHAM, R, L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan [1978], «Optimization and approximation in deterministic sequencing and scheduling: a survey,» Ann. Discrete Math, (to appear). (A5.1)

GRAHAM, R. L. See also Chung, F. R. K.; Coffman, E. G., Jr; GAREY, M. R.; JOHNSON, D. S.

187. Grasselli, A., and F. Luccio [1966], «A method for the combined row-column reduction of flow tables,» Proc. 7th Ann. Symp. on Switching and Automata Theory, IEEE Computer Society, Long Beach, CA, 136-147. (A10.1)

188. Greibach, S. [1969], «Checking automata and one-way stack languages,» J. Comput. System Sci. 3, 196-217. (A10.1)

189. Greibach, S. A. [1973a], «Jump PDA's, deterministic context-free languages, principal AFDL's and polynomial time recognition-Extended abstract,» Proc. 5th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 20-28. (A10.2)

190. Greibach, S. A. [1973b], «The hardest context-free language,» SIAM J. Comput. 2, 304-310. (A10.2)

Greibach, S. See also Book, R. V.

191. Grimmet, G. R., and C. J. H. McDiarmid [1975], «On colouring random graphs,» Math. Proc. Cambridge Philos. Soc. 77., 313-324. (6.3)

Gross, D. A. See Fulkerson, D. R.

192. Gurari, E. M., and O. H. Ibarra [1978], «An NP-complete number theoretic problem,» Proc. 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 205-215. (A7.2)

193. Hack, M. [1972], «Analysis of production schemata'by Petri nets,» Report No. TR-94, Project MAC, Massachusetts Institute of Technology, Cambridge, MA. (A 12)

194. Hadlock, F. O. [1974], «Minimum spanning forests of bounded trees,» Proc. 5th Southeastern Conference on Combinatorics, Graph Theory, and Computing Utililas Malhematica Publishing, Winnipeg, 449-460. (A2.1; A2.2)

195. Hadlock, F. O. (1975], «Finding a maximum cut of a planar graph in polynomial time,» SIAM J. Comput. 4, 221-225. (4.1; AI.2; A2.2)

HaKEN, W. See APPEL, K.

Hakimi, S. L. See Kariv, O.

Hammer, P. L.'See Chvatal, V.

196. Harary, F. [1969], Graph Theory, Addison-Wesley, Reading, MA. (A1.2; A3.1, A13)

197. Harary, F. and R. A. Melter [1976], «On the metric dimension of a

graph,» ArsCombinatorical, 191-195. (A1.5)

198. Harary, F., and E. M. PalMer [1973], Graphical Enumeration, Academic Press, New York. (7.3; A2.1)

199. Harrison, M. A., W. L. Russo, and J. D. Ullman [1976], «Protection in operating systems,» Comm. ACM19, 461-471. (A4.3)

200. Hartmanis, J., and L. Berman [1978], «On polynomial time isomorphisms of some new complete sets,» J. Comput. System Sci. 16, 418-422. (7.1)

201. Hartmanis, J., and J. E. Hopcroft [1976], «Independence results in computer science,» SIGACT News 8:4, 13-24. (7.6)

202. Hartmanis, J., and H. B. Hunt, III [1974], «The LBA problem and its importance in the theory of computing,» in R. M. Karp (ed.). Complexity of Compulation, American Maihematical Society, Providence, RI, 1-26. (7.4)

203. Hartmanis, J., P. M. Lewis, and R. E. Stearns [1965], «Classification of computations by time and memory requirements,» Proc. IFIP Congress 1965, Spartan, New York, 31-35. (7.6)

204. Hartmanis, J., and R. E. Stearns [1965], «On the computational complexity of algorithms,» Trans, Amer. Math. Soc. 117, 285-306. (1.4; 7.6)

Hartmanis, J. See also Berman, L.

Hasegawa, T. See Ibaraki, T.

205. Havranek, T. [1975], «Statistical quantifiers in observational calculi: an application in GUHA methods,» Theory and Decision 6, 213-230. (A12)

Hecmt, M. S. See Frederickson, G. N.

Hedetniemi, S. T. See Cockayne, E.; Farley, A.; Mitchell, S.

206. Held, M., AND R. M. Karp [19711, «The traveling salesman problem and minimum spanning trees: part II,» Math. Programming 6, 62-88, (6.0)

Hell, P. See Kirkpatrick, D. G.

207. Herman, G. T., and G. Rozenberg 11975], Developmental Systems and Languages, North-Holland, Amsterdam. (A10.2)

208. Herrmann, P. P. [1973], «On reducibility among combinatorial problems,» Report No. TR-113, Project MAC, Massachusetts Institute of Technology, Cambridge, MA. (A1.1;A2.4)

209. Hirschberg, D., and M. Edelberg [1973], «On the complexity of computing graph isomorphism,» Report No. TR-130, Computer Science Lab., Dept.

of Electrical Engineering, Princeton University, Princeton, NJ. (A 13)

210. Hirschberg, D. S., and C. K. Wong [1976], «A polynomial-time algorithm for the knapsack problem with two variables,» J. Assoc. Comput. Mach. 23, 147-154. (A6)

211. Hopcroft, J. E. [1971], «An $n\, log_n$ algorithm for minimizing states in a finite automaton,» in Z. Kohavi and A. Paz (eds.), Theory of Machines and Computations, Academic Press, New York, 189-196. (A10.1)

212. Hopcroft, J. E., and R. M. Karp [1973,1, «An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs,» SIAM J. Comput. 2, 225-231. (3.1:2)

213. Hopcroft, J. E., and R. E. Tarjan [1974], «Efficient planarity testing,» J. Assoc, Comput. Mach. 21, 549-568. (4.1; A1.2; A13)

214. HOPCROFT, J. E., and J. D. Ullman [1967], «Nonerasing stack automata,» J. Comput. System Sci. 1, 166-186. (A10.1)

215. HOPCROFT, J. E., and J. D. Ullman [1969], Formal Languages and their Relation to Automata, Addison-Wesley, Reading, MA. (1.3; 2.2; 2.3; 7.4; 7,5; 7.6; A4.2; A10.1;A10.2)

216. HOPCROFT, J. E., and J. K. Wong [1974], «Linear lime algorithm for isomorphism of planar graphs (Preliminary report),» Proc. 6th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 172-184. (A13)

Hopcroft, J. E. See also Aiio, A. V.; Fortune, S.; Hartmanis, J.

217. Horn, W. A. [1972], «Single-machine job sequencing with treelike precedence ordering and linear delay penalties,» SIAM J. Appl. Math, 23, 189-202. (A5.1)

218. Horn, W. A. [1973], «Minimizing average flow lime with parallel machines,» Operations Res. 21,846-847. (A5.2)

219. Horn, W. A. [1974], «Some simple scheduling algorithms,» Naval Res. Logist. Quart. 21, 177-185. (A5.2)

220. Horowitz, E., and S. Sahni [1974], «Computing partitions with applications to the knapsack problem,» J. Assoc. Comput. Mach. 21, 277-292. (6.0)

221. HOROWITZ, E., and S. Sahni [1976], «Exact and approximate algorithms for scheduling nonidentical processors,» J. Assoc. Comput. Mach. 23, 317-327. (4.2.2; 6.1)

222. Horowitz, E., and S. Sahni [1978], Algorithms; Design and Analysis, Computer Science Press, Potomac, MD. (6.1)

223. Horvath, E. C, S. Lam, and R. Sethi [1977], «A level algorithm for preemptive scheduling,» J. Assoc. Comput. Mach. 24, 32-43. (A5.2)

224. Howell, T. D. [1977], «Grouping by swapping is NP-complete,» unpublished manuscript. (44.2)

225. Hu, X. C. [1961], «Parallel sequencing and assembly line problems,» Operations Res. 9, 841-848. (A5.2)

226. Hu, T. C. [1969], Integer Programming and Network Flows, Addison-Wesley, Reading, MA. (6.0)

227. Hu, T. C. [1974], «Optimum communication spanning trees,» SIAM J. Comput. 3, 188-195. (42.1)

HU, T. C. Sec also Adolphson, D.

228. Huet, G. P. [1973], «The undecidability of unification in third order logic,» Information and Control 11, 257-267. (A9.2)

229. HUGHES, G. E., and M. J. CRESSWELL [1968], An Introduction to Modal Logic, Methuen, London. (A9.2)

230. HUNT, H. B., III [1973a], On the Time and Tape Complexity of Languages, Doctoral Thesis, Dept. of Computer Science, Cornell University, Ithaca, NY. (A10.2)

231. Hunt, H. B., Ill [1973b], «On the time and tape complexity of languages I,» Proc. 5th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 10-19. (A 10.1; A 10.2)

232. HUNT, H.B., Ill [1976], «On the complexity of finite, pushdown, and stack autornata,» Math. Systems Theory, 10, 33-52. (A10.1)

233. Hunt, H. B., Ill [1977], «A complexity theory of computation structures: preliminary report,» unpublished manuscript. (A9.2; A11.2; A12)

234. Hunt, H. B. Ill [1978], «Uniform lower bounds on scheme equivalence,» unpublished manuscript. (A11.2)

235. Hunt, H. B., Ill, and D. J. Rosenkrantz [1977), «Complexity of grammatical similarity relations: preliminary report,» Proc. Conf. on Theoretical Computer Science, Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, 139-148. (A10.2;A13)

236. Hunt, H. B., Ill, and D. J. Rosenkrantz [1978], «Computational parallels between regular and context-free languages,» SIAM J. Comput. 7, 99-114. (A 10.2)

237. HUNT, H. B., Ill, D. J. Rosenkrantz, and T. G. Szymanski [1976a],

«On the equivalence, containment, and covering problems for the regular and context-free languages,» J. Comput. System Sci. 12, 222-268. (A10.2)

238. Hunt, H. B., Ill, D. J. Rosenkrantz, and T. G. Szymanski [1976b], «The covering problem for linear context-free grammars,» Theor. Comput. Sci. 2, 361-382 (A 10.2)

239. Hunt, H. B., Ill, and T. G. Szymanski [1976a], «Complexity metatheorems for context-free grammar problems,» J. Comput. System Sci. 13, 318-334. (A10.2)

240. Hunt, H. B., Ill, and T. G. Szymanski [1976b], «Dichotimization, reachability, and the forbidden subgraph problem (Extended abstract),1' Proc. 8th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 126-134: (A11.2)

241. Hunt, H. B., Ill, and T. G. Szymanski [1978], «Lower bounds and reductions between grammar problems,» J. Assoc. Comput. Mach. 25, 32-51. (A10.2)

242. Hunt, H. B., Ill, T. G. Szymanski, and J.D. Ullman [1975], «On the complexity of LR(k) testing,» Comm. ACM IS, 707-716. (A10.2)

Hunt, H. B., III. See also Constable, R. L.; Hartmanis, J.

