РОССИЙСКАЯ АКАДЕМИЯ НАУК
Сибирское отделение
Институт математики

На правах рукописи

СЕВАСТЬЯНОВ Сергей Васильевич

УДК 519.854

ГЕОМЕТРИЧЕСКИЕ МЕТОДЫ И ЭФФЕКТИВНЫЕ АЛГОРИТМЫ В ТЕОРИИ РАСПИСАНИЙ

01.01.09 — математическая кибернетика

Диссертация на соискание ученой степени
доктора физико-математических наук

Новосибирск — 2000

Оглавление

Моим любимым женщинам
Посвящаю этот многострадальный труд

Неформальное введение

Глава “Неформальное введение” является факультативной. Она не содержит строгих определений и утверждений и не является обязательной для понимания последующих доказательств теорем. Однако если прочтение этой главы поможет читателю лучше понять суть полученных в диссертации результатов, то ее предназначение можно будет считать выполненным.

1 “Откуда есть пошла . . .” Исторический экскурс

(или благодарности всем тем, без кого эта книга не была бы написана)

По-видимому, в написанной диссертации было бы неправильно не упомянуть (и не поблагодарить — хотя бы с большим опозданием) тех, без кого этой диссертации бы просто не было.

Хотя история не терпит сослагательного наклонения, но вполне возможно, что этой диссертации бы не было на свете, если бы моя школьная учительница (и замечательный педагог) Ксения Тимофеевна Богомазова не подбросила мне в далеком 1965 году “Комсомолку”, в которой на полном развороте были напечатаны задачи Всесоюзной заочной олимпиады по математике (а также по физике и химии), и где красочно описывалось, как в далеком заснеженном Академгородке по улицам мимо беленьких коттеджей бродят академики и физматшкольники и решают задачи. С тех пор целью моей жизни стало попасть в Академгородок, — я заболел им. Кстати, знакомству со своей первой “любимой женщиной” — геометрией — я также обязан Ксении Тимофеевне.

Позже в ФМШ мне посчастливилось попасть на спецкурс В.В. Войтишека по “Неевклидовым геометриям”, который он читал по книге Гильберта и Кон-Фоссена “Наглядная геометрия”. Когда я остался последним слушателем этого спецкурса, Вац.-Вац. просто отдал мне эту книгу “в долгосрочную аренду”, и я целый год с удовольствием читал ее. (Конспект этой книги у меня хранится до сих пор.) Чтобы оценить поступок Вацлава Вацлавовича, нужно знать, что в то время эта книга была огромной библиографической редкостью.

Как-то так случилось, что на первых курсах университета мое увлечение геометрией на время уснуло. На первое место вышли теория графов и теория расписаний. Спецкурс по теории расписаний читал В.А.Перепелица, и ему я обязан знакомству со своей второй "любимой женщиной" (то бишь, теорией расписаний)[1]. У Виталия Афанасьевича был (и я думаю, до сих пор сохранился) несомненный педагогический талант, умение слушать и заинтересовать слушателя, способность вдохновить молодого человека на самостоятельные исследования[2] ... Но ближе к делу.

Зимой 1973 года я безуспешно пытался обобщить алгоритм Джонсона (который так здорово справлялся с двухстаночной задачей) на случай трех станков. В то время еще не имели понятия об "NP-трудных" задачах (это понятие пришло чуть позже), и было совершенно не понятно "упрямство" этой задачи — почему не удается построить эффективный алгоритм для трех станков[3]? Построение алгоритмов приближенного решения тогда еще считалось неклассическим. — На худой конец, можно было решить задачу и приближенно, но это был результат "второго сорта". Это направление еще только начинало развиваться, благодаря усилиям как наших (Боровков, 1962 [8], Гимади, Глебов, Перепелица [27, 14, 15, 11, 12]), так и зарубежных математиков (Graham, 1966 [88])[4]. Одним из "начинателей" этого направления был Виталий Афанасьевич Перепелица. Перед отправкой на летние каникулы, — чтобы я, знать, не болтался всё лето зря, — он предложил мне две задачки (на выбор), чтобы я попробовал их решить как-нибудь приближенно.

Одной из них была задача Объемно Календарного Планирования (ОКП), другой — задача Джонсона для трех станков (попытки точного решения которой оказались безуспешными). Итак, — две совершенно разные задачи. В первой нам задан годовой план предприятия, состоящий из n наименований. Каждое наименование характеризуется m-мерным вектором. (В качестве компонент этого вектора могут выступать временны́е затраты на выпуск этого изделия по разным типам станков, потребности в других видах ресурсов, стоимость изделия, и т.п.) Требуется разбить годовой план равномерно по l периодам (например, четырем кварталам или двенадцати месяцам). Причем, равномерность требуется по каждой из m характеристик.

Очевидная трудность при решении этой задачи состояла в том, что если мы попытаемся добиться равномерности по одной из компонент (не обращая внимания на остальные), затем по другой, и т.д., то в итоге ничего хорошего не получится уже

[1]Кстати, знакомству со своей будущей женой Асей — в то время студенткой 5-го курса НГУ и крупным экспертом по БАМу — я также обязан Виталию Афанасьевичу. Немного позже к моим любимым женщинам добавились дочурки Женька и Ксенька. Им всем я очень благодарен за понимание и поддержку.

[2]Возможно, обладай этими педагогическими талантами в той же мере лектор по теории графов, одним "графистом" было бы больше, а одним "расписальщиком" — меньше. Так или иначе, я выбрал теорию расписаний.

[3]Если быть точным, то это непонятно до сих пор.

[4]В монографии (Hochbaum, 1997 [63]), посвященной построению аппроксимационных алгоритмов, Лэсли Холл пишет, что по-видимому работа Грэма является первым опытом построения приближенного алгоритма с гарантированной оценкой точности. — К сожалению, мы уже привыкли, что работы российских математиков остаются "незамеченными" на Западе.

при $m \geq 3$, — в том смысле, что не удастся получить хороших априорных оценок равномерности, справедливых в "худшем случае" (т.е. для всевозможных входов данной задачи).

Во второй задаче — задаче Джонсона — каждая из n работ состоит из m операций, выполняемых на разных станках. (Имеется m станков различного типа.) Существенным элементом постановки задачи является заданная *технология* выполнения каждой работы, т.е. заданная последовательность выполнения ее операций. Требуется найти расписание (т.е. определить, когда начинать и заканчивать каждую операцию), обеспечивающее выполнение всех операций в кратчайший срок. Необходимым требованием к расписанию является недопустимость одновременного выполнения любых двух операций, выполняемых на одном и том же станке.

При анализе первой задачи (ОКП) естественным образом возникла следующая геометрическая задача. Пусть $x_i \in \mathbb{R}^m$ — характеристический вектор i-го изделия $(i = 1, \ldots, n)$, $x = \sum x_i$ — суммарный характеристический вектор. Задача состоит в нахождении такого разбиения множества индексов $\{1, 2, \ldots, n\}$ на l частей $N_1, \ldots, N_l$, чтобы в каждой части N_i суммарный вектор $\sum_{j \in N_i} x_j$ мало отличался от вектора x/l. В частности, этого можно добиться, если найти такой порядок суммирования векторов $\{x_j\}$, чтобы полученная траектория частичных сумм проходила вблизи от узлов $x/l, 2x/l, \ldots, (l-1)x/l$. Если спроецировать все векторы $\{x_j\}$ на гиперплоскость, ортогональную вектору x, то сумма проекций $\{x'_j\}$ будет равна нулю. А если предположить, что исходные векторы x_j ограничены сверху по норме, то их проекции x'_j также будут ограничены по норме. В результате приходим к следующей задаче.

В нормированном пространстве $\mathbb{R}^{m-1}$ задано n "коротких" векторов (не более чем единичной длины каждый). Сумма векторов равна нулю. Требуется найти такой порядок суммирования векторов, чтобы все частичные суммы находились в шаре как можно меньшего радиуса.

Для евклидовой плоскости удалось показать, что всякое такое семейство векторов может быть просуммировано внутри шара радиуса $\sqrt{3}$. Причем искомая последовательность суммирования векторов находится эффективно, за время $O(n \log n)$. Это позволило построить эффективный алгоритм приближенного решения задачи ОКП с гарантированной оценкой точности в случае $m = 3$, — чего до сих пор никому не удавалось.

Так прошли те летние каникулы. Когда я уже сидел на вокзале в ожидании поезда в Новосибирск, нечаянно пришла в голову мысль: а нельзя ли применить задачу о суммировании векторов к задаче Джонсона? Минут через 5 я к своему удивлению понял — применяется!! Используя мой алгоритм суммирования векторов на плоскости в шаре радиуса $\sqrt{3}$, можно эффективно строить приближенные расписания, длина которых отличается от оптимума не более чем на константное число длин максимальной операции (и таким образом, не зависит от числа работ)[5].

Так я открыл для себя задачу о *компактном суммировании векторов* (КСВ)[6]. Поз-

[5]После этого я постоянно твержу своим студентам: "Как хорошо держать в голове сразу несколько разных задач!"

[6]Зимой 1973/74 года удалось построить другой алгоритм суммирования векторов — уже в простран-

же выяснилось, что **в то же самое время** независимо от меня связь между задачей Джонсона и задачей КСВ была обнаружена харьковскими математиками Беловым и Столиным [6]. Не удивительно ли? В том, что такая связь была обнаружена кем-то еще, нет ничего удивительного — это лишь свидетельствует, что данная связь является "внутренней сущностью" рассматриваемой задачи[7]. Удивительна синхронность. — 20 лет (с момента появления статьи Джонсона) в теории расписаний не было ничего такого, и вдруг появилось одновременно в двух разных местах. Правда, для вывода своих оценок Белов и Столин использовали другое нормированное пространство, но сути дела это не меняло. Когда же они спросили В.А. Перепелицу: "Каким доказательством **леммы Штейница** пользовался Ваш студент?", — я сделал для себя второе открытие. Оказывается, на заре нашего столетия известный математик Э. Штейниц[8] доказал лемму [131] о том, что в пространстве $\mathbb{R}^m$ существует такая константа C, что всякое семейство векторов в $\mathbb{R}^m$ не более чем единичной длины и с нулевой суммой может быть просуммировано в шаре радиуса C. Эту лемму Штейниц использовал для получения обобщения теоремы Римана об условно сходящихся рядах[9].

К моменту знакомства с результатами харьковчан я уже умел суммировать "короткие" векторы в m-мерном евклидовом пространстве в шаре радиуса $2^m/\sqrt{3}$ [31], поэтому не сильно удивился, когда узнал, что при решении задачи Джонсона Белов и Столин пользовались этой же оценкой радиуса (выведенной в 1953 году харьковским математиком М.И. Кадецом [20], семинар которого в харьковском университете посещали Белов и Столин). Гораздо большее удивление вызвал факт (обнаруженный мной при "библиографических раскопках"), что еще в 1931 году эта же оценка была выведена В. Бергстрёмом [77]. (Само по себе это не плохо — в качестве упражнения самостоятельно получить какой-то известный результат. Но ведь сколько лишней бумаги тратится, чтобы результаты этих "упражнений" публиковать!)[10]

После получения этих результатов возник естественный вопрос: куда двигаться дальше? С одной стороны, меня не очень устраивала экспоненциальная зависимость радиуса суммирования от размерности пространства. (Ведь с точки зрения приложе-

---

стве произвольной размерности m. (Доказать возможность суммирования векторов в шаре, радиус которого зависит от m и не зависит от числа векторов.) Благодаря этому эффективно строится приближенное расписание в задаче Джонсона уже для произвольного числа машин. Оценка погрешности этого расписания не зависит от числа работ.

[7] В 1982 году эта связь была "переоткрыта" венгерскими математиками Bárány и Fiala [74].

[8] Хотя правильнее произносить эту фамилию как Штайниц. Но отдавая дань русской традиции, будем писать "Штейниц". Впрочем, иногда ее даже пишут как "Стейниц".

[9] Известная теорема Римана о числовых рядах гласит, что если числовой ряд $\sum_{i=1}^{\infty} a_i$ сходится к числу a и существует перестановка $(\pi_1, \pi_2, \ldots)$ натуральных чисел $\{1, 2, \ldots\}$, для которой числовой ряд $\sum_{i=1}^{\infty} a_{\pi_i}$ сходится к числу a', отличному от a, то **вся числовая прямая** (включая ∞) является областью сумм для числового ряда $\{a_1, a_2, \ldots\}$. Штейниц обобщил теорему Римана на случай векторных рядов в пространстве $\mathbb{R}^m$, доказав, что область сумм векторного ряда является линейным многообразием.

[10] Всё это наводит на мысль, что ныне действующая система хранения и распространения научной информации очень неэффективна. — Трудно получить быструю справку о существующих результатах на интересующую тему. Уже назрела (и вероятно, в скором времени грядет) революция в этой сфере, одной из черт которой должна стать интернационализация (обобществление) науки. Признаки этого процесса уже просматриваются.

ний этой задачи — задачи ОКП и задачи Джонсона — чем меньший радиус удастся получить в задаче КСВ, тем более точные решения удастся гарантировать для исходных задач.) Кроме того, не устраивала трудоемкость ($O(n^m)$) алгоритма суммирования векторов[11]. С другой стороны, можно было попытаться применить задачу КСВ к более сложным моделям построения расписаний, допускающим наличие не одного технологического маршрута (прохождения деталей по станкам), а нескольких. Работы Белова с соавторами (см. [3, 4, 5]) являли прецеденты таких применений.

В первом направлении прогресс наступил осенью 1977 г., когда удалось показать, что радиус m достаточен для суммирования векторов в m-мерном пространстве, причем не только для евклидовой, но для любой симметричной нормы. Доказательство этого факта [32] было конструктивным, но оценка трудоемкости алгоритма ($O(n^2 2^m)$) не была вполне полиномиальной. (Полностью полиномиальный алгоритм, трудоемкости $O(n^2 m^2)$, появился немного позже и был опубликован в 1980 году [35].) Факт существования линейной оценки радиуса суммирования в задаче КСВ, по-видимому, произвел сильное впечатление на профессора М.И. Кадеца, и он пригласил никому не известного молодого человека из Новосибирска (то бишь, меня) выступить на своем семинаре по функ. анализу в харьковском университете. Такое выступление состоялось весной 1978 года. Доказательство оценки m было достаточно громоздким, и поскольку Кадеца в большей степени интересовал лишь сам факт линейности радиуса, то доказательство оценки m я рассказал полностью лишь на семинаре И.С. Белова и В.С. Гринберга[12], а для выступления на семинаре Кадеца я подготовил простое доказательство более слабой оценки ($2m$). Вскоре по возвращении в Новосибирск я получил письмо от В.С. Гринберга, в котором он писал, что в моем длинном доказательстве нет нужды, т.к. алгоритм, используемый мною в коротком доказательстве, также дает оценку m[13].

Существенного продвижения удалось добиться и на втором направлении — в направлении применения компактного суммирования векторов к другим задачам построения расписаний. Что касается задачи Джонсона, то там роль задачи КСВ выглядела достаточно простой, поскольку при построении алгоритма приближенного решения задачи Джонсона мы ограничили свой выбор множеством так называемых "перестановочных" расписаний, в которых на каждом станке детали обрабатываются в одном и том же порядке (т.е. согласно одной и той же перестановке индексов $\{1, \ldots, n\}$). Для нахождения этой перестановки каждой детали $j = 1, \ldots, n$ ставится в соответствие $(m-1)$-мерный вектор x_j, для полученного семейства векторов решается задача КСВ (т.е. находится

[11] В [31] я показал, что трудоемкость можно понизить до $O(n \log n)$ за счет увеличения радиуса суммирования в константу раз.

[12] Сегодня, с высоты своего 25-летнего научного опыта я могу утверждать, что тот семинар был уникальным. Редко у нас встречаются семинары, когда слушатели имеют твердое намерение полностью разобраться в излагаемом материале, не оставляя непонятой даже малейшей детали.

[13] Так появилась наша совместная с Гринбергом статья в журнале "Функциональный анализ и его приложения" [16]. В ней на полутора страницах неконструктивно доказывается, что оценка радиуса суммирования m справедлива для любой, не обязательно симметричной, нормы, причем для несимметричной нормы эта оценка не может быть улучшена. Любопытно, что при существующей сегодня системе подсчета рейтинга институтов СО РАН за эту статью, — до сих пор наиболее цитируемую из всех моих статей, — мне присудили бы "баранку", как за всякую статью размером меньше 3-х страниц.

перестановка индексов $\pi = (\pi_1, \ldots, \pi_n)$, согласно которой следует суммировать векторы $\{x_j\}$) и затем найденная перестановка π берется в качестве искомого порядка обработки деталей на каждом станке. Однако эта простая схема соответствия между порядком обработки деталей и порядком суммирования соответствующих векторов рушится, как только мы начинаем рассматривать модели построения расписаний с более сложными технологическими маршрутами прохождения деталей по станкам. Сразу становится неясным, чему соответствует порядок суммирования векторов при построении расписания.

Где-то в 1982 году обнаружилось[14], что такое соответствие все же удается реализовать в расписании, если использовать дополнительный "трюк": сдвиг каждой операции (в перестановке операций данной машины, исходно согласованной с порядком суммирования векторов) вправо на фиксированное число позиций, зависящее от "уровня" операции, где уровень операции есть ее порядковый номер в данной работе. Таким образом, все операции первого уровня остаются на своих местах, все операции второго уровня сдвигаются относительно исходного положения на одно и то же число позиций ν_2, все 3-и операции сдвигаются на ν_3 (где $\nu_3 > \nu_2$), и т.д. Сдвиг осуществляется путем добавления (в начало перестановки) фиктивных операций фиксированной длины. Более подробно с алгоритмом построения расписания можно ознакомиться в главе 4. Сейчас лишь заметим, что основное предназначение перестановки, находимой посредством компактного суммирования векторов, состоит в обеспечении равномерности возрастания нагрузки по разным машинам при добавлении новых работ.