Hwang, F. K. See Aho, A. V.

243. Hyafil, L., and R. L. Rivest [1973], «Graph partitioning and constructing optimal decision trees are polynomial complete problems,» Report No. 33, IRIA-Laboria, Rocquencourt, France. (A2.2)

244. Hyafil, L., and R. L. Rivest [1976], «Constructing optimal binary decision trees is NP-complete,» Information Processing Lett. 5, 15-17. (A12)

245. Ibaraki, T. [19781, «Approximate algorithms for the multiple-choice continuous knapsack problem,» unpublished manuscript. (A6)

246. Ibaraki, T., T. Hasegawa, K. Teranaka, and J. Iwase [1978], «The multiple-choice knapsack problem,» J. Oper. Res. Soc. Japan 21, 59-94. (A6)

247. Ibaraki, T., T. Kameda, and S. Toida [1977], «NP-complete diagnosis problems on systems graphs,» unpublished manuscript, (A3.1; A12)

Ibaraki, T. See also Kise.H.

248. Ibarra, O. H., and C. E. Kim [1975a], «Fast approximation algorithms for the knapsack and sum of subset problems,» J. Assoc. Comput. Mach. 22, 463-468, (6.1)

249. Ibarra, O. H., and C. E. Kim [1975b], «Scheduling for maximum profit,» Report No. 75-2, Computer Science Dept., University of Minnesota, Minneapolis,

MN. (A6)

250. Ibarra, O. H., and S. K. Sahni [1975], «Polynomially complete fault detection problems,» IEEE Trans. Computers C-24V 242-249. (A 12)

Ibarra, O. H. See also Gurari, E. M.

25I. ITAI, A. [1977], «Two commodity flow,» Report No. 93, Dept. of Computer Science, Technion, Haifa, Israel. (7.1; A2.4)

252. lTAi, A., Y. PERL, and Y. Shiloach [1977], «The complexity of finding maximum disjoint paths with length constraints,» Report No. 94, Dept. of Computer Science, Technion, Haifa, Israel. (A2.4)

253. ITAI, A., AND M. RoDEH [1977a], «Some matching problems,» in Automata, Languages, and Programming, Lecture Notes in Computer Science, Vol. 52, Springer, Berlin; 258-268. (A1.5)

254. ITAI, A., AND M. Rodeh [1977b], «Finding a minimum circuit in a graph,» Proc. 9th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 1-10. (A 2.3)

255. ITAI, A., M. Rodeh, and S. L. Tanimota [1978], «Some matching problems for bipartite graphs,» J. Assoc. Comput. Mach. (to appear). (A1.5)

Itai, A. See also Even, S.

Iwase, J. See Ibaraki, T.

256. Jackson, J. R. [1956], «An extension of JOHNSON's results on job lot scheduling,» Naval Res. Logist, Quart. 3, 201-203. (A5.3)

257. JAZAYER1, M., W. F. Ogden, and W. C. Rounds [1975], «The intrinsically exponential-complexity of the circularity problem for attribute grammars,» Comm. ACM 18, 697-706. (7.6)

258. JEROSLOW, R. G. [1973], «There cannot be any algorithm for integer programming with quadratic constraints,» Operations Res. 21, 221-224. (46)

259. Jeroslow, R. G. [1976], «Bracketing discrete problems by two problems of linear optimization,» Proc. First Symp. on Operations Research (at Heidelberg), Verlag Anton Hain, Meisenheim, 205-216. (A6)

260. JoHNSON, D. B., and S. D. Kashdan [1976], «Lower bounds for selection in $X+Y$ and other multisets,» Report No. 183, Computer Science Department, Pennsylvania State University, University Park, PA (to appear J. Assoc. Comput. Mach). (5.1; 7.3; A2.1; A2.3; A3.2)

261. JOHNSON, D. B., and T. Mizoguchi [1978], «Selecting the Kh element in $X+Y$ and $X_1+X_2+...+X_m$» SIAM J. Comput. 7, 147-153. (A3.2)

262. JOHNSON, D. S. [1973], Near-Optimal Bin Packing Algorithms, Doctoral Thesis, Dept. of Mathematics, Massachusetts Institute of Technology, Cambridge, MA. (6.1; 6.3)

263. JOHNSON, D. S. [1974a], «Approximation algorithms for combinatorial problems,» J. Comput. System Set. 9, 256-278. (6.1)

264. JOHNSON, D. S. [1974b], «Worst case behavior of graph coloring algorithms,» Proc. 5th Southeastern Conference on Combinatorics. Graph Theory, and Computing, Utilitas Mathematica Publishing, Winnipeg. 513-527. (6.1)

265. JOHNSON, D. S., A, Demers, J. D, Ullman, M. R. GAREY, and R. L GRAHAM [1974], «Worst-case performance bounds for simple one-dimensional packing algorithms,» SIAM J. Comput. 3, 299-325 (6.1)

266. JOHNSON, D. S., J. K. Lenstra, and A. H. G. Rinnooy Kan [1978], «The complexity of the network design problem,» Networks (to appear). (A2.1)

267. JOHNSON, D. S., and F. P. PrbparatA [1978], «The densest hemisphere problem.» Teor. Comput. Sci. 6, 93-107. (A6)

JOHNSON, D. S. See also Brucker, P.; Even, S.; Fraenkei, A. S.; GAREY, M. R.

JOHNSON, E. L. See Edmonds, J.

JOHNSON, S. C. See Aho, A. V.

268. JOHNSON, S. M. [1954], «Optimal two- and three-stage production schedules with setup times included,» Naval Res. Logist. Quart. 1, 61-68. (A5.3)

269. Jones, N. D. [1973], «Reducibility among combinatorial problems in log_n space,» Proc. 7th Ann. Princeton Conf. on Information Sciences and Systems, Dept. of Electrical Engineering, Princeton University, Princeton, NJ, 547-551. (7.5)

270. Jones, N. D». [1975], «Space-bounded reducibility among combinatorial problems,» J. Comput. System Sci. 11, 68-85. (7:5)

271. JONES, N. D., and W. T. LaasER [1976], «Complete problems for deterministic polynomial, time,» Theor. Comput. Sci. 3, 105-117. (7.5)

272. Jones, N. D., L. 11. LaNDWEBER, and Y. E. Lien [1977], «Complexity of some problems in Petri nets,» Theor. Comput. Sci. 4, 277-299. (A12)

273. Jones, N. D., Y. E. Lien, and W. T. Laasek [1976], «New problems complete for nondeterministic log space,» Math. Systems Theory 10, 1-17. (7.5)

274. Jones, N. D., and S.S. Muchnjk [1977], «Even simple programs are hard to analyze,» J. Assoc. Comput. Mach. 24, 338-350. (A11:2)

275. Jones, N. D., and S. Skyum [1976], «Complexity of some problems .concerning L systems (preliminary report), Report No. DAIMI PB-67, University of Aarhus, Aarhus, Denmark. (A10.2)

276. Jones, N.D., and S. Skyum [19.77], «Recognition of deterministic ETOL languages in logarithmic space,» Information and Control 35, 177-131; (A10.2)

Kameda, T. See Abdel-Wahab, II. M.; Ibaraki.T.

Kanellakis, P. C. See Papadimitriou, C. H.

277. KaRAGANIS, J. J. [I968J, «On the cube of a graph,» Canad. Math. Bull. 11, 295-296. (A1.3)

278. Kariv, O., and S. L, HAKIMJ [1976a], «An algorithmic approach to network location problems – Part I: the p-centers,» unpublished manuscript: (A2.5)

279. Kariv, O. and S. I. Hakimi [1976b], «An algorithmic approach to network location problems - Part 2: the p-medians,» unpublislicd manuscript. (5.2)

280. KARP, R. M; [1922], «Reducibllity among combinatorial problems,» in R. E. Miller and J. W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, 85-103. (1.5; 3.1; 5.2; 7.1; 7.4; A1.1; A1.2; A1.3; A2.1; A2.2; A3.1; A3.2; A5.1; A6; A10.1)

281. Karp, R. M. [1975a], «On the complexity of combinatorial problems,» Networks 5, 45-68. (A2.4)

282. Karp, R. M. [1975b], «The fast approximate solution of hard combinatorial problems,» Proc. 6th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Ulilitas Mathematica Publishing, Winnipeg, 15-31. (6.3)

283. KaRp, R. M. [1976], «The probabilistic analysis of some combinatorial search algorithms,» in J. F. Traub (ed.), Algorithms and Complexity: New Directions and Recent Results, Academic Press, New York, 1-19. (6.3)

284. Karp, R. M. [1977], «Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane,» Math. Oper. Res. 2, 209-224. (6.3)

285. Karp, R. M, A. C. McKellar, and C. K. Wong [1975], «Near-optimal solutions to a 2-dimensional placement problem,» SIAM J. Comput. 4, 271-286. (6.1)

Karp, R. M. See also, Edmonds, J.; Held, M.; Hopcroft, J. E.