Найденный подход позволял строить расписания для довольно общих задач: не только для задачи job shop (которая сама по себе является достаточно общей, поскольку допускает наличие произвольных технологических маршрутов деталей по станкам), но и для ее обобщений, таких как задача dag shop[15], где порядок операций каждой работы не обязан быть цепью, а может быть произвольным частичным порядком. Длина такого приближенного расписания, как и в случае простейшей задачи Джонсона, отличается от тривиальной нижней оценки оптимума ($L_{\max}$) на величину, не зависящую от числа работ и пропорциональную максимальной длительности операции ($p_{\max}$). При этом коэффициент перед $p_{\max}$ выражается полиномом 4-й степени от величин m (число машин) и r (максимальное число операций работы), а трудоемкость алгоритма оценивается полиномом $O(n^2m^2r^2)$ [44][16].

Таким образом, помимо практического результата (построение алгоритма приближенного решения с гарантированной оценкой точности) был получен теоретический результат о свойствах оптимальных решений задачи dag shop, а именно, обнаружен *интервал локализации оптимумов* всех примеров данной задачи при каждом значении параметров m и r, причем длина этого интервала при выборе $p_{\max}$ в качестве единицы

[14] Вероятно, момент прозрения был достаточно ярким, поскольку он помнится до сих пор. Я сидел в читалке Института математики, и во все окна ярко светило солнце.

[15] Аббревиатура "dag" означает, вероятно, *directed acyclic graph*. Она использовалась, в частности, Шмойсом, Стайном, Вайном (1991) [130]. Первоисточник этой аббревиатуры автору не известен.

[16] В первой версии этого алгоритма, опубликованной в ДАН СССР в 1984 году [43], точность решения оценивалась полиномом 5-й степени, при худшей оценке трудоемкости.

измерения времени и при фиксированных m и r является константой. В этой ситуации естественно возникает вопрос о нахождении **минимально возможного** интервала локализации оптимумов. Ясно, что для более узких классов примеров такие минимальные интервалы должны быть более узкими, чем для самой общей задачи. (Например, для задачи Джонсона длина такого интервала сейчас оценивается сверху полиномом 2-й степени от m, вместо полинома 4-й степени — для более общей задачи job shop.) Однако есть подозрение (гипотеза), что эта разница в длине интервалов всё же не должна быть настолько большой (т.е. не должна отличаться на два порядка, как это есть на настоящий момент). А следовательно, будущим молодым исследователям задачи dag shop предстоит уменьшить степень полинома, оценивающего длину интервала локализации оптимумов этой задачи.

Для подтверждения этой гипотезы исследовались различные специальные случаи задачи dag shop, и для них делались довольно успешные попытки уточнения интервала локализации оптимумов. Так например, для задачи *встречных маршрутов*, так же как и для (одномаршрутной) задачи Джонсона, длина интервала оценивается сверху полиномом 2-й степени [40][17]. Как и в остальных случаях, оценка длины интервала доказывается конструктивно, т.е. строится (эффективный!) алгоритм нахождения приближенного решения задачи со значением целевой функции из заданного интервала. На основе алгоритма из [40] Душин получил результат [18], согласно которому интервал локализации оптимумов всякой ациклической двухмаршрутной задачи[18] также оценивается полиномом второй степени от m. Позже он распространил этот результат на случай произвольного фиксированного числа маршрутов [19].

Наибольшего успеха в уточнении интервала локализации оптимумов удалось достичь для различных частных случаев задачи job shop при малом числе машин. Так например, для всех трех неизоморфных типов ациклических двухмаршрутных задач с тремя машинами (которые в главе 4 обозначаются как R231, R213 и R321) были найдены значительно уточненные (по сравнению с Душинскими) интервалы локализации оптимумов. Все три интервала оказались, как ни странно, различными[19]. При этом минимальность найденного интервала удалось доказать лишь для задачи R321 (так обозначается трехмашинная задача встречных маршрутов). В разделе 40 доказывается, что этот же интервал является минимальным для трехмашинной задачи Джонсона.

Следует отметить, что нахождение упомянутых выше минимальных (как и "почти минимальных") интервалов локализации оптимумов различных задач, рассматриваемых в главах 2, 3 и 4, было бы невозможно без применения новой техники суммирования векторов, которую я называю "нестрогим суммированием векторов" (или коротко, НСВ). Суть этой техники заключается в том, что ищется решение не упомянутой

[17] Этот результат вошел в кандидатскую диссертацию автора.

[18] т.е. задачи, в которой допускается наличие двух различных маршрутов деталей по станкам, являющихся перестановками машин

[19] а именно, $[L_{\max}, L_{\max}+5p_{\max}]$, $[L_{\max}, L_{\max}+4p_{\max}]$ и $[L_{\max}, L_{\max}+3p_{\max}]$ соответственно. Возможно, что в этом различии нет ничего странного, поскольку для построения приближенных расписаний в этих задачах использовались различные алгоритмы. К сожалению, из полученных результатов мы не можем пока сделать вывод о "различной природе" трех родственных задач, поскольку первый и второй из этих трех интервалов, возможно, не являются минимальными.

выше задачи о суммировании векторов в шаре минимального радиуса, а некоторой другой (задачи НСВ), причем каждой расписательской задаче соответствует своя задача НСВ, зависящая от набора возможных маршрутов деталей по станкам. В каждой такой задаче НСВ ищется оптимальное (согласно некоторому критерию) семейство полупространств. При этом маршруты деталей задают ограничения на выбор этих полупространств, а "нестрогость" суммирования понимается в следующем смысле. Если при "строгом" суммировании ищется минимальная (в каком-то смысле) область, внутри которой при подходящем порядке суммирования находятся ВСЕ частичные суммы, то при нестрогом суммировании достаточно, чтобы, грубо говоря, "каждая вторая" частичная сумма находилась в заданной области. А точнее, на любом шаге допускается выход частичной суммы за пределы означенной области, если на следующем шаге она возвращается в данную область.

* * *

Таким образом, мы очертили основное направление, по которому развивались исследования автора в теории расписаний, а именно: методы суммирования векторов в нормированном пространстве и их применение в задачах построения расписаний. Другой важной составляющей этих исследований является отыскание широких полиномиально разрешимых классов примеров NP-трудных задач. Наиболее "податливой" в этом отношении оказалась задача open shop, для которой (как следует из результатов главы 1) множество неподдающихся эффективному решению примеров составляет лишь незначительную долю от множества всех примеров. Большинство же примеров образуют *эффективно нормальные классы*, допускающие, во-первых, эффективные алгоритмы их решения, и во-вторых, обладающие свойством *нормальности* (когда длина оптимального расписания совпадает с тривиальной нижней оценкой — величиной нагрузки наиболее загруженной машины). При этом для построения оптимальных расписаний используются как алгоритмы, основанные на методах суммирования векторов, так и простые жадные алгоритмы. Отдельной строкой стоит оригинальный *метод Фиалы*, несколько усовершенствованный автором диссертации.

Однажды поразившая автора диссертации идея близости традиционно далеких друг от друга областей математики, идея "родственности" возникающих в этих областях задач требовала новых и новых подтверждений. Это нашло своё отражение в исследованиях, результаты которых приведены в первой и второй частях диссертации. Так например, при построении нижней оценки функции Штейница (т.е. значения минимально возможного радиуса шара, внутри которого может быть просуммировано любое семейство "коротких" векторов с нулевой суммой) для норм $l_p,\ p \geq 1$, мною использовался "плохой" пример семейства векторов, который строился на ортогональном базисе пространства. Но для нормы l_∞ такая нижняя оценка, основанная на данном примере, вырождалась в 1 (т.е. в тривиальную оценку). Для нахождения нетривиальной оценки для этой нормы (и топологически близких к ней норм) требовалось плохой пример строить на другом ортогональном базисе в $\mathbb{R}^m$, — таком, чтобы все базисные векторы имели все компоненты, равные ± 1. И такие базисы предоставляют нам матрицы Адамара. Следовательно, от того, для какого максимального $m' \leq m+1$ нам удастся

построить матрицу Адамара порядка m', зависит значение нижней оценки функции Штейница для нормы l_∞. Таким образом, актуальными для нас становятся исследования, направленные на подтверждение гипотезы Адамара[20]. Во второй части диссертации представлен целый "клубок" взаимосвязанных задач из различных разделов дискретной математики.

Наконец, автор отдал дань и традиционным направлениям исследований, таким как построение алгоритмов приближенного решения с относительными оценками точности — для одних задач (результатом такого типа является представленная в разделе 35 PTAS для многопроцессорной задачи open shop с фиксированным числом машин) и доказательство "неприближаемости" — для других задач (см. раздел 36). Из этого кома разношерстных, но тем не менее тяготеющих друг к другу результатов и возникла данная диссертация.

2 О названии

В идеале, название должно определять область исследований, включающую **всё** содержимое диссертации, и в то же время, исключающую всё, что в эту диссертацию не входит. Однако очевидно, что если ограничить длину названия несколькими словами, то задача отыскания идеального названия становится невыполнимой. (В противном случае автор вынужден признать, что 280 страниц текста, потраченных на изложение содержания диссертации, являются избыточными.) Таким образом, всякое название будет так или иначе нарушать предъявленные выше требования. В частности, из оглавления диссертации легко заметить, что во-первых, предложенное автором название **не включает** всего содержимого диссертации. (Например, не все из рассматриваемых здесь задач принадлежат области теории расписаний.) И во-вторых, что название **не исключает** многих аспектов исследования, которые могли бы быть в диссертации (исходя из ее названия), но которых в ней нет. Например, в диссертации рассматриваются лишь **многостадийные** обслуживающие системы и задачи построения оптимальных расписаний для таких систем лишь по одному из возможных критериев, а именно — по критерию "минимум длины расписания". (Хотя этот критерий является наиболее часто употребимым, он не является единственно возможным.) Ясно также, что в диссертации описываются **не все** известные геометрические методы, применяемые в теории расписаний, а лишь те, разработкой которых занимается автор диссертации. Список таких "недостатков" названия можно было бы продолжить.

Чем же оправдан выбор тех 6 слов (не считая предлогов и союзов), которые фигурируют на титульном листе в качестве названия диссертации? Как представляется автору, это название описывает некоторое подмножество результатов, составляющих **ядро** диссертации. Остальные результаты так или иначе примыкают к ядру либо непосредственно, либо через другие результаты.

[20] о том, что при $m \geq 4$ матрица Адамара порядка m существует если и только если m кратно 4

3 Область исследования

Основными областями исследования в диссертации являются теория расписаний и комбинаторная геометрия. Что такое теория расписаний? Дать адекватное и исчерпывающее определение этой области столь же трудно, как и другим современным областям математики, таким как “кибернетика” и “информатика”. Вместо этого попытаемся дать ее неформальное описание.

Прежде всего, во всех моделях теории расписаний присутствует такой фактор как **время**. Как правило, в задаче теории расписаний уже известен ответ на классический вопрос “Что делать?”, но не известен ответ на вопрос “Как делать?”. Говоря точнее, известен заранее набор заданий — *работ*, которые требуется выполнить, известен набор исполнителей этих работ, которых принято называть *машинами* (хотя в реальных задачах это могут быть, например, живые люди или группы людей)[21], и задан набор технологических ограничений, описывающих множество допустимых способов выполнения исходного множества работ. В понятие “способ выполнения работ” могут входить различные аспекты: во-первых, определение набора используемых ресурсов и их количество (возможно, изменяющееся во времени); во-вторых, выбор исполнителя работы[22]; и в-третьих, определение *расписания* выполнения работ, т.е. моментов времени, когда начинает выполняться каждая работа или какая-то ее часть (*операция*), в какие моменты происходят ее прерывания и последующие возобновления. Зачастую определение способа выполнения работ сводится лишь к нахождению расписания. (Причем, если прерывания работ запрещены, то достаточно указать лишь момент начала каждой операции.) Для краткости, всякий способ выполнения работ будем называть “расписанием”[23].

Наконец, в любой модели теории расписаний, как и вообще в моделях дискретной оптимизации, как правило присутствует *целевая функция*, позволяющая оценивать качество того или иного допустимого расписания. В *задаче* теории расписаний требуется либо найти *оптимальное расписание* (на котором достигается минимум или максимум целевой функции — в зависимости от *типа* задачи), либо “построить” **произвольное допустимое** расписание, либо, наконец, доказать, что такого допустимого расписания не существует.

Название второй области исследования — “комбинаторная геометрия” — выбрано автором достаточно условно, в силу необходимости как-то обозначить ту область, в которой рассматриваются дискретные наборы геометрических объектов (таких как семейства m-мерных векторов, семейства вершин m-мерного куба, и т.п.), и требуется путем перебора конечного числа вариантов выбрать подходящее решение той или иной

[21] Часто вместо базисных понятий “работы” и “машины” используются термины “детали” и “станки”. Хотя в монографии [61] для этой цели придуманы более изящные термины: “требования” и “приборы”, — но в русскоязычной “расписательской среде” последние как-то пока не прижились.

[22] В некоторых моделях исполнителей удобно рассматривать как еще один тип ресурса. В других моделях они рассматриваются отдельно и получают названия *машин*.

[23] Автор не оригинален в такой “обобщенной” трактовке понятия *расписание*. Те, кто придумал название “теория расписаний”, также не имели в виду лишь способ “расписывания” заданий во времени.

комбинаторной задачи.

4 Объект и цели исследования

Далее мы будем различать понятия *индивидуальной задачи* (т.е. задачи, в которой все входные данные полностью определены) и *массовой задачи* (представляющей собой бесконечное семейство индивидуальных задач). Для краткости, массовую задачу будем называть просто "задачей" (например, будем рассматривать "задачу Джонсона"), а ее индивидуальную задачу будем чаще всего называть "примером"[24]. Термин *вход* задачи оставим за совокупностью входных данных конкретного примера[25]. Зафиксировав те или иные параметры задачи (например, число работ или число машин), мы получим *подзадачу*, или *класс примеров* исходной задачи.

Решение каждого конкретного примера не представляет теоретической трудности, поскольку множество его допустимых расписаний либо конечно, либо мы можем показать (в результате анализа свойств оптимального решения), что достаточно ограничиться рассмотрением некоторого конечного множества расписаний[26]. Таким образом, решение каждого конкретного примера может быть найдено за **константное время** (где "время" является синонимом числа элементарных вычислительных операций), а потому не представляет теоретического интереса. Гораздо интереснее с теоретической точки зрения находить универсальные способы решения **любых** примеров исходной задачи, или хотя бы — примеров из некоторых семейств бесконечной мощности.

Здесь мы подходим к понятию *алгоритма*, как **основной цели** решения той или иной дискретной оптимизационной задачи. (Таким образом, нам становится неинтересным само искомое расписание для какого-то конкретного примера. Нас теперь интересует **способ нахождения** такого расписания для любого примера исходной задачи, или **алгоритм** решения задачи.) *Алгоритм* — это набор предписаний, которым мы должны руководствоваться при решении задачи. Первым естественным требованием к алгоритму является его **конечность**, т.е. гарантированная остановка (с положительным или отрицательным результатом) через конечное время. Вторым требованием является **безусловная гарантия нахождения** искомого решения, если таковое существует[27].

[24] В выборе этого термина автор не оригинален, поскольку во всем мире специалисты по исследованию операций уже давно пользуются термином "instance".

[25] Если задача определена, то всякий ее пример однозначно определяется своим входом. Однако понятие "примера" не сводится целиком к безликому "входу", поскольку помимо этого включает в себя: а) способ интерпретации входных данных; б) критерий оптимальности данной задачи.

[26] Например, при нахождении расписаний, допускающих прерывания работ, и в условиях целочисленности всех исходных данных, как правило, удается доказать, что достаточно ограничиться рассмотрением расписаний, в которых все прерывания работ происходят в целочисленные моменты времени. — Отсюда вытекает конечность множества рассматриваемых нами расписаний. Однако часто такое свойство расписаний принимается без доказательства, в качестве "очевидного". К несчастью, существуют достаточно простые примеры, на которых это "очевидное" свойство нарушается. (Простейший такой пример — из известных автору — содержит всего три работы, выполняемые на трех машинах. Две работы выполняются по технологии flow shop, а одна — по технологии open shop. Оптимальное расписание этого примера имеет нецелую длину.) Рассмотрение задач с прерываниями, однако, выходит за рамки данной работы.

Однако и этого оказывается недостаточно. Среди всех алгоритмов, удовлетворяющих двум указанным выше требованиям, мы стремимся отыскать **наиболее быстрые**. Таким образом, какая бы ни была целевая функция в рассматриваемой задаче, помимо ее оптимизации во всех случаях мы стремимся также минимизировать время, затрачиваемое на решение задачи[28]. Это подводит нас к одному из основных понятий дискретной оптимизации — понятию *эффективного алгоритма*. Алгоритм решения задачи будем называть *эффективным*, если он безусловно гарантирует нахождение искомого решения, — если таковое существует, — и гарантирует остановку вычислений через время, не превышающее **полинома** от длины записи исходной информации.

В приведенном выше определении используются такие не определенные нами понятия как "время" и "длина записи исходной информации". На мой взгляд, в теории сложности (как она излагается Гэри и Джонсоном [17]) нет точного математического определения этих понятий (что, возможно, и препятствует доказательству главной гипотезы века о P≠NP). Мы также ограничимся полуинтуитивным объяснением этих понятий, отослав читателя для ознакомления с "более строгим" их описанием к монографии Гэри и Джонсона [17]. Предварительно сделаем одно замечание лингвистического характера.

Одновременно с каким-либо алгоритмом (т.е. набором предписаний) будем рассматривать некую абстрактную вычислительную машину, способную выполнять фиксированный набор элементарных действий согласно заданному алгоритму. Это позволит нам отождествлять алгоритм (как набор предписаний) и вычислительную машину, "реализующую данный алгоритм". В результате мы сможем употреблять ставшие привычными для слуха выражения типа "алгоритм находит решение..." или "на вход алгоритма подается множество..." Ясно, что если вместо слова "алгоритм" подставить в эти выражения "набор предписаний", то получится абракабадра. Если же подставить словосочетание "вычислительная машина", то всё становится на свои места. Именно такую подстановку мы будем мысленно осуществлять, употребляя (для краткости, и отдавая дань традиции) выражения типа "алгоритм находит".

Итак, под "временем работы алгоритма", или *трудоемкостью алгоритма* будем понимать число элементарных действий, которые должна выполнить абстрактная вычислительная машина в процессе решения задачи согласно данному алгоритму. Однако необходимо уточнить, какие действия мы считаем "элементарными". Часто в литературе под "элементарными действиями" понимаются арифметические (типа сложения,

[27] Мы не рассматриваем здесь *вероятностных* алгоритмов, которые гарантируют успех либо "почти всегда", либо с определенной долей вероятности.

[28] Чтобы понять, почему время выдвигается в ряд основных критериев оценки качества алгоритма, достаточно убедиться, что для одной и той же задачи различные алгоритмы могут настолько сильно различаться по времени работы, что какие-то из них становятся очевидно неприемлемыми. Рассмотрим задачу минимизации функционала на перестановках из n элементов и два алгоритма ее решения: один алгоритм перебирает все возможные перестановки (число которых равно $n!$), а другой находит решение за $n \log_2 n$ элементарных действий. Тогда при $n = 1000$ и быстродействии компьютера 10^{13} операций в секунду (скорость наиболее быстрого из современных компьютеров) второй алгоритм позволяет найти решение за одну десятимиллиардную долю секунды, тогда как первый алгоритм обрекает нас на "абсолютно вечное" вычисление — вечность, превышающую возраст вселенной в 10^{2532} раз ...

умножения) и логические операции (типа сравнения). Считается, что для оценки числа по-настоящему элементарных (битовых) операций достаточно число арифметических операций умножить на квадрат длины записи максимального из исходных чисел. К сожалению, такой подход, хотя и упрощает подсчет трудоемкости, не всегда является правильным (например, если в алгоритме участвует такая элементарная операция как умножение). Дело в том, что для строгого доказательства некоторых утверждений мы должны гарантировать, что вычисления ведутся **абсолютно точно**, т.е. сохраняются все значащие цифры во всех промежуточных вычислениях. Но тогда для записи результатов промежуточных вычислений мы не можем ограничиться каким-то заранее фиксированным значением максимальной длины записи чисел, т.к. при каждом умножении двух чисел эта величина может удвоиться. Итоговый объем памяти, потребный для записи промежуточных чисел, можно приблизительно оценить в зависимости от максимального *уровня* U чисел, возникающих в процессе вычислений. Этот уровень определяется по индукции: числа, определяющие вход текущего примера, имеют нулевой уровень; если операция умножения выполняется над числами уровня не более k, то результат имеет уровень, не превышающий $(k+1)$. Легко понять, что длина записи чисел уровня U может в 2^U раз превышать длину записи исходных чисел. Таким образом, при полиномиальном числе операций умножения число элементарных (битовых) операций может оказаться **экспоненциальным от длины записи исходной информации**. Следовательно, эффективность алгоритма можно гарантировать лишь в том случае, если в процессе его работы возникают числа, максимальный уровень U которых не превышает $O(\log N)$, где N — длина записи входа. К счастью, для большинства существующих алгоритмов (и для всех алгоритмов, рассматриваемых в данной диссертации) уровень U не превышает некоторой константы, что позволяет избежать подсчета количества битовых операций и ограничиться оценкой количества арифметических операций. Мы также будем следовать этому традиционному подходу при оценке трудоемкости того или иного алгоритма. (Проверку условия $U \leq \text{const}$ в каждом конкретном случае будем оставлять читателю.)

Теперь хотелось бы разъяснить смысл термина “длина записи исходной информации”, используемого в определении эффективного алгоритма. Прежде всего, для записи целых чисел мы будем использовать бинарную нотацию, т.е. запись чисел в двоичной кодировке, употребляемую во всех реальных вычислительных машинах. Так называемую *унарную нотацию*, требующую n битов для записи числа n, будем использовать для доказательства *псевдополиномиальности* того или иного алгоритма. Способ записи более сложных объектов (таких как графы) в каждом конкретном случае будет оговариваться отдельно, так как ясно, что от способа задания этих объектов на входе алгоритма зависит работа алгоритма и — в конечном итоге — оценка его трудоемкости.

Вернемся к “эффективным алгоритмам”, под которыми как правило будем понимать полиномиальные (реже — псевдополиномиальные) алгоритмы. В дискретной оптимизации практически с первых ее шагов сложилось убеждение, что эффективные алгоритмы — это единственное, что мы можем себе позволить при решении практических задач[29].

[29]Под “практическими” часто понимают задачи, весьма далекие от практики в силу их абстрагирован-

(Пример, приведенный на стр.15, подтверждает это весьма красноречиво.) В силу этого и "теория" пришла к выводу, что при исследовании какой-либо задачи дискретной оптимизации целью номер один является построение эффективного алгоритма ее решения. (Кажется, впервые эта мысль была сформулирована в 60-х годах Эдмонсом.) Однако, немало копий сломалось об эту цель, прежде чем возникли знаменитая теория сложности и понятие NP-полноты, а вместе с ними — и убеждение, что попытки построения эффективного алгоритма точного решения той или иной задачи следует прекращать, как только доказана ее NP-трудность.

Итак, первым шагом в анализе сложности какой-либо задачи является определение ее сложностного статуса: является ли она полиномиально разрешимой (в этом случае говорят, что задача "принадлежит классу P") или, напротив, является NP-трудной (т.е. представляется не более легкой, чем любая другая задача из класса NP). Следует отметить, что до сравнительно недавнего времени анализ сложности большинства задач дискретной оптимизации и заканчивался этим первым шагом. Все "теоретические" результаты подразделялись на два класса: положительных (полиномиальная разрешимость) и отрицательных (NP-трудность)[30]. Но что делать, если для большинства рассматриваемых задач "теоретический" результат оказывался отрицательным? Практическая необходимость решения этих задач никуда не исчезает после того как доказана их NP-трудность. Именно практическая необходимость привела к возникновению и развитию таких направлений исследования как:

— методы сокращения перебора вариантов при отыскании точного решения задачи (методы типа "ветви и границы");

— нетрудоемкие алгоритмы нахождения приближенного решения (куда, например, относятся различные алгоритмы "локального поиска"), — без каких-либо гарантий качества получаемых решений;

— алгоритмы полиномиальной трудоемкости для нахождения приближенного решения с **гарантированными оценками качества**.

Попробуем кратко охарактеризовать эти направления, не претендуя на безоговорочную истинность наших характеристик.

Большие надежды, возлагаемые вначале на методы типа ветвей и границ ("бум" развития этих методов приходится, на наш взгляд, на 60–70-е годы), довольно скоро сменились разочарованием. Выяснилось, что достигаемого в них сокращения перебо-

ности от многих реальных условий. В этих случаях практически единственное, что остается от практики, это *размерность* задачи. (Уточним, что под "размерностью задачи" в дискретной оптимизации понимается не общепринятое в математике понятие размерности какого-либо пространства, а всего лишь **размер**, т.е. длина записи исходных данных.) Так вот, задача называется "практической", если ее размерность близка к той, что может представлять практический интерес.

[30]В подтверждение сказанного достаточно заглянуть в вышедший 10 лет назад второй том монографии Танаева, Сотскова и Струсевича по теории расписаний [61]. Он является сборником упомянутых выше "положительных" и "отрицательных" результатов и практически не содержит результатов по приближенным алгоритмам с оценками. С одной стороны, это объясняется тем, что таких результатов в то время было еще сравнительно немного. С другой стороны, это отражает научный менталитет того времени. Приближенные алгоритмы тогда считались "не вполне чистой" теорией, поскольку не гарантировали абсолютно точного решения задачи.

ра вариантов оказывается недостаточно для решения задач "практических" размерностей[31]. С другой стороны, каких-либо теоретических оценок величины сокращения перебора (иначе говоря, оценок ожидаемого времени работы алгоритмов) в большинстве работ, посвященных данному направлению, не приводится, что снижает и теоретическую ценность подобных результатов. Конечно, полностью отрицать целесообразность этих исследований было бы неправильно. Часто методы сокращения перебора вариантов используются как вспомогательные при исследовании поведения других алгоритмов (например, алгоритмов локального поиска), когда требуется знать точные значения оптимумов для каких-то эталонных примеров, чтобы различать "попадание в оптимум". При этом нам, в принципе, не важно, за сколько часов счета был получен этот самый оптимум, однако есть заинтересованность в увеличении размерностей исследуемых примеров. Поэтому применяются методы сокращения перебора, которые все же лучше, чем полный перебор. Впрочем, мы отвлеклись, поэтому лучше двинуться дальше.

Второе направление исследований, бум которого просматривается сейчас, имеет, на взгляд автора, больше шансов быть полезным для практики, поскольку уже сегодня позволяет находить "сравнительно хорошие" решения задач практических размерностей за "сравнительно небольшое" время. Огромное количество статей на эту тему, как правило, содержат "численные результаты", т.е. таблицы чисел, отражающие время работы того или иного алгоритма на конкретном типе компьютеров и для конкретного класса *тестовых примеров*. Очевидно, что проверить какой-либо алгоритм **для всех** мыслимых классов примеров невозможно (если только не доказать оценку поведения алгоритма *в наихудшем случае*; но такой результат относится уже не ко второму, а к третьему направлению, речь о котором пойдет чуть ниже). Поэтому сравнение делается на так называемых "эталонных" примерах. При этом мало кто из пишущих программы людей задумывается над тем, почему именно тот, а не другой класс примеров выбран для сравнения, и насколько хорошо этот класс отражает частоту появления таких примеров в практических задачах. (Как правило, эти примеры порождаются "слепым" генерированием случайных чисел, не учитывающим реальных законов распределения.) Мы не будем здесь подробно касаться таких результатов, поскольку они принадлежат другой области (т.е. области не теоретических, а практических результатов). Конечно, нельзя исключить, что при анализе поведения алгоритмов "локального поиска" когда-нибудь появятся и теоретические результаты, обладающие свойством "общности" и не зависящие от таких факторов как "тип компьютера", "класс тестовых примеров", и т.п. Но это уже будут результаты не второго, а третьего направления, о котором сейчас и пойдет речь.

Качество того или иного алгоритма приближенного решения задачи можно оценивать различными способами, однако во всех случаях оценивается отклонение значения целевой функции на "приближенном" решении, найденном при помощи алгоритма, от *оптимума*, т.е. от значения целевой функции на оптимальном решении. Различают

[31] Например, большим достижением считается сравнительно недавний результат Карлье и Пинсона [79], состоящий в отыскании оптимального решения конкретного эталонного примера задачи job shop размера 10×10, — т.е. задачи нахождения расписания выполнения 10 работ на 10 машинах. Совершенно очевидно однако, что для практики эти размеры являются "игрушечными".

два типа оценок отклонения: *в среднем* и *в наихудшем случае*. Первые носят вероятностный характер и оценивают статистическое поведение алгоритма, т.е. насколько хорошие решения он получает в "большинстве случаев". Вторые носят абсолютный характер, т.е. гарантируют установленное качество получаемых решений для всех без исключения примеров исходной задачи. Естественным недостатком вторых оценок является то, что как правило, они являются сильно завышенными, т.е. для большинства примеров алгоритм дает значительно лучшие решения, чем это гарантировано а-приори. К недостатку же первых оценок можно отнести то, что они сильно привязаны к конкретной функции распределения на множестве значений исходных данных. Как правило, при исследовании задачи выбираются простейшие функции распределения, наиболее удобные для анализа (типа равномерного распределения на отрезке), а не те, которые более адекватно соответствуют реальности[32]. Тем самым исчезает якобы имеющееся преимущество первых оценок перед вторыми, состоящее в их большей "практичности". (Читатель, вероятно, уже догадался, что автор является приверженцем оценок второго типа.) Далее для оценивания качества решений, получаемых каким-либо алгоритмом, будут использоваться лишь оценки отклонения **в наихудшем случае**.

В качестве последней наиболее часто используется оценка *относительного уклонения* от оптимума. Иначе говоря, отношение $f(S_{\mathcal{A}})/f(S_{\text{opt}})$ оценивается сверху или снизу, в зависимости от того, минимизируется или максимизируется значение (неотрицательной) целевой функции $f(S)$, где $S_{\mathcal{A}}$ — решение, получаемое с помощью алгоритма $\mathcal{A}$, S_{opt} — оптимальное решение. Далее для определенности будем рассматривать лишь задачи на минимум функции f.

Алгоритм называется *ρ-приближающим* (в английской транскрипции — *ρ-approximation*), если для любого примера I исходной задачи гарантируется выполнение соотношения

$$f(S_{\mathcal{A}}(I))/f(S_{\text{opt}}(I)) \le \rho. \tag{4.1}$$

Как легко понять, оценки вида (4.1) удобны тем, что величина ρ **безразмерна**, она не зависит от того, в каких единицах измеряется значение функции f. (Например, если f имеет размерность времени, — что характерно для большинства задач третьей части данной книги, — то величина ρ будет иметь одно и то же значение независимо от того, измеряются ли длительности операций и производные от них величины в часах, минутах или секундах.) Неудобство же оценок вида (4.1) проявляется в тех случаях, когда на рассматриваемом классе примеров $\mathcal{I} = \{I\}$ исходной задачи величина оптимума $f(S_{\text{opt}}(I))$ может принимать значения, сколь угодно близкие к нулю. В частности, если $\min_{I\in\mathcal{I}} f(S_{\text{opt}}(I)) = 0$ и доказана NP-полнота распознавания равенства $f(S_{\text{opt}}(I)) = 0$[33], то для любой константы $\rho \ge 1$ вообще не существует эффективного ρ-приближающего алгоритма решения данной задачи (при условии, конечно, что P$\neq$NP). Этого недостатка лишены оценки *абсолютного уклонения* от оптимума вида

$$f(S_{\mathcal{A}}(I)) - f(S_{\text{opt}}(I)) \le G(I). \tag{4.2}$$

[32]Как в том анекдоте, где мужик искал потерянный кошелек под фонарем, потому что "там светлее".

[33]как это имеет место, например, для задачи коммивояжера с весами ребер 0 и 1

(Заранее отметим, что большинство оценок, доказываемых в данной книге, — хотя не все, — будут абсолютными.) Однако для того, чтобы оценки вида (4.2) были сравнимы между собой, необходимо привести их к единому стандарту, т.е. выбрать удобную единицу измерения функции f. Как это сделано в данной работе, мы увидим позже. А сейчас вернемся к оценкам вида (4.1).

В последние полтора–два десятилетия построение эффективных ρ-приближающих алгоритмов было одним из главных приоритетов в исследовании задач дискретной оптимизации вообще и теории расписаний — в частности. Действительно, поскольку подавляющее большинство интересующих нас задач оказываются "неподдающимися" с точки зрения эффективного точного решения, достойной целью становится построение эффективных алгоритмов, гарантирующих отыскание приближенных решений с хорошими оценками качества, и при этом — за вполне приемлемое время счета. Естественно полагать, что чем ближе величина ρ к единице, тем выше качество алгоритма (даже если это сопровождается некоторым увеличением трудоемкости алгоритма, при сохранении ее полиномиальности). Это положение послужило отправной точкой для развертывания настоящего спортивного состязания в научных журналах, направленного на построение ρ-приближающих алгоритмов для одних и тех же (наиболее модных или "эталонных") задач, со значениями ρ, всё более близкими к единице[34]. Поскольку нахождение точного решения NP-трудной проблемы невозможно за полиномиальное время[35], естественно ожидать, что гарантированное отыскание решений, всё более и более близких к оптимальному, не может происходить "бесплатно", без ухудшения эффективности алгоритма. Таким образом, если для сравнения ρ-приближающих алгоритмов применять двумерную характеристику (точность, эффективность), то в принципе возможно построение бесконечной серии **несравнимых по качеству** приближающих алгоритмов. В виде исключения, в некоторых случаях удается придумать алгоритм, превосходящий альтернативные как по точности, так и по эффективности[36].

К счастью (и возможно, к большому огорчению для производителей бумаги), существуют как внутренние, так и внешние причины, препятствующие бесконечному продолжению подобных соревнований для той или иной задачи. К внешним я бы отнес причины, проистекающие из способа получения оценок точности исследуемого алгоритма. Как правило, для обоснования оценки вида (4.1) используется нижняя оценка

[34] В качестве одного из ярких примеров такого состязания можно привести борьбу за улучшение оценок решения задачи упаковки в контейнеры, когда оценка ρ многократно уменьшалась.

[35] если P≠NP. В дальнейшем мы, по-возможности, будем опускать эту лицемерную оговорку. — Мало кто верит, что в один прекрасный день мы получим универсальный полиномиальный алгоритм для всех без исключения задач из класса NP.

[36] Несколько таких результатов было получено при участии автора диссертации. Так, для задачи flow shop с двумя машинами, n работами, поступающими в систему в заданные моменты времени $r_j,\ j = 1, \ldots, n$, и критерием "минимум $C_{\max}$" в соавторстве с К.Н. Каширских и Крисом Поттсом был построен $\frac{3}{2}$-приближающий алгоритм трудоемкости $O(n^3 \log n)$ [22]. До этого лучшим алгоритмом для данной задачи считался $\frac{5}{3}$-приближающий алгоритм Поттса, имеющий ту же оценку трудоемкости [102]. Другим примером является построенный совместно с И.Д. Черных [124] линейный по трудоемкости $\frac{4}{3}$-приближающий алгоритм решения трех-машинной задачи open shop, который по точности превосходит $\frac{3}{2}$-приближающий алгоритм Чена и Струсевича [82] при той же трудоемкости. Упомянутые результаты не вошли в данную диссертацию.