Kasahi, T. See Araki, T.; Fujii, M.; Sugiyama, Y.

Kashdan, S. D. See JOHNSON, D. B.

286. Kaufman, M. T. [1974], «An almost-optimal algorithm for the assembly line scheduling problem,» IEEE Trans. Computers C-23, 1169-1174. (6.1)

287. Kernighan, B. W. [1971], «Optimal sequential partitions of graphs,» J. Assoc. Comput. Mach. 18, 34-40. (A2.2)

Kim, C. E. See Frederickson, G. N.; Ibarra, O. H,

288. KIRKPATRICK, D. G., and P. Hell [1978], «On the complexity of a generalized matching problem,» Proc. 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 240-245. (A1.1)

289. KISE, H.,T. Ibaraki and H. Mine [1978], «A solvable case of the one-machine scheduling problem with ready and due times,» Operations Res. 26, 121-126: (A5.1)

290. KLEE, V. [1978], private communication. (A6)

291. Klee, V., and G. J. MfNTY [1972], «How good is the simplex algorithm?,» in O. Shisha (ed.), Inequalities III, Academic Press, New York, 159-175. (1.3)

292. Kleene, S. C. [1956], «Representation of events in nerve nets,and finite automata» in C. E. Shannon and M. McCarthy (eds.), Automata Studies, Annals of Math. Studies, No. 34, Princeton University Press, Princeton, NJ, 3-41, (A10.1)

Klein, M. See Florian, M.

Klipker, I. A. See GolDberg, M. K.

293. KNUTH, D. E. [l973], private communication. (A5.1)

294. KNUTH, D. E. [1974a], «A terminological proposal,» SIGACT News 6:1, 12-18. (5.2)

295. KNUTH, D. E. [1974b], «Postscript about NP-hard problems,» SIGACT News 6:2, 15-16. (5.2)

296. KNUTH, D. E. [1974c], private communication. (A2.4) KNUTH, D. E. See also GAREY, M. R.

297. Kou, L. T. [1977], «Polynomial complete consecutive information retrieval problems,» SIAM J. Comput. 6, 67-75. (44.2)

298. Kou, L. T., L. J. Stockmeyer, and C. K. Wong 11978], «Covering edges by cliques with regard to keyword conflicts and intersection graphs,» Comm. ACM 2l, 135-138. (A1.1;A1.5)

299. Kozen, D. [1976], «Complexity of finitely presented algebras,» Report No. 76-294, Dept. of Computer Science, Cornell University, Ithaca, NY. (A7.3)

300. Kozen, D. [1977al, «Complexity of finilely presented algebras,» Proc. 9th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 164-177. (7.1; 7.5; A7.3; A13)

301. Kozen, D. [1977b], «Finitely presented algebras and the polynomial time hierarchy,» Report No. 77-303, Dept. of Computer Science, Cornell University, Ithaca, NY, (7.2;A7.3)

302. Kozen, D. [1977c], «First order predicate logic without negation is NP-complete,» Report No. 77-307, Dept. of Computer Science, Cornell University, Ithaca, NY. (A 9.2)

303. Kozen, D. [I977d], «Lower bounds for natural proof systems.» Proc. 18th Ann. Symp. on Foundations of Computer Science IEEE Computer Society, Long Beach, CA, 254-266. (A10.1; A12)

304. Kozen, D. [1978], «A clique problem equivalent to graph isomorphism,» unpublished manuscript. (7.1; A13)

305. Krishnamoorthy, M. S. [1975], «An NP-hard problem in bipartite graphs,» SIGACT News 7:1, 26. (A1.3)

306. Krishnamoorthy, M. S., and N. Deo. [1977a], «Node deletion NP-complete problems,» Technical Report, Computer Centre, Indian Institute of Technology, Kanpur, India. (A1.2)

307. KrisHnamoorthy, M. S., and N. Deo [1977b], «Complexity of the minimum dummy activities problem in a Pert network,» Technical Report, Computer Centre, Indian Institute of Technology, Kanpur, India. (A2.5)

306. Kruskal, J. B. [1956], «On the shortest spanning subtree of a graph and the traveling salesman problem,» Proc. Amer. Math. Soc. 7, 48-50. (6,1)

309. Kucera, L. [1976], «The complexity of clique finding algorithms,» unpublished manuscript. (6.2)

310. Kuroda, S. Y. [1964], «Classes of languages and linear-bounded automata,» Information and Control 7, 207-223. (A10.2)

Laaser, W. T. See Jones, N. D.

311. Labetoulle, J., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan [1978], «Preemptive scheduling of uniform machines,» unpublished manuscript. (A5.1; A5.2)

Labetoulle, J. See also Lawler, E. L.

312. Ladner, R. E. [1975a], «On the structure of polynomial lime reducibility,» J. Assoc. Comput. Mach. 22, 155-171. (7.1)

313. Ladner, R. E. [1975b], «The circuit value problem is log space complete for P,» SIGACT News 7:1, 18-20. (7.5).

314. Ladner, R. E, [1977], «The computational complexity of provability in systems of modal propositional logic,» SIAM J. Comput. 6, 467-480. (49.2)

315. Ladner, R. E., and N. Lynch [1976], «Refativization of questions about log space computability,» Math, Systems Theory 10, 19-32. (7.6)

316. Ladner, R. E., N. A. Lynch, and A. L. Selman [1975], «A comparison of polynomial time reducibilities,» Theor. Comput. Sci. I, 103-123. (7.1;7.2)

Ladner, R. E. See also Dobkin, D.

317. Laoeweg, B. J., E. L. Lawler, J. K. Lenstra, an.d A. H. G. Rinnooy Kan [1978], «Computer aided complexity classification of deterministic scheduling problems,» unpublished manuscript, Mathematisch Centrum, Amsterdam. (5.2)

318. Lageweo, B. J., and t K. Lenstra [1977], private communication. (A5.2)

319. Lageweg, B. J., J. K. Lenstra, and A. H. G. Rinnooy Kan [1976], «Minimizing maximum lateness on one machine: computational experience and some applications,» Stalistica Neerlandica 30, 25-41. (A5.1)

Lam, S. See Horvath, E. C.

Landweber, L. H. See JONES, N. D.

320. Langmaack, H. [1973], «On correct procedure parameter transmission in higher programming languages,» Acta Informat. 2, 110-142. (A 11.2)

321. LaPaugh, A. S., and R. L. Rivest [1978], «The subgraph homeomorphism problem,» Proc. 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 40-50. (A 13)

322. Lawler, E. L. [1972], «A procedure for computing the K best solutions to discrete optimization problems and its' application to the shortest path problem,» Management Sci. 18, 401-405. (5.1; 42.1; 43.2)

323, Lawler, E. L. [1973], «Optimal sequencing of a single machine subject to precedence constraints,» Management Sci. 19, 544-546. (A5.1)

324. Lawler, E. L. [1976a], Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York. (6.1; 4.1,1; A1.3; A2.3; A2.4; A3.1; A6; A8; A13)

325. Lawler, E. L. [1976b], «A note on the complexity of the chromatic

number problem,» Information Processing Lett. 5, 66-67. (6.0)

326. Lawler, E. L. [1976c], «Sequencing to minimize the weighted number of tardy jobs,» Rev, Francaise Automat, Informat. Recherche Operationnelle Ser. Bleue 10.5 (suppl.), 27-33. (A5.1)

327. Lawler, E. L. [1976d], private communication. (A 13)

328. Lawler, E. L. [1977a], «A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness,» Ann. Discrete Math. 1, 331-342. (4.2.2; 45.1)

329. Lawler, E. L. [1977b], «Fast approximation algorithms for knapsack problems,» Proc. 18th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society. Long Beach, CA, 206-213. (6.1)

330. Lawler, E. L. [1978], «Sequencing jobs lo minimize total weighted completion time subject to precedence constraints,» Ann. Discrete Math. 2, 75-90. (A5.I)

331. Lawler, E. L., AND J. Labetoulle [1978], «Preemptive scheduling of unrelated parallel processors,» J. Assoc. Comput. Mach. (to appear). (A5.2)

332.Lawler, E. L., and J. M. Moore [1969], «A functional equation and its applications to resource allocation and sequencing problems,» Management Sci. 16, 77-84. (4.2.2; A5.1)

lawler, E. L. See also Gonzalez, T.; GRAHAM, R. L.; Labetoulle, J.; Lageweg, B. J.

333. Leggett, E. W., Jr [1977], Tools and Techniques for Classifying NP-Hard Problems, Doctoral Thesis, Dept. of Computer and Information Sciences, Ohio Slate University, Columbus, OH. (7.2)

Lempel, A. See Even, S.

334. LENSTRA, J. K. [1977], private communication. (A5.1; A5.3)

335. LENSTRA, J. K., and A. H. G. Rinnooy Kan [1976], «On general routing problems,» Networks 6, 273-280. (A2.3)

336.Lenstra, J. K., and A. H. G. RiNNOOY KaN [1978a], «Complexity of scheduling under precedence constraints,» Operations Res. 26, 22-35. (6.2; A5.1; A5.2)

337. Lenstra, J. K., and A. H. G. Rinnooy Kan [1978b], «Computational complexity of discrete optimization problems,» Ann. Discrete Math, (to appear). (A5.3)

338. Lenstra, J. K., A. H. G. Rinnooy Kan, and P. Brucker 11977] «Complexity of machine scheduling problems,» Ann. Discrete Math. 1, 343-362. (A5.1; A5.2; A5.3)

339. LENSTRA, J. K., A. H. G. Rinnooy Kan, and M. Florian [1978], «Deterministic production planning: algorithms and complexity,» unpublished manuscript. (A5.4)

Lenstra, J. K. See also GRAHAM, R. L.; JOHNSON, D. S.; Labetoulle, J.;Lageweg, B. J.