оптимума: $f(S_{\text{opt}}(I)) \geq F(I)$. Имея такую оценку, для получения оценки вида (4.1) достаточно выполнить две вещи: во-первых, доказать, что для любого примера I алгоритм $\mathcal{A}$ гарантирует нахождение решения $S_{\mathcal{A}}$, удовлетворяющего соотношению вида $f(S_{\mathcal{A}}(I)) \leq G(I)$, и во-вторых, оценить сверху величину $\sup_{I \in \mathcal{I}} G(I)/F(I)$. Если последнюю удается оценить сверху константой ρ, то наряду с построением ρ-приближающего алгоритма мы получаем теоретический результат, безотносительный к какому-либо алгоритму, а именно, что оптимум любого примера I лежит в интервале $[F(I), \rho F(I)]$. Ясно, что если найденный интервал (будем называть его *интервалом локализации оптимумов*) — минимально возможный[37], то дальнейшее улучшение оценки точности (4.1) возможно лишь при условии использования какой-либо нижней оценки оптимума, отличной от $f(S_{\text{opt}}(I)) \geq F(I)$. Известно однако, что поиск хороших нижних оценок оптимума в задачах дискретной оптимизации — проблема достаточно сложная. И это является техническим препятствием к построению алгоритмов с хорошими оценками точности вида (4.1).

Некоторые оптимизационные задачи обусловливают внутренние (т.е. проистекающие из существа самой задачи) причины, препятствующие построению эффективных ρ-приближающих алгоритмов[38]. В связи с этим оптимизационные задачи подразделяются на несколько классов по сложности их аппроксимации. К классу наиболее легко аппроксимируемых задач относятся те, которые допускают построение *полиномиальной аппроксимационной схемы* (PTAS), т.е. такого семейства алгоритмов $\{\mathcal{A}_\varepsilon \,|\, \varepsilon > 0\}$, что для любого фиксированного $\varepsilon > 0$ в данном семействе найдется $(1+\varepsilon)$-приближающий алгоритм $\mathcal{A}_\varepsilon$ полиномиальной трудоемкости. Ясно, что трудоемкость алгоритма $\mathcal{A}_\varepsilon$ должна ухудшаться по мере стремления ε к нулю. Если при этом она имеет полиномиальную зависимость от величины $1/\varepsilon$, то схема называется *вполне полиномиальной* и обозначается знакомой всем современным расписальщикам аббревиатурой FPTAS[39]. Наличие FPTAS для какой-либо задачи является, однако, редким явлением в теории расписаний. (В частности, FPTAS, как правило, не существует, если доказана NP-трудность задачи *в сильном смысле*[40].)

Следующий (по сложности аппроксимации) класс задач содержит такие задачи, которые не допускают построения ни FPTAS, ни PTAS, но допускают построение ρ-приближающих алгоритмов для некоторых значений ρ. В этот класс, например, входит задача $O||C_{\max}$, т.е. задача open shop с нефиксированным числом машин, рассматриваемая в главе 1 третьей части. В разделе 36 доказывается, что для данной задачи не существует ρ-приближающего алгоритма при $\rho < 5/4$. В то же время, простейший

[37] как это имеет место в упомянутом выше результате для трехмашинной задачи open shop: $4/3$-приближающий алгоритм находит решение $S_{\mathcal{A}}$ со значением $C_{\max}(S_{\mathcal{A}})$ из интервала $[F(I), 4/3F(I)]$, где $F(I) = \max\{L_{\max}, l_{\max}\}$, $L_{\max}$ — максимальная нагрузка машины, $l_{\max}$ — максимальная длительность работы. Поскольку $F(I)$ — нижняя оценка оптимума примера I, и существуют примеры, для которых выполнено равенство $C_{\max}(S_{\text{opt}}(I)) = \frac{4}{3}F(I)$, то найденный интервал локализации оптимумов является минимально возможным.

[38] Такие задачи в англоязычной литературе называют *hard to approximate*.

[39] от английского *fully polynomial time approximation scheme*

[40] По крайней мере, это верно для задач с целевыми функциями, оптимум которых не превосходит полинома от длины унарной записи исходных данных.

жадный алгоритм (как показано Анной Рачмань), является 2-приближающим. Таким образом, для данного класса задач представляет интерес нахождение *точного барьера приближаемости*, т.е. такого минимального числа ρ, для которого существует полиномиальный по времени ρ-приближающий алгоритм.

Более сложными для аппроксимации являются задачи, которые ни для какой константы ρ не допускают построения ρ-приближающего алгоритма. (К таким задачам, как было сказано выше, относится задача коммивояжера без неравенства треугольника.) Для таких задач всё же рассматриваются ρ-приближающиие алгоритмы, где ρ является уже не константой, а функцией от каких-то параметров задачи (например, от числа машин или работ). — И здесь возможна бесконечная иерархия алгоритмов, поскольку существует бесконечное разнообразие функций от заданных параметров. Однако мы не будем далее углубляться в теорию сложности. (Тем более, что диссертация не содержит результатов по ρ-приближаемости каких-то задач для неконстантных значений ρ.)

Основной целью исследования оптимизационных задач в нашей диссертации будет построение эффективных алгоритмов их приближенного решения, причем в качестве гарантированных оценок точности этих алгоритмов будут использоваться, как правило, оценки абсолютного отклонения от оптимума, т.е. оценки вида (4.2). Из этих абсолютных оценок нетрудно получить также оценки относительного отклонения от оптимума, которые (для рассматриваемой нами целевой функции $C_{\max}$) обладают замечательным свойством **стремления к нулю с ростом размерности задачи**. Поясним смысл последней фразы на конкретном примере. Для задачи Джонсона, к примеру, нами построен алгоритм $\mathcal{A}$ с оценкой

$$C_{\max}(S_{\mathcal{A}}) \leq L_{\max} + \mu(m)p_{\max}, \tag{4.3}$$

где $L_{\max} = \max_{i=1,\dots,m} \sum_{j=1}^{n} p_i^j$ — максимальная нагрузка машины, $p_{\max}$ — максимальная длительность операции, n — число работ, m — число машин. Поскольку величина $L_{\max}$ является нижней оценкой оптимума, то из (4.3) получаем оценку относительной погрешности решения $S_{\mathcal{A}}$:

$$\frac{C_{\max}(S_{\mathcal{A}}) - C_{\max}(S_{\text{opt}})}{C_{\max}(S_{\text{opt}})} \leq \frac{\mu(m)p_{\max}}{L_{\max}}. \tag{4.4}$$

Анализируя правую часть неравенства (4.4), замечаем, что знаменатель $L_{\max}$ является суммой n величин p_i^j, в то время как величина $p_{\max}$ в числителе равна максимальной из этих величин, а функция $\mu(m)$ не зависит от числа работ n. Таким образом, для любых фиксированных значений $\varepsilon > 0,\ \delta > 0$ и целого m найдется значение n_0, такое что для любого $n \geq n_0$ выполнено свойство

$$\mathbf{P}\{\mu(m)p_{\max}/L_{\max} > \varepsilon\} < \delta, \tag{4.5}$$

что гарантирует так называемую *асимптотическую оптимальность*[41] решений, получаемых алгоритмом $\mathcal{A}$, т.е. стремление к нулю их относительной погрешности при

[41]Прародителями этого термина, как и вообще направления построения эффективных приближающих алгоритмов с гарантированными оценками точности для дискретных оптимизационных задач, по-видимому, следует считать Э.Х. Гимади, Н.И. Глебова и В.А. Перепелицу [27, 14, 15, 11, 12]. Но еще раньше (в 1962 году) А.А. Боровковым [8] был построен алгоритм приближенного решения задачи коммивояжера в пространстве $\mathbb{R}^m$ с гарантированной (при определенных условиях) оценкой точности.

$n \to \infty$.

Для более точной формулировки утверждений такого типа нам пришлось бы конкретизировать функцию распределения величин p_i^j (поскольку в противном случае вывод оценки вероятности выполнения неравенства $\mu(m)p_{\max}/L_{\max} > \varepsilon$ был бы затруднителен). Однако мы не будем этого делать, а лишь заметим, что свойство (4.4) выполняется для довольно широкого семейства функций распределения. В самом деле, от функций распределения случайных величин p_i^j достаточно потребовать, чтобы отношение максимума из n величин к сумме этих величин имело тенденцию стремления к нулю с ростом n.

В четвертой главе третьей части диссертации будет показано, что аналогичные *асимптотически точные* алгоритмы, обладающие свойством асимптотической оптимальности получаемых ими решений, удается построить не только для простейших многостадийных задач теории расписаний, но и для значительно более сложных, таких как задачи job shop, dag shop и их обобщения (правда, пока исследовались лишь задачи на минимум $C_{\max}$). Нетрудно понять, что такие алгоритмы неплохо дополняют переборные алгоритмы (типа метода "ветвей и границ"): чем больше размерность задачи (иначе говоря, чем ближе эта размерность к размерам, представляющим практический интерес), тем хуже справляются с ней переборные алгоритмы, но тем более точные решения получаются с помощью наших нетрудоемких алгоритмов.

Иногда приходится сталкиваться с мнением, что алгоритмы с абсолютными оценками качества "плохи", поскольку не гарантируют хороших **относительных** оценок качества получаемых решений, справедливых для всех без исключения примеров данной задачи. В качестве контраргумента в этом споре можно было бы выдвинуть симметричное утверждение о том, что алгоритмы с хорошими относительными оценками качества также далеко не всегда обеспечивают хорошие абсолютные оценки. Однако мы поступим более правильно, если согласимся с героем Ильфа и Петрова, заявившего, что "спор здесь неуместен". Спор о том, какие оценки (абсолютные или относительные) более нужны, а какие — менее, совсем недавно блестяще диалектически разрешился благодаря результатам Йенсена, Солис-Оба и Свириденко [91]. Ими была построена серия аппроксимационных схем (PTAS) для различных вариантов задач job shop и flow shop (с различными наборами ограничений), — т.е. получены результаты, наиболее уважаемые апологетами относительных оценок. При этом важнейшим элементом их аппроксимационных схем являются (излагаемые в третьей части настоящей диссертации) алгоритмы решения задач job shop и flow shop с **абсолютными** оценками качества, которые Йенсен, Солис-Оба и Свириденко применяют для так называемых "маленьких" работ. И если вдуматься, ничего удивительного в этом нет. Как показывает практика построения аппроксимационных схем, самая большая проблема как правило возникает именно с "маленькими" работами. (Построение расписания для "больших" работ, — число которых обычно ограничено константой, — может быть выполнено полным перебором.) А наши алгоритмы как раз и хороши тогда, когда все работы достаточно маленькие, т.е. имеют небольшие длительности. Таким образом, победила дружба[42].

[42]Кстати, схемам Йенсена, Солис-Оба и Свириденко предшествовал построенный Шмойсом, Стайном,

* * *

Резюмируем содержание этого раздела. Итак, объектами нашего исследования будут дискретные оптимизационные задачи (в некоторых случаях — задачи на распознавание существования допустимого решения, без какой-либо целевой функции), а также алгоритмы их решения. Целью исследования интересующих нас задач является анализ их сложности — как с точки зрения их точного решения, так и ρ-приближения. Алгоритмы решения задач будут исследоваться с целью отыскания **эффективных** методов решения этих задач, — либо точного, либо приближенного, с гарантированными относительными (вида (4.1)) либо абсолютными (вида (4.2)) оценками качества получаемых решений.

Некоторые задачи будут исследоваться с целью выявления наиболее широких классов примеров этих задач, допускающих эффективное точное решение.

Наконец, установление связей и зависимостей между различными дискретными оптимизационными задачами также входит в цели наших исследований, поскольку способствует отысканию новых эффективных методов решения этих задач.

5 Методы исследования

Здесь, согласно протоколу, мы должны пояснить, с использованием каких методов были получены результаты, представленные в данной диссертации. Если вдуматься, то возможных “методов исследования”, используемых математиками, не так уж много. А именно, два: с помощью головы и с помощью компьютера. Иногда удается применить их комбинацию. К компьютерным методам исследования я отношу те, в которых программируется какой-то алгоритм, затем “гоняется” на миллионе задач, а полученные числа оформляются в виде красивых таблиц, графиков и диаграмм.

Методы “с помощью головы” в каком-то смысле проще, поскольку не требуют наличия под рукой дорогостоящего оборудования. (За “голову” у нас платят гораздо меньше.) Иногда не требуется даже авторучки[43].

Очень эффективной, на мой взгляд, является комбинация двух указанных методов. Так например, для доказательства того, что интервал $I = [L_{\max}, 4/3L_{\max}]$ является минимально возможным интервалом локализации оптимумов трехмашинной задачи open shop, мы (т.е. я и мои студенты) сначала использовали только первый метод. Однако после трех лет исследований, когда было рассмотрено и описано огромное количество

Вайном [130] алгоритм решения задачи job shop с относительной оценкой точности порядка $O(\log^2(mr))$, где m — число машин, а r — максимальное число операций работы. Этот алгоритм также опирался на построенный нами алгоритм с абсолютной оценкой точности из раздела 44.

[43]Например, результат из раздела 15 (доказательство возможности суммирования векторов из $\mathbb{R}^m$ в области, ограниченной по модулю единицей — по одной из координат, и функцией от m — по другим координатам) был получен во время дежурства в ДНД (Добровольной Народной Дружине). Большинство деталей этого результата прояснилось к концу третьего круга по маршруту Ильича – Морской проспект – Жемчужная. “Методом исследования” в данном случае явилось равномерное вышагивание на свежем воздухе.

"случаев и подслучаев", стало ясно, что нерассмотренных случаев еще больше. Появилось ощущение, что мы делаем "что-то не так". Проанализировав метод нашего доказательства, мы пришли к выводу, что его можно формализовать, свести к решению конечного (правда, довольно большого) числа линейных программ, а решение последних поручить компьютеру. — Что и было сделано. Первая версия этой программы "доказала" гипотезу о локализации оптимумов внутри указанного интервала I "всего" за две недели непрерывного счета. Более усовершенствованной программе для этого потребовалось не более суток. Одновременно с положительным ответом на интересующий нас вопрос было построено *дерево доказательства*, которое позволяло по каждому конкретному примеру практически мгновенно находить приближенное расписание со значением длины из интервала I: для этого достаточно за один просмотр списка работ (т.е. за линейное от n время) "склеить" их в пять агрегированных работ, после чего отыскать в дереве доказательства путь из корня в вершину, соответствующий полученному агрегированному примеру (этот путь содержит не более восьми вершин-расписаний) и из этих лежащих на пути 8 расписаний выбрать наиболее короткое. Таким образом, мы, во-первых, построили рекордный по быстродействию алгоритм приближенного решения трехмашинной задачи open shop с оценкой точности $4/3$ (т.е. сэкономили время "практиков", желающих воспользоваться нашим алгоритмом для отыскания приближенного решения задачи $O3||C_{\max}$ с гарантированной оценкой точности $4/3$), а во-вторых, значительно сократили длину доказательства теоретического результата. — Вместо скрупулезного разбора случаев и подслучаев (на описание которых потребовалось бы, вероятно, не менее 200–300 страниц) читателю достаточно, во-первых, разобраться с (гораздо более коротким) описанием алгоритма, реализованного в программе (т.е. убедиться, что он **способен** доказать или опровергнуть интересующее нас утверждение), а во-вторых, запустить эту программу на счет и убедиться, что генерируемый ею ответ **действительно положителен**. Тем самым, экономится время "теоретиков", желающих убедиться в правильности доказательства нашей теоремы[44].

[44] Хотя этот результат также не вошел в диссертацию, он является довольно поучительным примером человеко-машинного доказательства теорем, и его несомненно следует включить в содержание книги, посвященной описанию Эффективных методов (приближенного или точного) решения задач теории расписаний — буде таковая когда-нибудь написана.

Вполне формальное введение

6 Общая характеристика работы

А. Общее направление исследований

Диссертация посвящена разработке эффективных методов решения NP-трудных дискретных оптимизационных задач. Главной “мишенью” при этом являются многостадийные системы теории расписаний. При анализе возникающих здесь задач обнаруживается их тесная связь с другими дискретными задачами, не относящимися к области теории расписаний. Установление таких связей позволяет, во-первых, лучше понять природу исследуемых задач, и во-вторых, способствует разработке новых эффективных методов их решения. Так например, в диссертации установлена связь задач на построение кратчайших расписаний в многостадийных системах — с семейством геометрических задач о суммировании векторов. Разработка эффективных алгоритмов нахождения оптимальных (или близких к ним) порядков суммирования векторов, укладывающих траекторию суммирования векторов в некоторую экстремальную область или семейство областей m-мерного пространства, позволяет строить эффективные алгоритмы приближенного решения задач теории расписаний с гарантированными оценками точности. В свою очередь, анализ геометрических задач о суммировании векторов обнаруживает их тесную взаимосвязь с задачами, лежащими в других областях дискретной математики.

Другим направлением исследования в диссертации является отыскание широких полиномиально разрешимых классов примеров NP-трудных задач. Наиболее “удобной” в этом отношении объектом оказалась задача open shop, исследованию которой посвящена самая большая глава из третьей части диссертации. Здесь автором введено понятие *нормального примера*, т.е. примера, длина оптимального расписания которого совпадает с максимальной машинной нагрузкой. Главным результатом этих исследований является вывод о том, что “нормальность” — это чрезвычайно распространенное явление среди примеров задачи open shop. Кроме того, выделяются широкие *эффективно нормальные* классы примеров, т.е. классы нормальных примеров, допускающие их эффективное точное решение. Представлено три разных подхода к построению таких классов: на основе метода компактного суммирования векторов, с использованием модифицированного метода Фиалы, и на основе анализа разностей нагрузок машин. Показано, что три указанных метода покрывают значительную часть множества всех

примеров задачи open shop. (Более того, при **фиксированном** числе машин и произвольном числе работ множество "ненормальных" примеров есть множество меры нуль.)