340. LEVIN, L. A. [1973], «Universal sorting problems,» Problemy Peredaci Informacii 9, 115-116 (in Russian). English translation in Problems of Information Transmission 9, 265-266. (5.2; A1.4)

341. Lewis, H. R. [1978], «Satisfiability problems for prepositional calculi,» unpublished manuscript. (A9.1)

342. LEWIS, H. R., and C. H. Papadimitriou [1978], private communication. (A8)

342a. LEWIS, J. M. [1976], private communication. (A 1.2)

343. LEWis, J. M. [1978], «On the complexity of the maximum subgraph problem,» Proc. 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 265-274. (A1.2)

Lewis, P. M. See Hartmanis, J.; Rosenkrantz, D. J.

344. Lichtenstein, D. [1977], «Planar satisfiability and its uses,» SIAM J. Comput. (to appear). (A2.5;A9.1)

345. LICHTENSTEIN, P.; and M. Sipser [1978], «GO is Pspace hard,» Proc. 19th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 48-54. (7.4; A8)

Lichtenstein, D. I. See also Even, S.

346. LIEBERHERR, K. [1977], private communication. (A11.2)

Lien, Y. E, See Jones, N. D.

347. Lin, S. [1975], «Heuristic programming as an aid to network design,» Networks 5, 33-43. (6.0)

348. LipSHITZ, L. (1977], «A remark on the Diophantine problem for addition and divisibility,» unpublished manuscript. (A7.1)

349. Lipshitz, L. [1978], «The Diophantine problem for addition and divisibility,» Trans. Amer. Math. Soc. 235, 271-283. (A7.1)

350. Lipsky, W., Jr [1977a], «Two NP-complete problems related to information re-trieval,» in Fundamentals of Computation Theory, Lecture Notes in Computer Scl-ence, Springer, Berlin, (to appear). (A4.3)

351. LIPSKy, J., Jr [1977b], «One more polynomial complete consecutive retrieval problem,» Information Processing Lett. 6, 91-93. (A4.2)

352. LIPSKY, W., Jr [1978], private communication. (A4.2)

353.LIPTON, R. J. [1975], «The reachability problem requires exponential space,» Report No. 62, Dept. of Computer Science, Yale University, New Haven, CT. (A12)

354. Lipton, R. J., AND R. E. TARJAN [1977], «Applications of a planar separator theorem,» Proc. 18th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 162-170. (6.1; 6.2)

355. LIPTON, R. J., and Y. Zalcstein [1977], «Word problems solvable in logspace,» J. Assoc. Comput Mach. 24, 522-526. (7.5)

Lipton, R. J. See also Dobkin, D.

356. LiTVINTCHOUK, S. D., and V. R. Pratt f 1977], «A proof checker for dynamic logic,» Proc. 5th Internal. Joint Conf. on Artificial Intelligence, International Joint Conferences on Artificial Intelligence, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 552-558. (49.2)

357. Liu, C. L. [1968], Introduction to Combinatorial Mathematics, McGraw-Hill, New York. (6.1; A1.3)

358. Liu. P. C. and R. C. Geldmacher [1978], «On the deletion of nonplanar edges of a graph,» SIAM J. Comput. (to appear). (3.1.4; 3.2.2; A 1.2)

359. Lloyd, E. L. [1977], «On triangulations of a set of points in the plane,» Proc. 18th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 228-240. (A2.5; A13)

360. Lovasz, L. [1973], «Coverings and colorings of hypergraphs,» Proc, 4th Southeastern Conference on Combinatorics. Graph Theory, and Computing, Ulilitas Mathematica Publishing, Winnipeg, 3-12. (A3.1)

L. See also Burr, S.

361. LUCCHESI, C. I. 11976), A Minimax Equality for Directed Graphs, Doctoral Thesis, University Of Waterloo, Waterloo, Ontario. (A1.1)

362.Lucchesi, C. L. and S. I. OsBORN [1977], «Candidate keys for relations,» J. Comput. System Sci. (to appear. (A4.3)

Luccio, F, See Grasselli, A.

363. Luckham, D. C, D. M. Park, and M. S. Paterson [1970], «On formalised computer programs,» J. Comput. System Sci. 4, 220-249. (A11.2)

364. LUEKER, G. S. [1975], «Two NP-complete problems in nonnegalive integer programming,» Report No. 178, Computer Science Laboratory, Princeton University, Princeton, NJ. (A6)

Lueker, G. S. See also Booth, K. S.; Rose, D. J.

365. Lukes, J. A. [1974], «Efficient algorithm for lhe partitioning of trees,» IBM J. Res. Develop, 18,217-224. (A2.2)

366. Lynch, J. F, [1975], «The equivalence of theorem proving and the interconnection problem,» ACM SIGDA Newsletter 5:3. (A2.4)

367. Lynch, J. F. [1976], private communication. (A2.4)

368. Lynch, N. [1977], «Log space recognition and translation of parenthesis languages,» J. Assoc. Comput. Mach. 24, 583-590. (7.5)

369. Lynch, N. [1978], «Log space machines with multiple oracle tapes,» Theor. Comput. Sci. 6, 25-39. (7.6)

LYNCH, N. See also Ladner, R. E.

370. Maheshwari, S. [1976], «Traversal marker placement problems are NP-complete,» Report No. CU-CS-092-76, Dept. of Computer Science, University of Colorado, Boulder, CO. (A1.2; A1.5)

Maheshwari, S. N. See also Gabow, H. N.

371. Maier, D. [1977], «The complexity of some problems on subsequences and su-persequences,» J. Assoc. Comput. Mach. 25, 322-336. (A4.2)

372. Maier, D., and J. A. StoRER [1977], «A note on the complexity of the superstring problem,» Report No. 233, Computer Science Laboratory, Princeton University, Princeton, NJ. (A1.2; A4.2)

373. Manders, K., and L. Adleman [3978], «NP-complete decision problems for binary quadratics,» J. Comput. System Sci. 16, 168-184. (A7.1; A7.2)

Manders, K: See also Adleman, L.

374.Masek, W. J. [1978], «Some NP-complete set covering problems,» unpublished manuscript. (a4.2;A9,1)

375. Mathon, R. [1978], «A note on the graph isomorphism counting problem,» Information Processing Lett. (to appear). (A 13)

376. MatijaSevic, Y. V. [1970], «Enumerable sets are Diophantine,» Dokl. Akad. Nauk SSSR 191, 279-282 (in Russian). English translation in Soviet Math. Dokl. II, 354-357. (1.4)

377. Matijasevic, Y., and J. Robinson [17751, «Reduction of an arbitrary Diophantine equation to one in 13 unknowns,» Acta Arith. 27, 521-553. (A7.2)

Matula, D. W. See Edmonds, J.

Maxwell, W. L. See Conway, R. W.

378. McDiarmid, C. [1976], «Determining the chromatic number of a graph,» Report No. STAN-CS-76-576, Computer Science Dept., Stanford University, Stanford. CA. (7.6)

McDlARMlD, C. See also Grimmet, G. R.

McEliece, R. J. See Berlekamp, E. R.

McHugh, J. A. M. See BoESCH, F. T.

McKellar, A. C. See Karp, R. M.

379. McNaughton, R. [1959], «Scheduling with deadlines and loss functions,» Management Sci. 6, 1-12. (A5.2)

380. MeGIDDO, N. [1977], private communication. (A2.3)

Megiddo, N. See also Galil, Z. Melter, R. A. See Harary, F.

Merlin, P. M. See Chandra, A. K.

381. MEYER, A. R. [1975], «Weak monadic second order theory of successor ij> not elementary recursive,» in R. Parikh fed.), Logic Colloquium, (Proc. Symposium on Logic, Boston, 1972), Lecture Notes In Mathematics, Vol. 453, Springer, Berlin, 132-154. (7.6)

382. Meyer, A. R., and D. M. Ritchie [1967], «The complexity of loop programs,» Proc. 22nd Natl, Conf. of the ACM, Thompson Book Co., Washington, DC, 465-469. (A11.2)

383.MeVer, A. R., and M. I. Shamos [1977], «Time and Space,» in A. K. Jones fed.), Perspectives on Computer Science, Academic Press, New York, 125-146. (7.5)

384.Meyer, A, R., and L. J, Stockmeyer [1972], «The equivalence problem for regular expressions with squaring requires exponential time,» Proc. 13th Ann. Symp. on Switching and Automata Theory, IEEE Computer Society, Long Beach, CA, 125-129. (1.4; 7.2; 7.4; 7.6)

Meyer, A. R. See also Bauer, M.; Seiferas, J. I.; Stockmeyer, L. J.

Miller, G. A. See Chomsky, N.

385.MILLEK, G. L. [1976], «Riemann's Hypothesis and tests for primality,»

J. Comput. System Sci. 13,300-317. (7.1; A13)

386. Miller, G. L. [1977], «Graph isomorphism, general remarks,» Proc. 9th Ann, ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 143-150. (7.1; A13)

387. MlLLER, G. L. [1978], «On (he nlni» isomorphism technique: A preliminary report,» Proc. 10th Ann. ACM Symp. on Theory of Computing, Association fo: Computing Machinery, New York, 51-58. (A 13)

Miller, G. L. See also GAREY, M. R.

Miller, L. W. SeeCONWAY, R. W. Mink, H. See Kise, H.

388. Minsky, M. [1967], Computation: Finite and Infinite Machines, Prentice Hall, Engle-wood Cliffs, NX (2.2)

389. Minty, G. J. [1977], «On maximal independent sets of vertices in claw-fregraphs,» unpublished manuscript. (A1.2)

MiNTY, G. J. See also KLEE, V.

390. Mitchell, S., and S. Hedetniemi [1977], «Edge domination in frees,» Proc. 8th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Maifiematica Publishing, Winnipeg, 489-509 (A1.1)

Mitchell, S. See also Farley, A.

MlZOGUCHi, T. See Jownson, D. B.

391. Monma, C, L., and J., B. Sidney [1977], «A general algorithm for optimal job sequencing with series-parallel, precedence constraints,» Report No. 34.7, School of Operations Research, Cornell University, Ithaca, NY. (A5.1)

392. Moore, J. M. [1968], «Ann job, one machine sequencing algorithm for minimizing the number of late jobs,» Management Sci. 15, 102-109. (45.1)

Moore, J. M. See also Lawxer, E. L.

393. Morrow, C. and S. Goodman [1976], «An efficient algorithm for finding a longest cycle in a tournament,» Proc. 7th Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Mathematics Publishing, Winnipeg, 453-462. (A1.3; A2.3)

Mucmnik, S. S. See Jones, N. D.

MUNRO, I. See RoBERTSON, E.

394. Muntz, R. R., and E. G. CofFMAN, Jr [1969], «Optimal preemptive scheduling on two-processor systems,» IEEE Trans. Computers C-18, 1014-

1020. (A5.2)

395. Muntz, R. R., and E. G. Coffman, Jr [1970], «Preemptive scheduling of realtime tasks on multiprocessor systems,» J. Assoc. Comput. Mach. 17, 324-338. (A5.2)

396. MURTY, K. O. [1972], «A fundamental problem in linear inequalities with applications to the traveling salesman problem,» Math. Programming 2, 296-308. (A6)

397. Murty, K. G. [1976], Linear and Combinatorial Programming, John Wiley and Sons, Inc., New York. (A 13)

398. NelSon, G., and D. C. Oppen [1977], «Fast decision algorithms based on union and find,» Proc. 18th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 114-119. (A9.2)

399. Nemhauser, G. L., L. A. Wolsey, and M. L. FiSMER [1978J, «An analysis.of approximations for maximizing submodular set functions - I,» Math. Programming 14, 265-294. (6.1)

Nemhauser, G; L. See also Cornuejols, G., Garfinkel, R. S.

400. Nesetril, J., and A. Pultr [1977], «The complexity of a dimension of a graph,» Proc. Wroclaw Conf. on Foundations of Computer Science (to appear). (A 1.5)

401. Nesetril, J., and V. RoDL [1977], «A simple proof of Galvin-Ramsey properties of finite graphs and a dimension of a graph,» unpublished manuscript. (A1.5)

NieVerqelt, J, See Reingolp, E. M.

402. NiGMAlULLlN, R. G. [1975], «Complexity of the approximate solution of combinatorial problems,» Dokl Akad. .Nauk. SSSR 224, 289-292 (in Russian). English translation in Soviet Math- Dokl. 16, 1199-1203 (6.2)

Ninomiva, K., See Fujii, M.

Ogden, W. F.; See JazAYERi, M.

Okui. J. See Araki, t.; SugiyaMa, Y.

403. Opatrny, J. 11978], «Total ordering problem,» unpublished manuscript. (A12)

404. Opatrny, J., and K. Culik, II [1975], «Time complexity of L languages,» Abstracts of Papers: Conference on Formal Languages, Automata, and Development, University of Utrecht, Netherlands. (A10.2)

Oppen, D. C. See Nelson, G.

405. Orlin, J. [1976], «Contentment in graph theory: covering graphs with cliques,» unpublished manuscript. (A1.1)

Orlin, J. B. See also Bartholdi, J. J., III.

406. Orlova, G. I., and Y. G. Dorfman [19721, «Finding the maximum cut in a graph,» Engrg. Cybernetics 10, 502-506. (4.1; A1.2; A2.2)

Osborn, S. L. See LUCCHESI, C. L.

OSTERWEIL, L. See GABOW, H. N.

Palmer, E. M. See Harary, F.

407. Papadimitriou, C. H. [1976a], «The NP-completeness of the bandwidth minimization problem,» Computing 16, 263-270. (A1.3;A4.2)

408. Papadimitriou, C. H. [1976b], «On the complexity of edge traversing,» J. Assoc. Comput. Mach. 23,.544-554. (A2.3)

409. Papadimitriou, C. H. (1976c], «The complexity of the capacitated tree problem,» Report No. TR-21-76, Center for Research in Computing Technology, Harvard University, Cambridge, MA. (A2.1)

410. Papadimitriou, C. H. [1977], «The Euclidean traveling salesman problem is NP-complete,» Theor. Comput. Sci. 4, 237-244. (A2.3)

411. Papadimitriou, C. H. [1978a], «The adjacency relation on the traveling salesman, polytope is NP-complete» Math. Programming 14, 312-324. (A6)

412. Papadimitriou, C. H. [1978b], «Efficient search for rationals,» Report No. TR-01-78, Center for Research in.Computing Technology, Harvard University, Cambridge, MA. (A13)

413. Papadimitriou, C. H. [1978c], «Serializability of concurrent updates,» Report No, TR-14-78, Center for Research in Computing Technology, Harvard University, Cambridge, MA. (A4.3)

414. Papadimitriou, C. H. [1978d], private communication. (A1.1)

415. Papadimitriou,.C. H., P. A. Bernstein, and J. B. Rothnie [1977], «Some computational problems related to database concurrency control,» Proc. Conf. on Theoretical Computer Science, University of Waterloo, Waterloo, Ontario, 275-282. (A4.3)

416. Papadimitriou, C. H., and P. C. Kanellakis [1978], «Flowshop scheduling with limited temporary storage,» unpublished manuscript. (A5.3)

417. Papadimitriou, C. H., and K. Steiglitz [1976], «Some complexity results for the traveling salesman problem,» Proc. 8th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 1-9. (A 1.3)

418. Papadimitriou, C H., and M. Yannakakis [1978a], «On the complexity of minimum spanning tree problems,» unpublished manuscript. (A2.1)

419. Papapimitriou, C, II., and M. Yannakakis [1978b], «Scheduling interval-ordered tasks,» Report No. TR-11-78, Center for Research in Computing Technology, Harvard Universty, Cambridge, MA. (A5.2)

Papadimitriou, C. H. See also GAREY, M. R.; Lewis, H. R.

Park, D. M. See Luckuam, D, C.

420. Paterson, M. S, [1967J, Equivalence Problems in a Model of Computation, Doctoral Thesis, Cambridge University, Cambridge, England. (A11.2)

421. Paterson, M. S., and M, N, Wegman [1978], «Linear unification,» J. Comput. System Sci. 16, 158-167. (A7.3;A9.2)

Paterson, M. S.,See also Bauer, M.; LuckhaM, D. C.

422. Paul, W. J. [1977], «A 2.5n lower-bound on the combinational complexity of Boolean functions,» SIAM J. Comput. 6, 427-443, (7.6)

423. Paull, M, and S. Unger [1959], «Minimizing (he number of stales in incompletely specified sequential switching functions,» IRE Trans, Electron. Comput. EC-8, 356-367. (A10.1)

424. Perl, Y., and Y. Shiloach [1978], «Finding two disjoint paths between Iwo pairs of vertices in a graph,» J. Assoc. Comput. Mac], 25, 1-9. (A2.4)

425. PERL, Y., and S. Zaks [1978], private communication. (A 1.5)

Perl, Y. See also Itai, A.

426. Pfleeger, C. F. [1973], «Stale reduction in incompletely specified finite-slate machines,» IEEE Trans. Computers C-22, 1099-1102. (A10.1)

427. Pfleeger, C. F. [1974], Complete Sets and Time and Space Bounded Computation, Doctoral Thesis, Computer Science Dept., Pennsylvania State University, University Park, PA. (A10.1)

428. Pippenger, N., AND L. G. VaUANT [1976], «Shifting graphs and their applications,» J. Assoc. Comput. Mach, 23, 423-432. (7.6)

429. Plaisted, D. [1976], «Some polynomial and integer divisibility problems are NP-hard,» Proc. 17th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 264-267. (A3.1; A6; A7.1; A7.2; A7.3)

430. Plaisted, D. [1977a], «Sparse complex polynomials and polynomial reducibility,» J. Comput. System Sci. 14, 210-221. (A7.1; A7.2)

431: Plaisted, D. [1977b], «New NP-hard and NP-complete polynomial and integer divisibility problems,» Proc. 18th Ann. Symp. on Foundations of Computer Science IEEE Computer Society, Long Beach, CA, 241-253. (7.1; A7.1; A7.2)

432. Plesnik, J. [1978], «The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two,» unpublished manuscript. (A1.3)

Pnueli, A. See Even, S.

433. Poljak, S. [1974], «A note on stable seis and colorings of graphs,» Comment. Math. Univ. Carolinae 15, 307-309. (A1.2)

434. Pratt, V. [1975], «Every prime has a succinct certificate,» SIAM J. Comput. 4, 214-220. (7.1; A1.3)

435. Pratt, V. [1977], «Two easy theories whose combination is hard,» unpublished manuscript. (A9.2)

Pratt, V. See also Litvintchouk, S. D.

Preparata, F. P. See JOHNSON, D. S.

436. PROMEL, H. J. [1978], private communication. (A2.4)

Proskurowski, A. See Farley, A.

437. Pudlak, P. [1975], «Polynomial complete problems in the logic of automated discovery,» in Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Vol. 32, Springer, Berlin, 358-361. (A9.2)

438. Pudlak, P., and F. N. Springsteel [1975], «Complexity in mechanized hypothesis formation,» unpublished manuscript (A 12)

Pultr, A. See Nesetril, J.

439. QueyraNNe, M. [1976], private communication. (A 1.5)

440. Rabin, M. O. [1958], «Recursive unsolvability of group theoretic problems,» Ann. of Math. 67, 172-194. (1.4)

441. Rabin, M. O. [1976], «Probabilistic algorithms,» in J. F. Traub (ed.), Algorithms and Complexity: New Directions and Recent Results, Academic Press, New York, 21-39. (6.3)

Rabin, M. O. See also Fischer, M. J.

442. Rafsky, L. [1977], private communication. (A 12)

Rao, M. R. See Dantzig, G. B.

Ratliff, H. D. See Bartholdi, J. J., III.