В. Главные результаты диссертации

- Разработан геометрический метод эффективного приближенного решения ряда NP-трудных задач теории расписаний (типа задачи job shop) с критерием *минимум длины расписания* [42]–[45],[50], [52]–[53],[56]–[57],[103]– [109],[113]–[115],[117], [126], [128], для которых ранее не было известно приемлемых методов решения. Метод основан на идее приближенного сведения рассматриваемых задач к задачам о суммировании векторов в конечномерном нормированном пространстве и последующем эффективном приближенном решении полученных геометрических задач. Построенные алгоритмы обладают свойством *асимптотической оптимальности*, т.е. стремления погрешности получаемых ими решений к нулю с ростом размерности задачи.

- Доказана теорема 11.5 [48], устанавливающая возможность суммирования произвольного 0-семейства векторов в различных выпуклых областях пространства $\mathbb{R}^m$, зависящих от произвольного вектора $a \in \mathbb{R}^m$. Из теоремы вытекает следствие, согласно которому для любой нормы s с центрально симметричным (не обязательно относительно начала координат) единичным шаром всякое s-семейство векторов может быть просуммировано в шаре радиуса $m-1+1/m$. Этот результат улучшает как известные ранее оценки минимального радиуса суммирования в задаче о компактном суммировании векторов (в частности, экспоненциальные от m оценки Бергстрёма [77] и Кадеца [20]), так и результаты самого автора диссертации, полученные им ранее. (Оценка радиуса m, справедливая для любых, в том числе несимметричных норм [35]. Ранее было показано [16], что оценка m неулучшаема для несимметричных норм, и было не известно, можно ли ее улучшить для произвольного m в случае симметричных норм.) Примечательно, что улучшение гарантированных оценок радиуса суммирования достигнуто без потери эффективности алгоритма.

- Исследованы свойства и получены верхние и нижние оценки *функций Штейница* (принимающих значения минимального радиуса суммирования векторов в пространстве $\mathbb{R}^m$ с той или иной нормой s) [45].

- Для задач теории расписаний типа задачи о сборочной линии и задачи job shop установлено существование *интервала локализации оптимумов* [104],[44],[111], [127]. Знание этих интервалов позволяет с достаточной точностью (не зависящей от таких параметров исходной задачи как число работ) эффективно локализовать значение оптимума любого примера рассматриваемой NP-трудной задачи.

- Определено понятие *нестрогого суммирования векторов* в семействе областей m-мерного пространства. Для произвольной нормы s в $\mathbb{R}^2$ получены достаточные

условия, гарантирующие возможность нестрогого суммирования любого s-семейства векторов в исходной выпуклой области пространства (теоремы 13.8 и 13.21 [53]). Разработаны эффективные алгоритмы нахождения такого суммирования при выполнении достаточных условий. С использованием нестрогого суммирования векторов найдены неулучшаемые интервалы локализации оптимумов для ряда задач нахождения кратчайшего расписания на трех машинах.

- На основе понятия *нормальности* [95] найдены широкие полиномиально разрешимые классы примеров задачи open shop. Представлено три разных подхода к построению таких классов: на основе метода компактного суммирования векторов [114], с использованием усовершенствованного метода Фиалы [49] и на основе анализа разностей нагрузок машин [51],[54], [95]. Из полученных результатов следует, что нормальность является типичным явлением на множестве примеров задачи open shop.

- Для многопроцессорной задачи open shop с фиксированным числом машин совместно с Г. Вёгингером построена полиномиальная аппроксимационная схема (PTAS) линейной трудоемкости [9]. Таким образом, найден ответ на давно стоявший открытый вопрос о границе приближаемости задачи open shop. В то же время, для любого $\rho < 5/4$ доказана невозможность (при условии P≠NP) построения ρ-приближающего полиномиального алгоритма решения задачи open shop в случае, когда число машин является переменной[45]. Тем самым, в анализе сложности этой задачи установлен "водораздел" между случаями, допускающими построение полиномиальной аппроксимационной схемы (когда m — любое фиксированное число), и случаями, когда такой схемы не существует (m — переменная).

- Найдена тесная связь между комбинаторными задачами из различных областей дискретной математики, позволяющая использовать результаты анализа одних задач для отыскания интересных свойств решений других задач [45], [121]. Так например, нетривиальные нижние оценки определенных нами функций Штейница для норм $l_p,\ p \geq 2$, могут быть получены с использованием матриц Адамара, а для произвольной нормы s — с помощью функции Дворецкого. Установлена взаимосвязь между задачами: *эквидистанции* на булевом кубе, балансировки, равномерного разбиения, нахождения подмножества векторов с заданной суммой, компактного суммирования векторов, объемно календарного планирования, и др.

C. Публикации и апробация результатов исследований

Диссертант является автором 83 научных работ. По теме диссертации опубликовано 73 работы, в том числе:
21 — в международных журналах,

[45] Результат получен автором диссертации независимо и опубликован в совместной статье с шестью соавторами [136].

14 — в центральных изданиях,
12 — в сборниках трудов Института математики,
6 — в препринтах европейских университетов,
19 — в тезисах конференций.

Результаты диссертанта по теме диссертации опубликованы в работах: [9], [13]–[16], [22]–[28], [30]–[59], [66]–[69], [89], [94]–[95], [103]–[104], [107]–[129], [133], [136].

Результаты диссертанта включались в монографии, учебники и обзоры по теории расписаний и по функциональному анализу, написанные другими авторами. Из известных диссертанту:

- монография Танаева, Сотскова и Струсевича по Многостадийным системам теории расписаний [61];
- обзор Чена, Поттса, Вёгингера по теории расписаний [81];
- учебное пособие Сотскова, Струсевича и Танаева по Математическим моделям и методам календарного планирования [60];
- учебное пособие Гимади и Глебова по Дискретным экстремальным задачам принятия решений [10];
- учебное пособие В.М. Кадеца и М.И. Кадеца по Перестановкам рядов в пространствах Банаха [21];
- монография Мас-Колела по Теории общего экономического равновесия [99];
- спецкурс Бенгта Нилссона по теории расписаний в университете г. Лунда (Швеция) [100].

Результаты, представленные в диссертации, докладывались автором:[46]

- Конференция по теоретической кибернетике, Кишинев (1982);
- 2-й Всесоюзный семинар по дискретной математике и ее приложениям, Москва (1987);
- Семинар профессора Катоны по Комбинаторике в Математическом институте Венгерской академии наук (Будапешт, Венгрия, 1989);
- Конференция по теоретической кибернетике (Тбилиси, 1990);
- 14-й Международный симпозиум по Математическому программированию (Амстердам, Нидерланды, 1991);

[46]Приводятся выступления диссертанта, начиная с 1982 года, — после защиты им кандидатской диссертации.

- Международная конференция по Исследованию операций (Берлин, Германия, 1994);
- Второй рабочий семинар по Моделям и Алгоритмам в Планировании и Теории Расписаний (Вернигероде, Германия, 1995);
- Семинар профессора Ленстры по дискретной математике в Техническом Университете Эйндховена (Эйндховен, Нидерланды, 1995);
- Семинар профессоров Дойбера и Аалсведе по дискретной математике в Техническом Университете Билефельда (Билефельд, Германия, 1996);
- 11-я Международная конференция по проблемам теоретической кибернетики (Ульяновск, 1996);
- Второй Сибирский Конгресс по Прикладной и Индустриальной Математике — ИНПРИМ-96 (Новосибирск, 1996);
- Третий рабочий семинар по Моделям и Алгоритмам в Планировании и Теории Расписаний (Кембридж, Англия, 1997);
- Дагштуль-семинар по Комбинаторным Приближенным Алгоритмам (Шлес Дагштуль, Германия, 1997);
- Пятый Европейский Симпозиум по Алгоритмам — ESA'97 (Грац, Австрия, 1997);
- Семинар профессора Брэзел по дискретной математике в Техническом Университете Магдебурга (Магдебург, Германия, 1998);
- 13-й рабочий семинар по Дискретной оптимизации (Бург, Германия, 1998);
- Третий Сибирский Конгресс по Прикладной и Индустриальной Математике — ИНПРИМ-98 (Новосибирск, 1998);
- Четвертый рабочий семинар по Моделям и Алгоритмам в Планировании и Теории Расписаний (Ренесса, Нидерланды, 1999);
- 14-й рабочий семинар по Дискретной оптимизации (Хольцхау, Германия, 2000);
- Семинар профессора Брэзел по дискретной математике в Техническом Университете Магдебурга (Магдебург, Германия, 2000);
- Международная конференция по дискретному анализу и исследованию операций — DAOR'2000 (Новосибирск, 2000);
- Международный рабочий семинар по методам дискретной оптимизации в теории расписаний и компьютерном дизайне (Минск, 2000);
- Семинар профессора Танаева по теории расписаний в Институте Технической Кибернетики (Минск, 2000).

D. Структура работы

Диссертация состоит из введения (неформального и вполне формального) и трех частей. Третья часть состоит из четырех глав, а главы и части в свою очередь состоят из разделов.

В неформальном введении описывается история возникновения данного направления исследований, объясняются мотивы выбора названия диссертации, а также описываются область, объект, цели и методы исследования.

В формальном введении дается общая характеристика работы, приводится обзор результатов и вводятся основные понятия и определения.

Первая часть диссертации посвящена исследованию задач суммирования векторов. Вводятся понятия строгого и нестрогого суммирования в заданной области или семействе областей. Строятся эффективные алгоритмы суммирования векторов в различных областях пространства. Доказывается NP-трудность проверки существования допустимого суммирования в заданной области для простейших областей пространства. Исследуются свойства и выводятся нижние и верхние оценки функций Штейница. Решаются также оптимизационные задачи об отыскании оптимальных подмножеств векторов.

Во второй части рассматриваются комбинаторные задачи различной природы (такие как задача о равномерном разбиении предметов, задача о камнях с запретами, о максимальном потоке, равномерном по стокам, об эквидистанции, и др.). В основном эти задачи носят вспомогательный характер, а разрабатываемые для их решения эффективные алгоритмы используются для построения эффективных алгоритмов для задач построения расписаний, рассматриваемых в третьей части. Устанавливаются связи между задачами, соотношения между определяемыми в них экстремальными функциями.

Третья часть диссертации содержит главный объект исследований — задачи построения оптимальных расписаний в многостадийных обслуживающих системах. Рассматриваемые здесь задачи различаются по типу обслуживающих систем, поэтому вся часть разбита на главы. Первая (наибольшая по объему) глава посвящена всестороннему исследованию систем с нефиксированными маршрутами (так называемых систем *открытого типа*). Бóльшая часть приводимых здесь результатов направлена на отыскание полиномиально разрешимых классов примеров задачи open shop. Приводятся также алгоритмы приближенного решения с абсолютными оценками точности, полиномиальная аппроксимационная схема, доказывается теорема о “неприближаемости” задачи в случае нефиксированного числа машин.

Вторая глава рассматривает системы поточного типа. Строятся эффективные приближенные алгоритмы для классической и для многопроцессорной задач flow shop. Уточняются интервалы локализации оптимумов задачи.

Четвертая глава посвящена исследованию систем наиболее общего вида, допускающим одновременное наличие различных технологических маршрутов. Здесь рассматриваются задачи job shop, dag shop и их обобщение на случай существования альтернативных исполнителей каждой операции. Приводится эффективный алгоритм приближенного решения наиболее общей задачи. В то же время, для частных случаев задачи job shop (таких как задача Акерса-Фридман для трех машин и двухмаршрутные задачи

трех машин) приводятся существенно более эффективные алгоритмы с лучшими оценками точности. В некоторых случаях полученные оценки являются неулучшаемыми.

Наконец, в отдельную главу выделен результат по задаче о сборочной линии, так как она не вкладывается ни в одну из систем, рассматриваемых в других главах.

Нумерация.

При изложении материала используются такие структурные элементы как теоремы, леммы, утверждения, замечания, следствия и определения. Все эти “теоремоподобные” элементы имеют единую нумерацию, которая начинается заново в каждом разделе. Например, если после теоремы с номером 5 идет следствие, то оно получает номер 6, а следующее за ним утверждение — номер 7, и т.д. Другие (не “теоремоподобные”) элементы, такие как рисунки, таблицы, вынесенные формулы и алгоритмы, имеют независимую нумерацию, которая также начинается заново в каждом разделе. Полный номер того или иного элемента состоит из двух чисел: номера раздела и номера элемента внутри раздела. Для всех разделов принята сквозная, не зависящая от частей и глав нумерация, при этом каждый раздел нумеруется единственным числом. Всего диссертация содержит 45 разделов.

Таким образом, нумерация всех структурных элементов подчиняется единому, легко запоминающемуся правилу. Как представляется автору, такая нумерация позволяет читателю легко ориентироваться в тексте. Например, если где-то есть ссылка на теорему 28.15, то ясно, что ее следует искать в разделе 28 где-то между определением 28.14 и утверждением 28.16. Впрочем, в особо важных случаях при ссылке на теорему указывается номер страницы, на которой она находится.

Понятно, что достоинства любой системы порождают соответствующие им недостатки. (Как говорится, за всякое удовольствие нужно чем-то платить.) Например, при выбранной нами системе труднее наводить статистику (т.е. подсчитывать, сколько каких элементов содержится в диссертации). Исправляя этот недостаток, приведем несколько цифр[47].

Статистика.

Диссертация изложена на 280 страницах, содержит 63 теоремы, 50 лемм, 36 следствий, 38 утверждений, 11 замечаний и 25 определений, библиографию из 136 наименований, 23 рисунка и 2 таблицы.

7 Обзор результатов диссертации

Первая часть диссертации посвящена исследованию задач суммирования векторов. Рассматриваемые здесь задачи делятся на две большие группы: задачи нахождения подмножеств векторов и задачи нахождения порядка суммирования векторов.

[47] Для подсчета этой статистики автору не пришлось перелистывать всю диссертацию. Достаточно было в редакторе WinEdit для каждого элемента (например, теорем) запустить команду глобальной замены (по всем открытым файлам) слова {theorem} на слово {theorem}.

Первая группа задач рассматривается в разделе 10. Вначале доказывается важная лемма 10.2, лежащая в основе доказательства теоремы 11.5 о компактном суммировании векторов, а также используемая в конце раздела 10 при решении задачи о подмножестве векторов с заданной суммой. Затем мы переходим к рассмотрению подмножеств ребер графа и определяем понятие *независимого подмножества ребер*. Это понятие оказывается тесно связанным с понятием независимого множества векторов. Доказывается лемма 10.6 об алгоритме нахождения базы семейства ребер, являющаяся аналогом леммы 10.2 об алгоритме нахождения базы семейства векторов. Далее рассматривается оптимизационная задача о нахождении подсемейства векторов заданного семейства X с суммой, наиболее близкой к заданному значению x (где близость оценивается в заданной — вообще говоря, произвольной — норме s). Выделяется два случая этой задачи, допускающие эффективное приближенное решение с гарантированной точностью (при этом построение эффективных алгоритмов основывается на доказанных ранее леммах): когда точка x принадлежит параллелепипеду, построенному на векторах из X (в этом случае задача интерпретируется как задача о нахождении вершины параллелепипеда, наиболее близкой к его заданной внутренней точке), и когда норма s допускает линеаризацию (т.е. ее единичный шар представи́м в виде пересечения конечного числа замкнутых полупространств). В конце раздела рассматривается задача (ПО(k,q)) о нахождении такого подмножества операций, которое содержало бы заданное количество q операций каждой работы, и при этом давало бы нагрузку на каждую машину, пропорциональную числу q/k. В лемме 10.14 доказывается возможность решения этой задачи с большой точностью и малой трудоемкостью.

В разделе 11 рассматривается задача о компактном суммировании векторов (КСВ). Формулируется одномерный аналог этой задачи — задача о компактном суммировании чисел (КСЧ), соответствующая задача на распознавание, зависящая от параметра $\mu > 0$, обозначается КСЧ(μ). В теореме 11.4 доказывается, что при любом фиксированном $\mu \in [\frac{1}{2}, 1)$ задача КСЧ(μ) NP-полна в сильном смысле, в то время как при $\mu \geq 1$ ответ в задаче КСЧ(μ) всегда положителен, а при $\mu \leq \frac{1}{2}$ — всегда отрицателен. Как следствие, задача КСВ оказывается NP-трудной в сильном смысле, что оправдывает последующие результаты об алгоритмах ее приближенного решения. Важный результат содержится в теореме 11.5, где для любого 0-семейства векторов $\{x_1, \ldots, x_n\} = X \subset \mathbb{R}^m$ и любого вектора $a \in \mathbb{R}^m$ гарантируется нахождение (с трудоемкостью $O(n^2m^2)$) перестановки $\pi = (\pi_1, \ldots, \pi_n)$, для которой все частичные суммы $\sum_{i=1}^{k} x_{\pi_i}$ содержатся в выпуклом множестве $(m-1)B(X) + B_a$, где $B(X)$ — выпуклая оболочка векторов из X, а B_a — выпуклое множество, зависящее от вектора a. В качестве прямого следствия получаем возможность эффективного суммирования любого 0-семейства векторов из единичного шара произвольной симметричной нормы внутри шара радиуса $m - 1 + \frac{1}{m}$. Полученная оценка радиуса суммирования является наилучшей на сегодняшний день. Результат остается в силе и в случае несимметричной нормы, если ее единичный шар центральносимметричен (не обязательно относительно начала координат). Из доказываемой далее теоремы 11.10 следует, что "коэффициент растяжения" выпуклого множества Ψ, содержащего векторы исходного семейства X, можно понизить до $m-1$, если Ψ есть выпуклое неограниченное множество, имеющее объемный рецессивный конус.

В разделе 12 выводятся оценки и исследуются свойства функций Штейница $\varphi_s(m)$. (Напомним, что функция $\varphi_s(m)$ принимает значения, равные минимальному радиусу шара нормы s в пространстве $\mathbb{R}^m$, внутри которого при подходящем порядке суммирования может быть просуммировано любое s-семейство векторов.) Для норм $l_p,\ p \geq 1$, получены оценки (см. следствие 12.10 и теорему 12.13), из которых следует, что инфимум значений $\varphi_{l_p}(m)$ по всем нормам $l_p,\ p \geq 1$, стремится к бесконечности с ростом m (теорема 12.11). Вопрос о бесконечном росте минимума значений функций Штейница по **всем** нормам s остается открытым. (Как показывается в разделе 20, этот вопрос сводится к "более простому", однако остающемуся до сих пор открытым вопросу о нахождении бесконечно растущей нижней границы функции Дворецкого для произвольной нормы s.) Для случая двумерного пространства в теореме 12.4 доказывается оценка

$$\inf_s \varphi_s(2) \geq \sqrt{9/8}.$$

Вывод нижних оценок функций $\varphi_{l_p}(m)$ для норм $l_p,\ p \geq 2$, основывается на существовании матриц Адамара A_n, порядок которых (n) близок к размерности пространства m (теорема 12.6). Понятно, что наилучшие оценки получаются в том случае, если справедлива гипотеза Адамара (о существовании матриц A_n для всех n, кратных 4). Однако в случае нормы l_∞, как показано в лемме 12.8, для справедливости наилучшей оценки

$$\varphi_{l_\infty}(m) \geq \left\lceil \frac{1}{2}\sqrt{m+2} \right\rceil$$

достаточно, чтобы гипотеза Адамара выполнялась в ослабленной форме: для всех n, равных квадрату четного числа.

В разделе 13 вводится понятие *нестрогого суммирования векторов* в заданной области или семействе областей. Поскольку при нестрогом суммировании (по сравнению со "строгим") ослабляются требования к траектории суммирования (частичным суммам разрешается иногда выходить из заданной области), это позволяет гарантировать возможность суммирования в меньших областях. (И как мы увидим в третьей части диссертации, в ряде случаев — для задач с тремя машинами — находить минимально возможные интервалы локализации оптимумов.) Как и в случае строгого суммирования доказывается, что задача о нестрогом суммировании чисел (т.е. задача о суммировании одномерных векторов) NP-трудна в сильном смысле (теорема 13.18). Для двумерного пространства и произвольной нормы s находятся достаточные условия (теоремы 13.8 и 13.21), которым должно удовлетворять выпуклое множество $G \subset \mathbb{R}^2$, чтобы всякое s-семейство векторов из $\mathbb{R}^2$ могло быть нестрого просуммировано в G. В качестве такого достаточного условия в теореме 13.8 требуется, чтобы G было выпуклым неограниченным множеством, содержащим начало координат, и чтобы все его хорды, проходящие через точку 0, имели длину (в норме s) не меньше 1. Если при этом известен рецессивный конус множества G (т.е. конус, составленный из всех направлений, по которым G удаляется в бесконечность), то искомый порядок векторов, задающий их нестрогое суммирование в G, находится эффективно (за $O(n \log n)$ операций). В теореме 13.21 множество G не обязано быть неограниченным, но всякая

его хорда, имеющая непустое пересечение с единичным шаром, должна иметь длину не меньше 1. При выполнении достаточного условия искомая перестановка векторов также находится с трудоемкостью $O(n \log n)$.

В разделе 14 мы переходим от задач нестрогого суммирования в единичных областях к задачам о нестрогом суммировании в семействах областей. Как мы увидим в третьей части диссертации, при анализе задач о нахождении кратчайших расписаний в качестве таких семейств выступают семейства замкнутых полупространств m-мерного пространства. Поэтому далее исследуется именно этот случай. Формулируется 6 оптимизационных задач о нахождении нестрогого суммирования векторов из исходного 0-семейства в оптимальном (согласно какому-то критерию) семействе полупространств. Задачи различаются критериями и ограничениями на выбор полупространств. В лемме 14.4 показывается, что одна из этих задач без потери точности сводится к другой с понижением размерности пространства на единицу. Далее переходим к решению сформулированных задач в двумерном пространстве. С использованием результатов из предыдущего раздела получаем эффективные алгоритмы приближенного решения этих задач с гарантированными оценками точности, причем показано, что две из полученных оценок являются неулучшаемыми.

В разделе 15 доказываются теорема о суммировании l_∞-семейства векторов в прямоугольной области m-мерного пространства, ограниченной единицей по одной координате и константами — по другим координатам[48], а также теоремы о суммировании векторов на плоскости внутри минимального угла и в 3-мерном пространстве — внутри октанта[49].

Во второй части диссертации рассматриваются комбинаторные задачи различной природы. По большей части они носят вспомогательный характер и используются для решения задач из третьей части. Однако зачастую эти задачи представляют и самостоятельный интерес.

В разделе 17 вводится понятие *ранга* $r_m(x)$ целого числа $x \in \{0, 1, \ldots, m\}$. (С небольшой поправкой, — за исключением числа m, — ранг $r_m(x)$ равен номеру младшего бита, содержащего единицу в двоичной записи числа x.) Вводится отношение порядка на множестве чисел $\{0, 1, \ldots, m\}$ (сравнение чисел по рангу) и с помощью рекуррентных соотношений определяются функции $\delta_m(x)$, $\psi_m^+(x)$, $\psi_m^-(x)$ и число $\psi_{m'}$. Главным результатом раздела является лемма 17.10, в которой находится точное значение этого числа. (Результаты этого раздела используются при построении эффективного алгоритма решения задачи open shop в разделе 26.)

В разделе 18 рассматривается задача РРП о равномерном разбиении множества из n предметов на l подмножеств. При этом каждый предмет характеризуется m параметрами, и равномерность разбиения требуется по каждому из m параметров. Экономической интерпретацией этой задачи является известная *Задача объемно-календарного планирования* (ОКП). В задаче ОКП требуется годовой план предприятия, представленный n наименованиями, разбить по кварталам (или по месяцам) как можно равномернее

[48] Принадлежащий диссертанту результат из совместной статьи с В. Банащиком [122].

[49] Результаты из совместной статьи с С.В. Августиновичем [69].

по каждому из m параметров. В теореме 18.1 доказывается возможность эффективного приближенного решения задачи РРП с гарантированной оценкой точности при условии существования подобного алгоритма для задачи КСВ. С учетом известного алгоритма решения задачи КСВ (из раздела 11) получаем следствие 18.2, согласно которому задача $\text{РРП}_{m,s}(X,l)$ с n предметами и произвольной нормой s решается с трудоемкостью $O(n^2m^2)$ и гарантированной оценкой

$$\zeta_X^s(H) \le 2\left(m - 1 + \frac{1}{m}\right).$$

Однако последующий анализ, проведенный в теореме 18.3 и следствии 18.4, показывает, что более перспективным методом решения задачи РРП оказывается ее сведение к задаче о подмножестве векторов с заданной суммой (ПВЗС), рассматривавшейся нами в разделе 10, причем — к одному из двух ее эффективно разрешимых случаев, а именно, к задаче об отыскании вершины параллелепипеда, ближайшей к его заданной внутренней точке. Применяя лемму 10.8, получаем возможность решения задачи $\text{РРП}_{m,s}(X,l)$ с n предметами и произвольной нормой s с трудоемкостью $O(nlm^2)$ и гарантированной оценкой

$$\zeta_X^s(H) \le 1.00023m.$$

В этой оценке экзотический коэффициент перед m является точной (с точностью до последней значащей цифры) верхней границей значений функции $\nu(l)$, такой что задача о разбиении на l подмножеств решается с оценкой

$$\zeta_X^s(H) \le \nu(l)m$$

для любого l. При этом показывается, что максимум функции $\nu(l)$ достигается при $\nu = 909$. При $l \le 250$ коэффициент может быть заменен на 1.

В разделе 19 рассматривается задача РКЗ о равномерном распределении камней по кучам с ограничениями на множества допустимых куч, заданными для каждого камня. Для этой задачи строится эффективный алгоритм приближенного решения с абсолютной погрешностью, не превышающей вес максимального камня (теорема 19.2). Этот алгоритм основывается на решении задачи МПРС о нахождении в сети потока максимальной мощности, наиболее равномерного по стокам. Последняя задача решается точно (лемма 19.3), однако получаемое в ней решение порождает дробное распределение камней по кучам. При “округлении” этого распределения до целого и возникает обозначенная выше погрешность.

В последнем разделе второй части рассматривается несколько различных комбинаторных задач, таких как задача эквидистанции (о нахождении вершины единичного куба, наиболее одинаково удаленной от вершин из заданного семейства), задача о балансировке (для заданного семейства подмножеств конечного множества $\mathbb{N}_n$ требуется найти подмножество $N' \subset \mathbb{N}_n$, наиболее равномерно разделяющее каждое из подмножеств), задача Бека-Фиалы об округлении и задача нахождения функции Дворецкого. Показывается, что эти задачи близки к рассматриваемым нами в предыдущих разделах

задачам РРП, ПВЗС и КСВ. “Близость” между задачами позволяет строить эффективные алгоритмы приближенного решения одних задач при условии существования таких алгоритмов для других задач. Для каждой задачи определяется некоторая экстремальная функция, которая характеризует поведение оптимального решения в худшем случае. Устанавливаются соотношения между экстремальными функциями разных задач. В частности, доказывается оценка

$$\varphi_s(m) \geq \frac{1}{2} D_s(m),$$

где $D_s(m) \doteq \sup \min \left\| \sum_{i=1}^{p} \varepsilon_i x_i \right\|_s$ — функция Дворецкого (sup берется по всем конечным семействам векторов $\{x_i\} \subset \mathbb{R}^m$, $\|x_i\|_s \leq 1$, а min — по всем наборам из p чисел $\{\varepsilon_i = \pm 1\}$).

Третья, наибольшая по объему часть диссертации посвящена исследованию многостадийных систем теории расписаний. В четырех главах рассматривается 4 типа обслуживающих систем: системы с нефиксированными маршрутами, с одинаковыми маршрутами, сборочная линия и системы с различными маршрутами работ по машинам.

В первой главе рассматриваются системы с нефиксированными маршрутами. В разделе 23 вводится понятие *нормального* примера, т.е. примера, длина оптимального расписания которого совпадает с максимальной машинной нагрузкой. Для каждого примера определяется *вектор разностей* нагрузок машин (*VOD*). Вводится понятие *нормализующего* вектора, т.е. такого вектора $\Delta \in \mathbb{R}^m$, что любой m-машинный пример с $VOD = \Delta$ является нормальным. Вектор Δ называется *эффективно нормализующим*, если он нормализующий, и класс примеров с $VOD = \Delta$ является эффективно разрешимым. Показывается, что вектор $(0, 1)$ является единственным минимальным нормализующим и единственным минимальным эффективно нормализующим вектором в $\mathbb{R}^2$. В разделе 24 доказывается, что вектор $(0, 0, 2)$ является единственным минимальным нормализующим и единственным минимальным эффективно нормализующим вектором в $\mathbb{R}^3$ (результат из совместной статьи с Кононовым и Черных [95]).

В следующих двух разделах (25 и 26) эффективно нормальные классы примеров строятся на основе сравнения максимальной машинной нагрузки ($L_{\max}$) с длительностью максимальной операции ($p_{\max}$). При этом в разделе 25 оптимальное расписание строится с использованием алгоритма компактного суммирования векторов, а в разделе 26 — с помощью модифицированного алгоритма Фиалы. В обоих случаях показывается, что если выполняется соотношение вида

$$L_{\max} \geq \eta(m) p_{\max}$$

для определенной функции $\eta(m)$ от числа машин, то длина оптимального расписания равна $L_{\max}$, и это расписание строится эффективно. В первом случае функция $\eta(m)$ имеет порядок $O(m^2)$, а во втором — $O(m \log m)$, но при малых m проигрывает первой оценке из-за большой константы. По сравнению с аналогичным алгоритмом Фиалы модифицированный алгоритм из раздела 26 сужает примерно в 6 раз интервал неблагоприятных значений $L_{\max}$, а в благоприятном случае имеет на два порядка меньшую

трудоемкость. Для 3-машинной задачи open shop на основе результатов о нестрогом суммировании векторов из разделов 13, 14 строится алгоритм трудоемкости $O(n \log n)$, который отыскивает оптимальное решение для любого примера, удовлетворяющего соотношению $L_{\max} \geq 7p_{\max}$.

В связи с приведенными выше результатами естественно задать вопрос, при какой минимальной функции $\eta(m)$ соотношение $L_{\max} \geq \eta(m)p_{\max}$ гарантирует свойство нормальности примера. Результаты предыдущих разделов дают верхние оценки этой функции. Нижняя оценка сформулирована в теореме 27.1: $\eta(m) \geq 2m-2$. В частности, при $m = 3$ получаем соотношения $4 \leq \eta(3) \leq 7$.

В последующих разделах проводится анализ свойств плотных (жадных) расписаний, получаемых с использованием жадных алгоритмов. Из этих свойств вытекает теорема 30.2, в которой доказывается достаточное условие нормальности, являющееся комбинацией двух приводимых ранее типов условий, а именно: если разность в нагрузках первой и второй машины составляет не меньше $(m-1)p_{\max}$ (при нумерации машин по убыванию нагрузок) и при этом выполнено соотношение $L_{\max} \geq (5.45m-7)p_{\max}$, то пример является нормальным, а его оптимальное решение находится с помощью эффективного алгоритма.

Для задачи open shop известен простой результат Анны Рачманъ, согласно которому для любого примера может быть построено расписание с оценкой длины

$$C_{\max}(S) \leq L_{\max} + (m-1)p_{\max}.$$

В разделе 30 мы строим алгоритм той же трудоемкости ($O(nm^2)$), погрешность которого зависит от соотношения между $L_{\max}$ и $p_{\max}$. Одним из следствий этой теоремы является оценка

$$C_{\max}(S) \leq L_{\max} + \frac{m-1}{3}p_{\max},$$

которую гарантирует наш алгоритм при выполнении соотношения $L_{\max} \geq (7m-6)p_{\max}$.

В следующих разделах разрабатывается метод последовательной достройки нормального расписания. С помощью этого метода удается получить целую серию эффективно нормализующих векторов в $\mathbb{R}^m$ при $m > 3$ (см. утверждение 34.4, теорему 34.5, следствие 34.6). И хотя в пространстве $\mathbb{R}^4$ нам пока не известно ни одного минимального нормализующего (минимального эффективно нормализующего) вектора, мы можем утверждать, что в $\mathbb{R}^4$ имеется по меньшей мере два различных MN-вектора и два различных MEN-вектора (утверждение 34.7).

В разделе 35 для многопроцессорной задачи open shop (в которой имеется r цехов с идентичными параллельными машинами в каждом цехе, а каждая работа имеет r операций) строится полиномиальная аппроксимационная схема (PTAS) при условии, что общее по всем цехам число машин (m) ограничено константой[50]. Схема имеет линейную от числа работ (n) трудоемкость, причем коэффициент перед n полиномиален

[50] Результат получен совместно с Г. Вёгингером ([123], [9]). Насколько известно автору, построенная PTAS является второй аппроксимационной схемой в теории многостадийных систем. Первая такая схема была незадолго до этого построена Лэсли Холл для задачи flow shop [90].

от m и не зависит от задаваемой точности ε. (Экспоненциальная зависимость от m и ε "спрятана" в аддитивной константе.) В противоположность этому результату в теореме 36.2 доказано, что при нефиксированном числе машин для обычной задачи open shop и любой константы $\rho < 5/4$ не существует полиномиального ρ-приближающего алгоритма (при условии P≠NP), а следовательно, не существует и PTAS[51].

В разделе 37 рассматривается общая система открытого типа. От классической задачи open shop она отличается тем, что количество операций (r_j) каждой работы j является параметром, не зависящим от числа машин m, и тем, что множество допустимых исполнителей каждой операции может быть произвольным подмножеством множества $\{M_1, \ldots, M_m\}$. Доказывается теорема 37.1, согласно которой для открытой системы общего вида с m машинами и R операциями существует алгоритм трудоемкости $O(R^2m^2)$, строящий приближенное расписание S с оценкой точности $C_{\max}(S) - C_{\max}(S_{\text{opt}}) < r_{\max}p_{\max}$, где $r_{\max} = \max_j r_j$. При этом если все операции имеют одинаковую длину, то задача решается точно (теорема 37.2).

Вторая глава посвящена задаче flow shop. Показывается ее сведение к задаче о нестрогом суммировании векторов в соответствующем семействе полупространств. С использованием известных алгоритмов нестрогого суммирования из первой части получаем теорему 39.3, согласно которой 4-машинная задача flow shop может быть решена с трудоемкостью $O(n \log n)$ и гарантированной оценкой $C_{\max}(S_\pi) \leq L_{\max} + 6p_{\max}$, причем отыскиваемое алгоритмом расписание S_π является перестановочным. Для трех машин нестрогое суммирование дает алгоритм трудоемкости $O(n)$ с гарантированной неулучшаемой оценкой $C_{\max}(S_\pi) \leq L_{\max} + 3p_{\max}$ (теорема 39.4). В случае произвольного числа машин с использованием алгоритма компактного суммирования векторов строится алгоритм трудоемкости $O(n^2m^2)$, который для любого примера с m машинами и n работами находит перестановочное расписание с оценкой длины $C_{\max}(S_\pi) \leq L_{\max} + (m-1)\left(m - 2 + \frac{1}{m-2}\right) p_{\max}$.

Далее в разделе 40 выводятся оценки функции $\tilde{\mu}_{FS}(m)$, определяющей минимальный интервал локализации оптимумов по всем примерам m-машинной задачи flow shop.

В последнем разделе главы рассматривается многопроцессорная r-стадийная задача flow shop. В теореме 41.1 доказывается возможность ее эффективного решения с оценкой

$$C_{\max}(S) \leq L_{\max} + \left(r^2 - 3r + 3 + \frac{1}{r-2}\right) p_{\max},$$

не зависящей ни от числа работ, ни от числа машин.