443. REIF, J. H. [1978a], «A note on the complexity of imbedding extension problems,» unpublished manuscript. (A 13)

444. REIF, H. J. [1978b], «Polynomial time recognition of graphs of fixed genus,» unpublished manuscript. (A 13)

445. REINGOLD, E. M., J. NIEVERGELD, and N. Deo [1977], Combinatorial Algorithms: Theory and Practice, Prentice-Hall, Inc., Englewood Cliffs, NJ. (4.0)

446. Reiss, S. P. [1977a], Inverse Translation: the Theory of Practical Automatic Programming, Doctoral Thesis, Dept. of Computer Science, Yale University, New Haven, CT. (A 10.2)

447. Reiss, S. P. [1977b], «Statistical database confidentiality,» Report No. 25, Dept. of Statistics, University of Stockholm, Stockholm, Sweden. (A4.3)

448. REISS, S. P., and D. P. Dobkin [1976], «The complexity of linear programming, Report No. 69,» Dept. of Computer Science, Yale University, New Haven, CT. (7.1; A6; A13)

Reiss, S. P. See also Dobkin, D.

449. Reyner, S. W. [1977], «An analysis of a good algorithm for the subtree problem,» SIAM J. Comput. 6, 730-732. (4.2.2; A1.4)

Rinnooy Kan, A. H. G. See GRAHAM, R. L.; JOHNSON, D, S.; Labetoulle, J.: Lageweg* B. J.; Lenstra, J. K.

Ritchie, D. M. See Meyer, A. R.

Rivest, R. L See Hyafil, L.; LaPaugh, A. S,

450. Robertson, E. L. [1977], «Code generation.for,short/long address machines,» Report No. 1779, Mathematics Research Center, University of Wisconsin, Madison, WI. (A11.1)

451. Robertson, E. L. [1978], «Microcode bit optimization is.NP-complete,» IEEE Trans. Computers (to appear). (A11.1)

452. Robertson, E., and I. Munro 11978], «NP-completeness, puzzles, and games,» Utititas Math. 13, 99-116. (7.4; A8)

Robinson, J. See Matijasevic, Y. Rodeh, M. See Itai, A. Rodl, V. See Nesetril, J.

453. Rogers, H., Jr [1967], Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, (7.2)

454. Rose, D. J., and R. E. Tarjan [1978], «Algorithmic aspects of vertex elimination on directed graphs,» SIAM J. Appl. Math. 34, 176-197. (A1.3).

455. Rose, D., R.Tarjan, and G. Lueker [1976], «Algorithmic aspects of vertex elimination on graphs,» SIAM J. Comput. 5, 266-283. (A13)

456. Rosenkrantz, D. J, [1969], «Programmed grammars and classes of formal languages,» J. Assoc. Comput. Mach. 16, 107-131. (A10.2)

457. Rosenkrantz, D. J., R. E. Stearns, and P. M. Lewis [1977], «An analysis of several heuristics for the traveling salesman problem,» SIAM J. Comput. 6, 563-581. (6.1)

Rosenkrantz, D. J. See also Hunt, H. B., III.

458. RosENTHAL, A. [1974], Computing Reliability of Complex Systems, Doctoral Thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, CA. (A2.2)

459. Rosenthal, A. [1977], «Computing the reliability of a complex network,» SIAM J. Appl Math. 32, 384-393. (6.2)

ROTHNIE, J. B. See PAPADIMITRIOU, C. H.

460. Rounds, W. C. [19731, «Complexity of recognition in intermediate level languages,» Proc. 14th Ann. Symp. on Switching and Automata Theory, IEEE Computer Society, Long Beach, CA, 145-158. (A10.1; A10.2)

Rounds, W. C. See also Jazayeri, M.

Rozenberg, G. See Herman, G. I. Russo, W, L. See Harrison, M. A.

461. Rutledge, J. [1964], «On lanov's program schemata,» J. Assoc. Comput. Mach. 11, 1-9. (A11.2)

462. Sagiv, Y., and M. Yannakakis [1978], «Equivalence among relational expressions with the union and difference operations,» Report No. 241, Dept. of

Electrical Engineering and Computer Science, Princeton University, Princeton, NJ. (A4.3)

SAGIV, Y. See also Aho, A. V.

463. SAHNl, S. [1974], «Computationally related problems,» SIAM J. Comput. 3, 262-279. 0.2; A1.2; A2J; A6; A7J; A12)

464. Sahni, S. [1975], «Approximate algorithms for the u/i knapsack problem,» J. Assoc. Comput. Mach. 11, 115-124. (6.1)

465. SAHNi, S. [1976], «Algorithms for scheduling independent tasks,» J. Assoc. Comput. Mach, 23, 116-127. (4.2.2; 6.1))

466. Sahni, S. [1977], «General techniques for combinatorial approximation,» Operations Res. 25,920-936. (6.3).

467. SaHNI, S., and Y. Cho [1977a], «Scheduling independent tasks.with due times on a uniform processor system,» Report No. 77-7, Computer Science Dept., University of Minnesota, Minneapolis, MN. (A5.2)

468. Sahni, S., and Y. Cho [1977b], «Complexity of scheduling shops with no wait in process,» Report No. 77-20, Computer Science Dept., University of Minnesota, Minneapolis, MN. (A5.1)

469. Sahni, S., and T. Gonzalez [1976], «P-complete approximation problems,» J. Assoc. Comput. Mach. 23, 555-565. (6.2; A2.3)

Sahni, S. See also Cho, Y.; Constable, R. L.; Gonzalez, T.; Horowitz, E.; Ibarra, O. H.

470. Savitch, W. J. [1970], «Relationship between nondeterministic and deterministic tape complexities,» J. Comput. System Set. 4, 177-192. (7.4)

471. Savitch, W. J. [1974], «Nondeterministic log n space,» Proc. 8th Ann. Princeton Conf. on Information Sciences and Systems Dept. of Electrical Engineering, Princeton University, Princeton, NJ, 21-23. (7.5)

472. Schaefer, T. J. [1974], private communication. (A1.1)

473. Schaefer, T. J, [1978a], «Complexity of some two-person perfect-information games,» J. Comput. System Sci. 16, 185-225. (7.4; A8)

474. Schaefer, T. J. [1978b], «The complexity of satisfiability problems,». Proc. 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 216-226. (A1.1; A9.1; A9.2)

ScHAEfER,T. See also FRAENKEL, R. S.

Schmidt, E. M. See Fortune, S.

475. Seiferas, J. I., M. J. Fischer, and A. R. Meyer [1978], «Separating nondeterministic time complexity classes,» J. Assoc. Comput. Mach. 25; 146-167. (7.6)

Selman, A; L. See Baker, T. P.; Ladner, R. E.

476. SETHI, R. [1973], «A note on implementing parallel assignment instructions,» Information Processing Lett. 2, 91-95. (A11.1)

477. Sethi, R. [1975], «Complete register allocation-problems,» SIAM J. Comput. 4, 226-248. (3.2.3; A11.1)

478. SETHI, R. [1977a], «On the complexity of mean flow time scheduling,» Math. Oper. Res. 2, 320-330. (A5.2)

479. SETHI, R. [1977b], private communication, (A7.3)

48O. Sethi, R., and J. D. Ullman [19701, «The generation of optimal code for arithmetic expressions,» J. Assoc. Comput. Mach. 17, 715-728. (A11.1)

Sethi, r. See also Aho, A. v.; Bruno, J.; Comer, D.; Cook, S.; Downey, P. J.; GAREY, M. R.; Horvath, E. C.

481. Shamir, A. [1977], «Finding minimum cutsets in reducible graphs,» Report No. MIT/LCS/TM-85, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA. (A1.1)

Shamir, A. See also Even, S.

482. Shamir, E., and C. Beeri (1974], «Checking stacks and context-free programmed grammars accept P-complete languages,» in J. Loeckx (ed.), Proc. 2ndColloq. on Automata, Languages, and Programming, Lecture Notes in Computer Science, Vol. 14, Springer, Berlin, 27-33. (A10.1; A10.2)

483. SHAMOS, M. 1. [1976], «Geometry and statistics: problems at the interface.» in J. F. Traub (ed.). Algorithms and Complexity: New Directions and Recent Results, Academic Press, New York, 251-280. (A12)

Shamos, M. I. See also Meyer, A. R.

484. Shapiro, S. D. [1977], «Performance of heuristic bin packing algorithms with segments of random length,» Information and Control 35, 146-158. (6.3)

485. Shapley, L. S., and M. Shubik [1954], «A method of evaluating the distribution of power in a committee system,» Amer. Pol, Sci. Rev. 48, 787-792. (A 12)

486. Shiloach, Y. [1976), «A minimum linear arrangement algorithm for undirected trees,» Report, Dept. of Applied Mathematics, Weizmann Institute, Rehovot, Israel. (A1.3)

487. Shiloach, Y. (1978], «The two paths problem is polynomial,» Report No. STAN-CS-78-654, Computer Science Department, Stanford University, Stanford, CA. (A2.4;A13)

Shiloach, Y. See also Even, S.; Itai, A.; Perl, Y.

Shubik, M. See Shapley, L. S.

488. Sidney, J. B. [1973], «An extension of Moore's due date algorithm,» in S. E. EImaghraby (ed.), Symposium on the Theory of Scheduling and its Applications, Lecture Notes in Economics and Mathematical Systems, Vol. 86, Springer, Berlin, 393-398. (A5.1)

489. Sidney, J. B. [1975], «Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs,» Operations Res. 23, 283-298. (A5.1)

Sidney. J. B. See also Monma, C. L.

490. Simon, J. [1975], On Some Central Problems in Computational Complexity, Doctoral Thesis, Dept. of Computer Science, Cornell University, Ithaca, NY. (7.3)

491. Simon, J. [1977], «On the difference between the one and the many (preliminary version),» in Automata, Languages, and Programming, Lecture Notes in Computer Science, Vol. 52, Springer, Berlin, 480-491. (7.3)

492. Simons, B. [1978], «A fast algorithm for single processor scheduling,» Proc. 19th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long. Beach, CA, 246-252. (A5.1)

Simons, B. See also GAREY, M. R.