В третьей главе рассматривается задача о сборочной линии. Система "сборочная линия" характеризуется тем, что в ней не действует запрет на одновременное выполнение операций, принадлежащих одной работе. По этой причине данная система не вписывается ни в одну из трех других рассматриваемых нами систем. Однако для задачи построения кратчайшего расписания в этой системе удается также показать ее сводимость к некоторой задаче о нестрогом суммировании векторов (теорема 41.2). Далее,

[51] Результат из совместной статьи с шестью соавторами, см. [136].

как обычно, возможно применение одного из разработанных нами алгоритмов. В частности, в случае трех машин это сведение позволяет построить алгоритм трудоемкости $O(n \log n)$, который для всякого примера задачи СЛ с тремя машинами и n работами строит перестановочное расписание S_π с неулучшаемой оценкой длины:

$$C_{\max}(S_\pi) \leq L_{\max} + 1.25 p_{\max}.$$

В последней главе рассматриваются наиболее общие (и наиболее сложные для анализа) многомаршрутные задачи типа job shop. Долгое время для них не было известно никаких приближенных алгоритмов с гарантированными оценками качества. Нашим главным результатом здесь является теорема 44.2, утверждающая, что для любого примера задачи job shop с m машинами, n работами и не более чем r операциями каждой работы его оптимум лежит в интервале $I_{JS} = [L_{\max}, L_{\max} + \psi p_{\max}]$, где $\psi = r(r-1)(mr+1)$. При этом существует алгоритм трудоемкости $O(n^2 m^2 r^2)$, который для любого примера строит приближенное расписание S со значением длины из интервала I_{JS}. Таким образом, при фиксированных m и r гарантируется эффективное построение расписания, длина которого отличается от оптимума не более чем на константное число величин $p_{\max}$, в то время как отношение оптимума к $p_{\max}$ имеет тенденцию бесконечного роста с ростом числа работ (n). Это обеспечивает *асимптотическую точность* алгоритма при фиксированных m и r и растущем n[52].

Результат теоремы 44.2 обобщается в теореме 44.3 на случай произвольного множества альтернативных исполнителей каждой операции и произвольного частичного порядка на множестве операций каждой работы. Более точно, в качестве интервала локализации оптимумов в этом случае берется интервал $[L^{\text{opt}}_{\max}, L^{\text{opt}}_{\max} + (\psi+1)p_{\max}]$, где $L^{\text{opt}}_{\max}$ — эффективно находимое значение оптимума в задаче о максимальном потоке, равномерном по стокам (из раздела 19).

Найденный нами в общем случае интервал локализации оптимумов может быть значительно уменьшен, если рассматриваются частные случаи задачи job shop. Так например, согласно теореме 42.2, для ациклической задачи job shop (известной также по фамилиям ее авторов — Шэлдона Акерса и Джойс Фридман [64]) в случае трех машин оптимумы всех примеров содержатся в интервале $[L_{\max}, L_{\max} + 5.5 p_{\max}]$, а приближенное решение из этого же интервала находится с трудоемкостью $O(n \log n)$. В разделе 43 рассматриваются двухмаршрутные задачи для трех машин. Как нетрудно убедиться, всего возможно три несводимых друг к другу конфигурации из двух маршрутов. В диссертации соответствующие им задачи обозначены через R231, R213 и R321. Ясно, что

[52] В строгой формулировке асимптотической точности алгоритма при $n \to \infty$ требуется, чтобы для любых $\varepsilon > 0$ и $\delta > 0$ существовало такое $n_0(\varepsilon, \delta)$, что

$$\mathbf{P}\{(C(S) - C(S_{\text{opt}}))/C(S_{\text{opt}}) > \varepsilon\} \ < \delta, \ \forall\, n \geq n_0(\varepsilon, \delta).$$

При этом если величины p^j_q (то бишь, длительности операций o^j_q) являются независимыми одинаково распределенными случайными величинами, то для доказательства асимптотической оптимальности нашего алгоритма достаточно потребовать выполнение одного необременительного условия: чтобы матожидание величин p^j_q было конечно и отлично от нуля.

двухмаршрутные задачи занимают промежуточное положение между (одномаршрутной) задачей flow shop и задачей Акерса-Фридман (в которой при $m = 3$ возможно одновременное присутствие до 6 различных маршрутов), поэтому естественно ожидать, что их интервалы будут промежуточными по длине. Это ожидание подтвердилось, причем длины всех трех найденных интервалов оказались разными: 5, 4 и 3, соответственно (в единицах величины $p_{\max}$). Наилучшего результата удалось достичь для задачи встречных маршрутов (R321): ее интервал ($[L_{\max}, L_{\max} + 3p_{\max}]$) совпадает с неулучшаемым интервалом одномаршрутной задачи, а следовательно, является неулучшаемым и для задачи R321[53].При отыскании интервалов для задач R231 и R213, а также трехмашинной задачи Акерса-Фридман использовались алгоритмы решения задач о нестрогом суммировании векторов из раздела 14.

В качестве вспомогательной при построении расписания в задаче R321 использовалась задача ДЗВМ (двухстадийная задача встречных маршрутов). В ней все работы первого маршрута имеют операции с ненулевыми длительностями лишь на машинах 1 и 2, а все работы второго маршрута — на машинах 3 и 2 (выполняемые в указанном порядке). В теореме 43.1 доказана NP-трудность задачи ДЗВМ в сильном смысле. Нетрудно видеть, что при перенумерации машин (при выборе второй машины в качестве третьей) задача становится эквивалентной двухстадийной задаче flow shop, причем такой, в которой только две из трех групп работ с различными маршрутами являются непустыми. Это усиливает результат, приведенный в монографии [61, стр. 42].

8 Основные понятия и определения

Дадим определения некоторых понятий, используемых в диссертации. Определения понятий из теории графов соответствуют стандарту, принятому в [62]. Определения понятий из выпуклого анализа даны по книге [29]. В употреблении терминологии теории расписаний будем в основном следовать монографии [61], хотя вместо базисных понятий “требования” и “приборы” нам привычнее будет говорить о “работах” и “машинах”.

Для обозначения конкретной задачи будем часто использовать аббревиатуру (например, ДЗВМ — вместо “двухстадийной задачи встречных маршрутов”) либо ее стандартное обозначение, принятое по негласной договоренности большинством “расписальщиков” мира — никто не обязывает, но все добровольно придерживаются. Эти стандарты в обозначениях задач были опубликованы в статье признанных авторитетов теории расписаний — Лолэ, Ленстры, Риннууй Кана и Шмойса [96]. (Например, стандартное обозначение для 3-машинной задачи Джонсона — $F3||C_{\max}$.)

Будем различать понятия *индивидуальной задачи* (т.е. задачи, в которой все входные данные определены) и *массовой задачи* (представляющей собой бесконечное семейство индивидуальных задач). Для краткости, массовую задачу будем называть просто “задачей” (например, “задачей Джонсона”, “задачей open shop”, и т.п.), а ее индивидуальную задачу — *примером* данной задачи[54].

[53]Результат опубликован в совместной работе с Ю. Неумытовым [26].

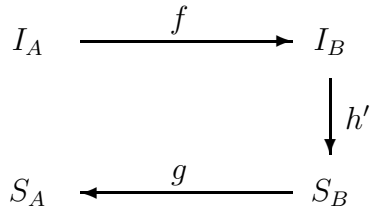


Рис. 8.1: Схема решения задачи A путем сведения ее к задаче B.

Каждая оптимизационная задача A будет характеризоваться пятеркой $(I_A, S_A, \mathrm{sol}_A, \theta_A, \mathrm{type}_A)$, где

- I_A — множество входов задачи A;
- S_A — множество ее решений для всевозможных входов $I \in I_A$;
- sol_A — функция, которая каждому входу $I \in I_A$ ставит в соответствие *множество* $\mathrm{sol}_A(I) \subseteq S_A$ *допустимых решений* примера, соответствующего входу I;
- $\theta_A : \ I_A \times S_A \to \mathbb{R}^+$ — неотрицательная *целевая функция*, определенная на множестве входов $I \in I_A$ и их решений $S \in \mathrm{sol}_A(I)$;
- $\mathrm{type}_A \in \{\max, \min\}$ — тип задачи (на максимум или минимум целевой функции).

Решением задачи A называется функция $h : \ I_A \to S_A$, устанавливающая такое соответствие между входами задачи A и ее решениями, что для любого входа $I \in I_A$ выполнено $h(I) \in \mathrm{sol}_A(I)$. Произвольное решение задачи A будем называть также *допустимым* или *приближенным*. Множество (допустимых) решений задачи A будем обозначать H_A. Решение $h^* \in H_A$ называется *точным* (или *оптимальным*), если

$$\theta(I, h^*(I)) = \mathrm{type}_A \ \theta_A(I, h(I)), \ \forall I \in I_A,$$

где $\min$ или $\max$ (в зависимости от значения type_A) берется по всем $h \in H_A$.

Для приближенного решения оптимизационных задач мы будем часто использовать их сведение к другим оптимизационным задачам. Сведение задачи A к задаче B задается парой отображений $(f; g)$ (см. рис. 8.1), где функция $f : \ I_A \to I_B$ любому входу I задачи A ставит в соответствие вход $I' = f(I)$ задачи B, а функция $g : \ S_B \to S_A$ отображает множество решений задачи B в множество решений задачи A так, что любое решение $S' \in \mathrm{sol}_B(f(I))$ отображается в решение $g(S') \in \mathrm{sol}_A(I)$.

Такая схема сведения гарантирует свойство, что при любом (допустимом) решении $h' \in H_B$ задачи B суперпозиция функций f, h' и g является (допустимым) решением задачи A.

[54] Автор не претендует на изобретение нового термина (которым уже пользуются расписальщики всего мира). Мы не отвергаем термина “индивидуальная задача”, но для краткости формулировок чаще будем пользоваться эквивалентным термином “пример”.

* * *

Множество натуральных чисел $\{1,\ldots,k\}$ будем обозначать через $\mathbb{N}_k$, множество $\{0,1,\ldots,k\}$ — через $\overline{\mathbb{N}}_k$, множество вещественных чисел — через $\mathbb{R}$, вероятность — через $\mathbf{P}$. Знак "$:=$" будем использовать для обозначения "присвоения нового значения какой-то переменной" (с помощью этого значка можно одной и той же переменной многократно переприсваивать различные значения), а знак "$\doteq$" — для присвоения какому-то объекту (функции, множеству, величине, и т.п.) какого-то значения "по определению". (Такое присвоение происходит только однажды.) Значком $\square$ будем отмечать конец доказательства какого-либо утверждения (леммы, теоремы, и т.п.) либо его формулировки — если доказательство отсутствует. Конец описания алгоритма будет отмечаться знаком $\blacksquare$.

Часть I

Задачи суммирования векторов

9 Определения и обозначения

Через $\mathbb{R}^m$ будем обозначать m-мерное пространство над полем действительных чисел. Если в $\mathbb{R}^m$ задана норма s, то через $B_{m,s}$ обозначим единичный шар этой нормы с центром в начале координат, т.е.

$$B_{m,s} = \{x \in \mathbb{R}^m \;\Big|\; \|x\|_s \le 1\}.$$

Определение 9.1 *Выпуклой оболочкой* $B(X)$ множества $X \subset \mathbb{R}^m$ называется минимальное выпуклое множество в $\mathbb{R}^m$, содержащее множество X.

Для конечного семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ вместо $B(X)$ будем также писать $\mathrm{conv}\{x_1, \ldots, x_n\}$ и использовать эквивалентное определение выпуклой оболочки:

$$\mathrm{conv}\{x_1, \ldots, x_n\} = \left\{ \lambda_1 x_1 + \ldots + \lambda_n x_n \,|\, \lambda_i \ge 0,\ i \in \mathbb{N}_n;\ \sum_i \lambda_i = 1 \right\}.$$

Определение 9.2 *Линейной оболочкой* $\mathrm{lin}(X)$ множества $X \subset \mathbb{R}^m$ называется минимальное линейное многообразие, содержащее множество X.

Для конечного семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ вместо $\mathrm{lin}(X)$ будем также писать $\mathrm{lin}\{x_1, \ldots, x_n\}$ и использовать эквивалентное определение линейной оболочки:

$$\mathrm{lin}\{x_1, \ldots, x_n\} = \left\{ \lambda_1 x_1 + \ldots + \lambda_n x_n \;\Big|\; \sum_i \lambda_i = 1 \right\}.$$

Скалярное произведение векторов $c = (c^1, \ldots, c^m),\ x = (x^1, \ldots, x^m)$, будем обозначать (c, x) и вычислять по формуле $(c, x) = c^1 x^1 + \cdots + c^m x^m$.

Сумму векторов семейства $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ будем обозначать через $\Sigma(X)$.

Семейство векторов с нулевой суммой будем называть *0-семейством*.

Далее $l(c) = \{tc \,|\, t \in R\}$ обозначает прямую, а $Q(c) = \{tc \,|\, t \in \mathbb{R}^+\}$ — луч в направлении вектора c.

Перестановку $\pi = (\pi_1, \ldots, \pi_n)$ чисел $\{1, 2, \ldots, n\}$ будем называть просто *перестановкой* $(\pi_1, \ldots, \pi_n)$.

Определение 9.3 Подмножество $K \subset \mathbb{R}^m$ называется *конусом*, если оно замкнуто относительно умножения на положительные числа, иначе говоря, если $\lambda x \in K$ для всех $x \in K,\ \lambda > 0$. *Выпуклый конус* — это конус, являющийся выпуклым множеством.

Определение 9.4 Конус $C \subseteq \mathbb{R}^m$ назовем *объемным*, если существуют вектор $b \in \mathbb{R}^m$ и число $\varepsilon > 0$ такие, что $b + \varepsilon B_{m,s} \subset C$.

Определение 9.5 Множество $X \subset \mathbb{R}^m$ называется *ограниченным*, если существует число $d > 0$ такое, что X содержится в шаре радиуса d: $X \subset dB_{m,s}$.

Утверждение 9.6 *Пусть C — объемный конус, X — ограниченное множество. Тогда существует вектор $a \in \mathbb{R}^m$ такой, что $a + X \subset C$.*

Доказательство. Из определений объемного конуса и ограниченного множества следует, что существуют вектор $b \in \mathbb{R}^m$ и числа $\varepsilon > 0,\ d > 0$ такие, что $b + \varepsilon B_{m,s} \subset C,\ X \subset dB_{m,s}$. Тогда по определению конуса имеем $C \supset \frac{d}{\varepsilon}(b + \varepsilon B_{m,s}) = \frac{d}{\varepsilon}b + dB_{m,s} \supset \frac{d}{\varepsilon}b + X$. □

Определение 9.7 Пусть C — непустое выпуклое множество в $\mathbb{R}^m$. Будем говорить, что множество C *удаляется в бесконечность* (или *неограничено*) в направлении y, где $y \in \mathbb{R}^m,\ y \neq 0$, если

$$x + \lambda y \in C,\ \forall \lambda \geq 0,\ \forall x \in C.$$

Множество всех векторов y, удовлетворяющих последнему соотношению, с присоединенным к нему началом координат будем называть *рецессивным конусом множества* C и обозначать через 0^+C.

Утверждение 9.8 *Пусть заданы вектор $c \in \mathbb{R}^m,\ c \neq 0$; l векторов $c_i \in \mathbb{R}^m$ таких, что $(c_i, c) < 0,\ i \in \mathbb{N}_l$; l чисел $\{\beta_1, \ldots, \beta_l\}$ и полиэдр $\Psi \subset \mathbb{R}^m$, определяемый линейной системой из l неравенств:*

$$\Psi = \{x \in \mathbb{R}^m \,|\, (c_i, x) \leq \beta_i,\ i \in \mathbb{N}_l\}.$$

Тогда Ψ — выпуклое неограниченное множество, имеющее объемный рецессивный конус $0^+\Psi$.

Доказательство. Определим $\alpha = \max_i \frac{-\|c_i\|}{(c,c_i)}$ и покажем, что для любого числа $\gamma \in \mathbb{R}^+$ и любого вектора $z \in B_{m,s}$ множество Ψ неограничено в направлении вектора $\gamma(\alpha c + z)$. Отсюда будет следовать, во-первых, неограниченность множества Ψ, и во-вторых, объемность его рецессивного конуса $0^+\Psi$.

Действительно, для любой точки $x \in \Psi$ и вектора c_i $(i \in \mathbb{N}_l)$ имеем

$$(x + \gamma(\alpha c + z), c_i) = (x, c_i) + \gamma(\alpha(c, c_i) + (z, c_i)) \leq \beta_i + \gamma(\|c_i\| + \alpha(c, c_i)) \leq \beta_i,$$

следовательно, $x + \gamma(\alpha c + z) \in \Psi,\ \forall\, \gamma \in \mathbb{R}^+$, откуда $\alpha c + z \in 0^+\Psi,\ \forall z \in B_{m,s}$. □

10 Алгоритмы нахождения подсемейств векторов, ребер и операций

В этом разделе мы определим понятия базы семейства векторов и базы семейства ребер графа и приведем два результата, касающихся алгоритмов нахождения таких баз. Далее рассматриваются оптимизационные задачи о нахождении подсемейства векторов с заданной суммой и о нахождении подсемейства операций. Приводятся алгоритмы их приближенного решения. Результаты раздела используются при построении алгоритмов в разделах 11, 18 и 26.

Определение 10.1 Множество векторов $X_B \subset \mathbb{R}^m$ называется *базой* семейства векторов $X \subset \mathbb{R}^m$, если $X_B \subseteq X$, множество X_B линейно независимо и $\mathrm{lin}(X_B) \supset X$.