Sipser, M. See

Skyum, S. See JONES, N. D.

493. Slater, P. J. [1976], «fi-domination in graphs,» J. Assoc. Compul. Mach. 23, 446-450. (A2.5)

Slater, P. J. See also Cockayne, E.

494. Smith, W. E. [1956], «Various optimizers for single-slate production,» Naval Res. Logist. Quart. 3, 59-66. (A5.1)

Solovay, R. See Baker, T.

Springsteel, F. N. See Pudlak, P.

495. STATMAN, R. [1976], private communication. (A1.4; A9.1)

Stearns, R. E. See Hartmanis, J.; Rosenkrantz, D. J.

Steiglitz, K. See Papadimitriou, C. H.

496. STOCKMEYER, L. J. [19.73], «Planar 3-colorabilily is NP-comptete,» SiGACT News 5:3, 19-25. (3.2.3; 4.1)

497. STOCKMEYER, L. J. [1974a], The Complexity of Decision Problems in Automata Theory and Logic, Doctoral Thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA. (A 10.2)

498. Stockmeyer, L, J. [1974b], private communication. (A1.3)

499. STOCKMEYER, L. J. [1975], «The set basis problem is NP-complete,» Report No RC-543I, IBM Research Center, Yorktown Heights, NY. (A3.1)

500. Stockmeyer, L. J. [1976a], «The polynomial-time hierarchy,» Theor. Comput. Sci. 3, 1-22. (7.2; 7.4; 7.5; A7.3)

501. Stockmeyer, L. J. [1976b], private communication. (A4.1)

502. Stockmeyer, L. J., and A. K. Chandra [1978], «Provably difficult combinatorial games,» Report No. RC-6957, IBM Thomas J. Watson Research Center, York-town Heights, NY. (7.6)

503. StockMEYER, L. J., AND A. R. Meyer [1973], «Word problems requiring exponential lime,» Proc. 5th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 1-9. (7.4; 7.5; A7.1; A7.3; A8; A9.2; A10.2)

Stockmeyer, L. J, See also Chandra, A. K.; GAREY, M. R.; Kou, L. T.; Meyer, A. R.

504. STONE., H. S., and S. H. Fuller [1973], «On the near-optimality.of the shortest-latcncy-time-first drum scheduling discipline,» Comm. ACM 16, 352-353. (6.1)

505. Storer, J. A. [1977], «NP-completeness results concerning data compression,» Report No. 234, Dept. of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ. (A4.2)

506. Storer, J. A., and T.G. Szymanski [1978], «The macro model for data compression (Extended abstract),» Proc. 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 30-39. (A4.2)

Storer, J. A. See also Maier, D.

507. Sudborough, I. H. [1975], «A note on tape-bounded complexity classes and linear context-free languages,» J. Assoc. Comput. Mach. 22, 499-500. (7.5)

508.SUGIYAMA, Y., T. Araki, J. Okui, and T. Kasams [1977], «Complexity of the deadlock avoidance problem,» Trans. 1 ECE Japan 60-D, 251-258 (in Japanese) (45.3)

Sugiyama, Y. See also ARAKI, T.

509. Suurballe, J. W. [1975], «Minimal spanning trees subject to disjoint arc set constraints,» unpublished manuscript. (A2.1)

510. Syslo, M. M. [1973], «A new solvable case of the traveling salesman problem,» Math, Programming 4, 347-348. (A2.3)

511. Szymanski, T. G. [1978], «Assembling code for machines with span-dependent instructions,» Comm. ACM 21, 300-308. (3,2.2; A11.1)

Szymanski, T, G. See also Hunt, H, B., III; Storer, J. A,

Tanimota, S. L. See Itai, A.

512. TaRJAN, R. 6, [1977], «Finding optimum branchings,» Networks 7, 25-35. (A11)

513. Tarjan, R. E., and,A. E. Trojanowski [1977], «Finding a maximum independent set,» SIAM J. Comput. 6, 537-546. (6.0; 7.6)

Tarjan, R. E. See also Ehrlich, G.; Eswaren, K. P.; Even, S.; GAREY, M. R.; Hopcroft, J. E.; Lipton, R. J.; Rose, D. J,

Teranaka, K, See IBARAKi,T.

Thomassen, G. See Chvatal, V.

ToiDA, S. See IBARAKI, T.

514. TRAKHTENBROT, B. A., and V. M. Barzdin [1973], Finite Automata, North-Holland, Amsterdam. (A10.1)

Treybig, L. B. See Borosh, I.

TROJANOWSKI, A. E. See TARJAN, R. E.

515.Tseitin, G. S. .[1970], «On the complexity of derivaiion in propositional calculus,» in A. O. Slisenko (ed.), Studies in Constructive Mathematics and Mathematical Logic - Partly Consultants Bureau,- New York, 115-125. (1.6)

516. TsiCHRffzis, D. [19.70], «The-equivalence problem of simple programs,» J. Assoc. Comput. Mach.17, 729-738. (A11.2)

517. Turing, A. [1936], «On computable numbers, with an application to the Entscheidungsproblem,» Proc. London Math. Soc. Ser. 2 42, 230-265 and 43, 544-546. (1.4)

518. Ullman, J. D. [I975], «NP-compiete scheduling problems,» J. Comput. System Sci. 10, 384-393. (4.1; A5.2; A13)

519. ULLMAN, J. D. [1976], «Complexity of sequencing problems,» in E. G. Coffraati, Jr., (ed.), Computer and Job/Shop Scheduling Theory, John Wiley & Sons, New York, 139-164. (A5.2)

Ullman, J. D. See also AHo, A. V.; Harrison, M. A.; Hopcroft, J, E.; Hunt, H. B., HI; JOHNSON, D. S.; Sethi, R.

Unger, S. See Paull, M.

520. Valiant, L. G. [1976a], «Relative complexity of checking and evaluating,» Information Processing Lett. 5, 20-23. (5.1)

521. Valiant, L. G. [1976b], «A polynomial reduction of satisfiability to Ha mil Ionian circuits that preserves the number of solutions,» unpublished manuscript, (7.3)

522. Valiant, L, G. [1977a], «The complexity of computing the permanent,» Report No. CSR-14-77, Computer Science Department, University of Edinburgh, Edinburgh, Scotland, (to appear Theor. Comput. Sci.). (7.3; A1.1; A7.3)

523. Valiant, L. G. [1977b], «The complexify of enumeration and reliability problems,» Report No. CSR-15-77, Computer Science Dept., University of Edinburgh, Edinburgh, Scotland. (7.3; A2.2)

524. ValIant, L. G. [1977c], private communication. (A1.5; A7.2)

Valiant, L, G. See also Anqluin, D.; Pippenger, N.

525. VAN Leeuwen, J. [1975], «The membership question-for ETOL-languages is polynomially complete,» Information Processing Lett. 3, 138-143. (A10.2)

526.VAN Leeuwen, J. [1976a], «Having a Grundy-numbering is NP-complete,» Report No. 207, Computer Science Dept., Pennsylvania State University, University Park, PA. (A1.5)

527. VAN Leeuwen, J. [1976b], «Variations on a new machine model,» Proc. 17th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society, Long Beach, CA, 228-235. (7.5)

528. VAN Leeuwen, J. [1977], «Inequivalence of program-segments and NP-completeness,» Report No. 216, Computer Science Dept., Pennsylvania State University, University Park, PA. (A11.2)

529. van Sickle, L., and K. M. Chandy [1977], «The complexity of computer network design problems,» unpublished manuscript. (A4.1)

VAN TlLBORG, H. C. A. See BERLEKAMP, E. R.

530. Wagner, R. A. [1975], «On the complexity of the extended string-to-string correction problem,» Proc. 7th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 218-223. (A4.2)

53I. Wagner, R. A., and M. J. Fischer [1974], «The string-to-string correction problem,» J. Assoc. Comput. Mach. 21, 168-173. (A4.2)

532. Walsh, A. M., and W. A. Burkhard [1977], «Efficient algorithms for (3,1) graphs,» Information Sci. 13, 1-10. (A1.1)

Wegman, M. N. See Paterson, M. S.

Weinberg, L. See Bruno, J.

533. Winklmann, K. A. [1977], A Theoretical Study of Some Aspects of Parameter Passing In ALGOL-60 and in Similar Programming Languages, Doctoral Thesis, Purdue University, Lafayette, IN. (A11.2)

534. Witsenhausen, H. S. [1978], «Information aspects of stochastic control,» unpublished manuscript. (A3.2)

Wolsey, L. A. See Nemhauser, G. L.

535. Wong, C. K., and A. C. Yao [1976], «A combinatorial optimization problem related to data set allocation,» Rev. Francaise Automat. Informat. Recherche Operationelle Ser. Bleue 10.5 (suppl.), 83-95. (A3.2)

Wong, C. K. See also Hirschberg, D. S.; Karp, R. M.; Kou, L. T.

Wong, J. K. See Hopcroft, J. E.

536. Wrathall, C. [1976], «Complete sets and the polynomial-time hierarchy», Theor. Comput. Sci. 3, 23-33. (7.2; 7.4)

537. Yannakakis, M. [1978a], «The node deletion problem for hereditary properties,» Report No. TR-240, Computer Science Laboratory, Princeton University, Princeton, NJ. (41.2)

538. Yannakakis, M. [1978b], «Node- and edge-deletion NP-complete problems», Proc, 10th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York, 253-264. (A1.1; A1.2; A2.2)

539. Yannakakis, M. [1978c], private communication. (A1.2)

540. Yannakakis, M. and F. Gavril [1978], «Edge dominating sets in graphs», unpublished manuscript. (A1.1; A1.2; A12)

Yannakakis, M. See also Papadimitriou, C. H.; Sagiv, Y.

541. Yao, A.C. [1976], privale communication. (6.3)

542. Yao, A.C. [1978al, «New algorithms for bin packing,» Report.No. STAN-CS-781-662, Computer Science Department, Stanford University, Stanford, CA. (6.1)

543. Yao, A.C. [1978b], private communication. (A3.2)

Yao, A.C. See also GAREY, M. R.; Wong, C. K.

Yesha, Y. See Fraenkel, A. S.

544. Zadeh, N. [1973], «A bad network problem for the simplex method and other minimum cost flow algorithms,» Math. Programming 5, 255-266. (1.3)

Zaks, S. See Perl, Y.

Zalcstein, Y. See Upton, R. J.

В2. Работы, переведенные на русский язык или первоначально изданные в СССР

[8] *Ахо А., Хопкрофт Дж., Ульман Дж.* Временнáя и ленточная сложность языков магазинных автоматов. В сб. «Языки и автоматы». – М. Мир, 1975, 185-197.

[9] *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов.–М.: Мир, 1979.

[14] *Ахо А., Ульман Дж.* Теория синтаксического анализа, перевода и компиляции. Т. 1. Синтаксический анализ,–М.: Мир, [978.

[90] *Конвей Р., Максвелл В., Миллер Л.* Теория расписаний.–М.; Наука, 1975.

[91] *Кук С.А.* Сложность процедур вывода теорем. – Киб. сб., нов. сер., вып. 12. –М.: Мир, 1975, 5-15.

[101] *Дейт К.* Введение в системы баз данных.–М.; Наука, 1980.

[196] *Харари Ф.* Теория графов. – М.: Мир, 1973.

[204] *Хартманис Дж., Стирнз Р. О* вычислительной сложности алгоритмов. – Киб. сб., нов. сер., вып. 4. – М.: Мир, 1967, 57-85,

[211] *Хопкрофт Дж.Е.* – Киб. сб., Нов. сер., вып. П. –М.: Мир, 1974, 177-184.

[226] *Ху Т.* Целочисленное программирование и потоки в сетях. – М.: Мир, 1974.

[280] *Карп Р.М.* Сводимость комбинаторных задач. – Киб. сб.–М.: Мир, 1975, 16-38.

[340] *Левин А.А.* Универсальные задачи перебора. –Проблемы передачи информации, 9, № 3, 1973, 115-116.

[373] *Мандерс К., Эдлимэн Л.* NP-полные проблемы разрешения для квадратных уравнений с двумя неизвестными. – Киб. сб"нов. сер., вып. 17. – М: Мир, 1980.

[376] *Матиясевич Ю.В.* Диофантовость перечислимых множеств. – ДАН СССР, 191, 2, 1970, 279-282.

[388] *Минский М.* Вычисления и автоматы. – М.: Мир, 1971.

[402] *Нигматулин Р.Г.* О приближенных алгоритмах с ограниченной абсолютной погрешностью для дискретных экстремальных задач. – Кибернетика, № 1, 1978, 95-101.

[453] *Роджерс X.* Теория рекурсивных функций и эффективная вычислимость. – М.: Мир, 1972.

[514] *Трахтенброт Б.А., Бардзинь Я.М.* Конечные автоматы: поведение и синтез. – М.: Наука, 1976.

В3. Литература, дополненная авторами при переиздании

1*. Filotti, I.S., and J.N. Mayer [1980], «A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus,» Proc. 12th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York.

2*. Fortune, S., J.Hopcroft, AND J. Wyllje [1980], «The directed subgraph homeomorphism problem,» Theor. Comput. Sci. 10, 111-121.

3*. Gacs, P., and L. Lovasz [1979], «Khachian's algorithm for linear programming», Math. Prog. Studies (to appear).

4*. Hoffmann, C [1980], «Testing isomorphisms of cone graphs,» Proc. 12th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York.

5*. Holyer, I. private communication.

6*. Khachian, L.G. [1979], «A polynomial algorithm in linear programming», Dokl. Akad. Nauk. SSSR 244, 1093-1096 (in Russian). English translation in Soviet Math. Dokl. 20, 191-194.

7*. Lichtenstein, D. [1980], «Isomorphism for graphs embeddable on the projective plane,» Proc. 12th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York.

8*. Lovasz, L. [1978], «The matroid parity problem,» Proc. Conf. Algebraic Methods in Graph Theory (Szeged, 1978) (to appear).

9*. Luks, E.M. [1980], «A polynomial time algorithm for isomorphism of trivalent graphs « (to appear).

10*. Miller, G.L. [1980], «Isomorphism testing for graphs of bounded genus», Proc. 12th Ann. ACM Symp. on Theory of Computing, Association for Computing Machinery, New York.

11*. Seymour, P.D. [1979], «Decomposition of regular matrolds,» Report No. CORR 79-8, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario.

В4. Литература, дополненная при переводе на русский язык

[12*] *Адельсон-Вельский Г.М., Диниц Е.А., Карзанов А.В.* Потоковые алгоритмы. – М.: Наука, 1975.

[13*] *Бабат Л.Г.* Приближенное вычисление линейной функции на вершинах единичного n-мерного куба. В сб. «Исследования по дискретной оптимизации».–М.: Наука, 1975.

[14*] *Бабат Л.Г.* Линейные функции на n-мерном единичном кубе. ДАН СССР, т. 221, № 4, 1975.

[15*] *Гене Г.В., Левнер Е.В.* Эффективные приближенные алгоритмы для комбинаторных задач. Препринт. – М: ЦЭМИ АН СССР, 1981.

[16*] *Гене Г.В., Левнер Е.В., Фрумкин М.А.* О вычислительной сложности экстремальных комбинаторных задач. – Советско-польский научный семинар по математическим методам в планировании и управлении экономикой. Сборник тезисов. –М.: ЦЭМИ АН СССР, 1979.

[17*] *Диниц Е.А., Карзанов А.В.* Об экспоненциальной сложности алгоритмов решения общей и транспортной задач линейного программирования. В сб. «Управление сложными системами». – М.: ИАТ, 1974, 32-41.

[18*] Исследования по дискретной математике. – М.: Наука, 1973.

[19*] Исследования по дискретной оптимизации. – М: Наука, 1976.

[20*] *Кнут Д.* Искусство программирования для ЭВМ. – Т. 1. Основные алгоритмы.–М.: Мир, 1976; Т. 2, Получисленные алгоритмы, М., Мир, 1977; Т. 3. Сортировка и поиск. – М.: Мир, 1978.

[21*] Комбинаторные методы в потоковых задачах. ВНИИСИ, сб. трудов, вып. 3. – М.: 1979.

[22*] *Кристофидес Н.* Теория графов. Алгоритмический подход.– М.: Мир, 1978.

[23*] *Кузнецов О.П., Адельсон-Вельский Г.М.* Дискретная математика для инженера. – М.: Энергия, 1980.

[24*] *Леонтьев В.К.* Дискретные экстремальные задачи 25.–Итоги науки и техники, сер. Теория вероятностей, матем. статистики, теория кибернетики, т. 16.–М.: 1979, ВНИИТИ, 39-102.

[25*] *Лившиц Э.М., Рублинецкий В.И.* О сравнительной сложности некоторых задач дискретной оптимизации. В сб. «Вычислительная математика и вычислительная техника». Вып. 3. – Харьков, 1972, 78-85.

[26*] *Мальцев А.И.* Алгоритмы и рекурсивные функции. – М.: Физматгиз, 1965.

[27*] *Матиясевич Ю.В.* Диофантовы множества. – Успехи матем. наук, т. XXVI, вып. 5 (167), 1972 (185-222).

[28*] *Пахомов С.В.* Машинно-независимое описание некоторых машинных классов сложности.–Записки науч. семинаров ЛОМИ, т. 88, 176-185.

[29*] *Прим Р.К.* Кратчайшие связывающие сети и некоторые обобщения,– Киб. сб., вып. 2, 1961. 95-107.

[30*] *Тарьян Р.Э.* Сложность комбинаторных алгоритмов. – Киб. сб., нов. сер., вып. 17.

[31*] *Фридман А.А., Фрумкин М.А., Хмелевский Ю.И., Левнер Е.В.* Исследование эффективных алгоритмов для дискретных и комбинаторных задач. Теории сводимости, универсальные задачи.–М.: ЦЭМИ АН СССР, 1976.

[32*] *Фрумкин М.А.* Сложность дискретных задач. –М.: ЦЭМИ АН СССР, 1981.

[33*] *Хачиян Л.Т.* Полиномиальный алгоритм в линейном программировании. ДАН СССР, т. 244, вып. 5, 1979.

[34*] *Шоломов Л.А.* Основы теории дискретных логических и вычислительных устройств. – М.: Наука, 1980.

[35*] *Grötschel M., Lovasz L., Schrijver A.* The ellipsoid method and its consequences in combinatorial optimisation. *Institut für Ökonometric and Operation Res, Rheinische Friedrich – Wilchelms - Universität. Behn, 1980. Report N80151 – OR.*

B5. Литература, дополненная при изготовлении элькопии

[01] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. – М.: МЦНМО, 1999 и последующие пара стереотипных изданий. 960 с., 263 илл.

NP-полноте в данной книге посвящена отдельная глава (n 36, со с. 837 по с. 879) со следующими разделами:

Полиномиальное время (с. 838)
Проверка принадлежности языку и класс *NP* (с. 845)
NP-полнота и сводимость (с. 849)
Доказательства *NP*-полноты (с. 857)
NP-полные задачи (с. 864).

[02] *Джон Хопкрофт, Раджив Мотвани, Джеффри Ульман.* Введение в теорию автоматов, языков и вычислений. – М.: Вильямс, 2002 (пер. 2-го изд.) 528 с., с илл.

NP-полноте в данной книге посвящена две последние главы (n 10-11, со с. 423 по с. 523) со следующими разделами:

Глава 10. Труднорешаемые проблемы

Классы *P* и *NP* (с. 423)