Лемма 10.2 *Пусть заданы семейство векторов* $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ *и семейство чисел* $\{\lambda_j \in [0,1] \mid j \in \mathbb{N}_n\}$; $\sum_{j=1}^n \lambda_j x_j = x$. *Тогда с трудоемкостью* $O(nm^2)$ *можно найти базу* X_B *семейства* X *и семейство чисел* $\{\lambda'_j \in [0,1] \mid j \in \mathbb{N}_n\}$, *такие что*

$$\left(\sum_{j=1}^n \lambda'_j x_j = x\right) \ \& \ \left(\{x_j \mid \lambda'_j \in (0,1)\} \subseteq X_B\right).$$

Доказательство. Нетрудно убедиться в существовании искомой базы векторов и семейства чисел. Действительно, если на каком-то шаге алгоритма семейство векторов $\{x_j \mid \lambda_j \in (0,1)\}$ линейно зависимо, то мы найдем эту зависимость и с ее помощью изменим коэффициенты $\{\lambda_j \in (0,1)\}$ с сохранением суммы $\sum \lambda_j x_j$ так, что хотя бы один из коэффициентов обратится в 0 либо 1. Повторив эту процедуру не более n раз, мы придем к линейно независимому семейству векторов $\{x_j \mid \lambda_j \in (0,1)\}$, которое, очевидно, содержится в некоторой базе семейства $\{x_j\}$. Далее описывается алгоритм $\mathcal{A}_{10.1}$, реализующий эту идею с трудоемкостью $O(nm^2)$.

Алгоритм $\mathcal{A}_{10.1}$ состоит из 10 шагов. На шаге 1 происходит инициализация исходных параметров, а затем выполняется n итераций цикла по шагам 2–10. В конце каждой итерации $t = 0, \ldots, n$ (где под нулевой итерацией мы понимаем шаг 1) известны коэффициенты $\{\lambda'_j \in [0,1] \mid j \in \mathbb{N}_n\}$ и разбиение множества индексов $\mathbb{N}_n$ на три подмножества $\mathbb{N}_n = J_Z \cup J_W \cup J_R$, такие что

$$\sum_{j \in \mathbb{N}_n} \lambda'_j x_j = x; \tag{10.1}$$

$$\lambda'_j = \lambda_j, \ \forall j \in J_R = \{t+1, \ldots, n\};$$

$$\lambda'_j \in \{0,1\}, \forall j \in J_Z;$$

$$\{x_j \mid j \in J_W\} \doteq X_B — \text{ база семейства} \{x_j \mid j \in J_W \cup J_Z\}$$

Кроме того, известны:

— базис $B = \{b_1, \ldots, b_s\}$ пространства $\mathrm{lin}\{x_1, \ldots, x_t\}$ и первые s компонент перестановки $\pi = (\pi_1, \ldots, \pi_m)$ координат пространства $\mathbb{R}^m$, такие что

$$b_j(\pi_i) = 0, \ i = 1, \ldots, j-1,$$

$$b_j(\pi_j) \neq 0, \ j \in \mathbb{N}_s,$$

где $s = |J_W|$, $J_W = \{j_1, \ldots, j_s\}$;

— матрица $(c_{ij})_{i,j \in \mathbb{N}_s}$ перехода от базы X_B к базису B:

$$b_j = \sum_{i=1}^s c_{ji} x_{j_i}, \ j \in \mathbb{N}_s. \tag{10.2}$$

Таким образом, вначале все коэффициенты λ'_j совпадают с λ_j. На последующих итерациях t множество индексов $\{j \in J_R\}$ соответствует тем коэффициентам λ'_j, которые *еще* не преобразовывались, а множество J_Z — тем, которые *уже* не будут дальше преобразовываться (они целочисленны). Коэффициенты же $\{\lambda_j \mid j \in J_W\}$ находятся "в работе".

Алгоритм $\mathcal{A}_{10.1}$

1. $J_R := \mathbb{N}_n;\ J_Z := J_W := \emptyset;\ \lambda'_j := \lambda_j,\ \forall j \in \mathbb{N}_n;\ t := 1.$
   Очевидно, (10.1) выполняется.

2. Если $t > n$, то **stop**.

3. Берем индекс $t \in J_R :\ J_R := J_R \setminus \{t\}.$
   Если $x_t = 0$, то $\lambda'_t := 0;\ J_Z := J_Z \cup \{t\};$ **на** 10
   иначе $J_W := J_W \cup \{t\};\ j_{s+1} := t.$

4. Выясняем, зависимо ли множество векторов X_B. Для этого полагаем $b := x_t$, и для $\nu = 1, \ldots, s$ выполняем
$$\{\mu_\nu := b(\pi_\nu)/b_\nu(\pi_\nu);\ b := b - \mu_\nu b_\nu\}.$$
   В результате получаем вектор $b = x_t - \sum_{\nu=1}^{s} \mu_\nu b_\nu$, для которого $b(\pi_\nu) = 0,\ \forall \nu \in \mathbb{N}_s$. Вычислим величины $\mu'_\nu = \sum_{i=1}^{s} \mu_i c_{i\nu},\ \nu \in \mathbb{N}_s;\ \mu'_{s+1} := -1$. Тогда
$$b = -\sum_{\nu=1}^{s+1} \mu'_\nu x_{j_\nu}. \tag{10.3}$$

5. Если $b = 0$, т.е. множество X_B зависимо, то **на** 7 иначе **на** 6.

6. Находим координату $\nu^* \in \mathbb{N}_m \setminus \{\pi_i \mid i \in \mathbb{N}_s\}$, для которой $b(\nu^*) \neq 0$. Положим $\pi_{s+1} := \nu^*$. Добавляем вектор b в базис: $b_{s+1} := b$, — и дополняем матрицу (c_{ij}) коэффициентами:
$$c_{s+1,\nu} := -\mu'_\nu,\ \nu \in \mathbb{N}_{s+1};$$
$$c_{\nu,s+1} := 0,\ \nu \in \mathbb{N}_s;$$
$$s := s + 1;\ \textbf{на}\ 10.$$

7. Находим максимальное $\varepsilon = \varepsilon^*$ такое, что $\lambda'_{j_\nu} + \varepsilon\mu'_\nu \in [0,1],\ \nu \in \mathbb{N}_{s+1}$ (такое ε^* существует, так как $N \doteq \{\nu \mid \mu'_\nu \neq 0\} \neq \emptyset$), и пересчитываем коэффициенты $\{\lambda'_j \mid j \in J_W\}$ по формуле

$$\lambda'_{j_\nu} := \lambda'_{j_\nu} + \varepsilon^*\mu'_\nu,\ \nu \in \mathbb{N}_{s+1}.$$

При этом сумма $\sum_{j\in\mathbb{N}_n} \lambda'_j x_j$ увеличивается на $\varepsilon^* \sum_{\nu=1}^{s+1} \mu'_\nu x_{j_\nu} = 0$, т.е. остается неизменной.

8. Находим $\nu^* \in N$ такое, что $\lambda'_{j_{\nu^*}} \in \{0,1\}$, и переносим индекс j_{ν^*} из J_W в J_Z:

$$J_W := J_W \setminus \{j_{\nu^*}\};\ J_Z := J_Z \cup \{j_{\nu^*}\}.$$

Если $\nu^* = s+1$, то **на** 10 иначе **на** 9.

9. В формуле (10.2) пересчитаем коэффициенты (c_{ji}) с учетом замены вектора базы $x_{j_{\nu^*}}$ на вектор x_t. (Базис $\{b_1,\dots,b_s\}$ остается неизменным, так как $\{x_{j_1},\dots,x_{j_{s+1}}\}$ зависимо.) Из (10.2) и (10.3) получаем:

$$\begin{aligned} b_j &= \sum_{\nu\notin\{\nu^*,s+1\}} c_{j\nu}x_{j_\nu} - \frac{c_{j\nu^*}}{\mu'_{\nu^*}} \sum_{\nu\neq\nu^*} \mu'_\nu x_{j_\nu} = \\ &= \sum_{\nu\notin\{\nu^*,s+1\}} \left(c_{j\nu}x_{j_\nu} - \frac{c_{j\nu^*}\mu'_\nu}{\mu'_{\nu^*}}\right) x_{j_\nu} + \frac{c_{j\nu^*}}{\mu'_{\nu^*}} x_t. \end{aligned}$$

Переставим индекс t с $(s+1)$-го места в J_W на освободившееся место j_{ν^*} :

$$\begin{aligned} &c_{j\nu^*} := c_{j\nu^*}/\mu'_{\nu^*},\ j \in \mathbb{N}_s; \\ &c_{j\nu} := c_{j\nu} - c_{j\nu^*}\mu'_\nu,\ j \in \mathbb{N}_s;\ \nu \in \mathbb{N}_s \setminus \{\nu^*\}; \\ &j_{\nu^*} := t. \end{aligned}$$

10. $t := t+1$; **на** 2.

Алгоритм $\mathcal{A}_{10.1}$ описан. ■

Легко видеть, что свойство (10.1) сохраняется на каждой итерации алгоритма. Кроме того, получаемая после последней итерации база $\{x_j \mid j \in J_W\}$ является базой всего семейства X, и для всех $j \in \mathbb{N}_n \setminus J_W = J_Z$ выполнено $\lambda'_j \in \{0,1\}$. Таким образом, полученные в результате семейство чисел $\{\lambda'_j\}$ и база X_B удовлетворяют всем требованиям леммы.

Для завершения доказательства леммы 10.2 остается заметить, что трудоемкость каждой из n итераций алгоритма $\mathcal{A}_{10.1}$ не превосходит $O(m^2)$. □

Определение 10.3 Пусть $G = (V, E)$ — обыкновенный неориентированный граф. Будем говорить, что подмножество ребер $E' \subseteq E$ *независимо*, если каждая компонента связности подграфа $G' = (V, E')$ содержит не более одного цикла (или, что то же, количество ребер в ней не превосходит количества ее вершин). В противном случае E' будем называть *зависимым*. Подмножество $E' \subseteq E$ будем называть *базой* множества E, если оно независимо, а добавление к нему любого ребра $e \in E \setminus E'$ делает его зависимым.

Пусть $G = (V, E)$, $|V| = \bar{m}$. Для каждого ребра $e = (v_i, v_j)$ определим две неотрицательные величины: $w(e, v_i), w(e, v_j)$, которые будем называть *весами* ребра e относительно его концевых вершин. Будем также считать, что

$$w(e, v) = 0, \text{ если } e \text{ неинцидентно } v. \tag{10.4}$$

Таким образом, w является неотрицательной числовой функцией на множестве $E \times V$. Будем рассматривать $\bar{m}$-мерное векторное пространство, каждая координата $\nu \in \mathbb{N}_{\bar{m}}$ которого соответствует вершине v_ν графа G, а каждому ребру $e = (v_i, v_j)$ поставлен в соответствие вектор $w_e = (w(e, v_1), \ldots, w(e, v_{\bar{m}})) \in \mathbb{R}^{\bar{m}}$.

Определение 10.4 Подмножество ребер $E' \subseteq E$ будем называть w-*зависимым* (w-*независимым*), если соответствующее ему семейство векторов $W(E') \doteq \{w_e \,|\, e \in E'\}$ линейно зависимо (независимо).

Лемма 10.5 *Подмножество $E' \subseteq E$ зависимо если и только если оно w-зависимо для любой весовой функции $w: \; E \times V \to \mathbb{R}^+$, удовлетворяющей (10.4).*

Доказательство. $\Longrightarrow$ Пусть подмножество $E' \subseteq E$ зависимо, и задана функция $w: \; E \times V \to \mathbb{R}^+$, удовлетворяющая (10.4). Тогда для некоторой компоненты связности $G'' = (V'', E'')$ графа $G' = (V, E')$ выполнено $|E''| > |V''|$. Поскольку все координаты $w(e, v_j)$ векторов $\{w_e \,|\, e \in E''\} = W(E'')$, соответствующие вершинам $v_j \notin V''$, равны нулю, то все векторы семейства $W(E'')$ находятся в $|V''|$-мерном пространстве. Так как количество векторов ($|E''|$) больше их размерности, то семейство векторов $W(E'')$ линейно зависимо, или согласно определению 10.4, множество ребер E'' (а значит, и множество E') w-зависимо.

$\Longleftarrow$ Пусть подмножество ребер $E' \subseteq E$ независимо. Определим функцию $w(e, v_\nu)$ равной 1 для каждого ребра $e = (v_i, v_j) \in E'$ и инцидентных ему вершин $v_\nu \in \{v_i, v_j\}$. Докажем, что множество E' w-независимо, для чего достаточно доказать w-независимость множества ребер E'' для каждой компоненты связности $G'' = (V'', E'')$ графа $G' = (V, E')$.

Вначале предположим, что G'' содержит цикл C, состоящий из k вершин. Перенумеруем вершины и ребра компоненты G'' следующим образом: сначала нумеруем числами $1, \ldots, k$ вершины и ребра цикла C (в порядке обхода цикла), а затем последовательно приписываем числа $k+1, \ldots, |E''|$ остальным ребрам и вершинам. При этом каждое следующее ($(k+j)$-е) ребро выбираем так, чтобы одна из его вершин была уже занумерована, а вторая — нет. (Приписываем ей номер $k+j$.) Тогда матрица W, составленная

из векторов-строк $w_{e_1}, \ldots, w_{e_{|E''|}}$, будет иметь вид, как показано на рис. 10.1, где A и B — квадратные матрицы (размера $k \times k$ и $(|E''|-k)\times(|E''|-k)$ соответственно). Матрица B имеет единицы по главной диагонали, и нули — выше главной диагонали. Содержание областей, помеченных буквой "Z", для нас несущественно. Нетрудно убедиться, что матрица W невырожденная. Следовательно, множество ребер E'' w-независимо.

$$W = \left(\begin{array}{c|c} A & \mathbb{O} \\ \hline Z & B \end{array}\right) \qquad A = \begin{pmatrix} 1\,1 & & \\ & 1\,1 & \mathbb{O} \\ & \bullet & \bullet \\ & \mathbb{O} & \bullet\ 1\,1 \\ 1 & & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & & \\ & 1 & \mathbb{O} \\ & \bullet & \bullet \\ & Z & \bullet\ 1 \\ & & 1 \end{pmatrix}$$

Рис. 10.1: Матрица W, составленная из векторов-строк $w_e,\ e \in E''$.

Остается рассмотреть случай, когда компонента G'' графа G' не содержит цикла. В этом случае матрица W имеет размер $|E''| \times (|E''|+1)$. Нумеруя ребра и вершины графа G'' как мы это делали в предыдущем случае для нециклических ребер и вершин, получим матрицу $W = (w_{ij})$, у которой $w_{i,i+1} = 1$ для всех $i \in \mathbb{N}_{|E''|}$ и $w_{ij} = 0$ для всех $j > i+1$. Ясно, что ранг этой матрицы равен $|E''|$, а следовательно, множество ребер E'' w-независимо. □

Сформулируем и докажем аналог леммы 10.2 применительно к семейству векторов $\{w_e \,|\, e \in E\}$, определенных для произвольного графа G.

Лемма 10.6 *Пусть задан граф $G = (V, E)$, где $V = \{1, \ldots, \bar{m}\}$ — множество вершин, $E = \{e_1, \ldots, e_{\bar{n}}\}$ — множество ребер; m — длина наибольшей простой цепи в G; задана функция $w : \ E \times V \to \mathbb{R}$, удовлетворяющая (10.4); $W(E) = \{w_e = (w(e, v_1), \ldots, w(e, v_{\bar{m}})) \,|\, e \in E\}$; заданы числа $\lambda_e \in [0,1],\ \forall\ e \in E;\ \sum \lambda_e w_e = x$. Тогда с трудоемкостью $O(\bar{n}m + \bar{m})$ можно найти базу $E' \subseteq E$ и семейство чисел $\{\lambda'_e \in [0,1] \,|\, e \in E\}$, такие что $(\sum \lambda'_e w_e = x)\&(\{e \in E \,|\, \lambda'_e \in (0,1)\} \subseteq E')$.*

Доказательство. Описываемая ниже общая схема алгоритма $\mathcal{A}_{10.2}$ нахождения базы $E' \subseteq E$ принципиально совпадает со схемой алгоритма $\mathcal{A}_{10.1}$ нахождения базы в лемме 10.2. Отличия между ними сводятся к замене (в п.5) понятия "зависимость семейства векторов" на понятие "зависимость множества ребер", а также к отличиям в используемых алгоритмах $\mathcal{A}^i$, что позволяет в случае леммы 10.6, используя специфику векторов $\{w_e\}$, значительно уменьшить трудоемкость алгоритма.

Алгоритм $\mathcal{A}_{10.2}$ состоит из 10 шагов. На шаге 1 происходит инициализация исходных данных, а затем выполняется $\bar{n}$ итераций цикла по шагам 2–10. В конце каждой итерации $t = 0, \ldots, \bar{n}$ (где под нулевой итерацией понимается шаг 1) известны коэффициенты $\{\lambda'_e \in [0,1] \,|\, e \in E\}$ и разбиение множества E на три подмножества $E = E^t_Z \cup E^t_W \cup E^t_R$, такие что:

$\sum \lambda'_e w_e = x;\ \lambda'_e = \lambda_e,\ \forall\ e \in E^t_R;\ |E^t_R| = \bar{n} - t;\ \lambda'_e \in \{0,1\},\ \forall\ e \in E^t_Z;$

E^t_W — база множества $E^t_W \cup E^t_Z$.

На каждой итерации t будет рассматриваться граф $G^t = (V, E^t_W)$. Для каждой компоненты связности G' графа G^t одну из ее вершин будем называть *корнем* и обозначать $v^*(G')$. Для каждой вершины $v \in G'$ будет известна первая вершина $p(v)$ простой цепи, ведущей из v в корень $v^*(G')$; $p(v) = 0$, если v — корневая вершина. Множество ребер $\{(v, p(v))\}$ по всем некорневым вершинам $v \in G'$ образует остовное дерево компоненты G'. При корневой вершине $v^*(G')$ каждой компоненты G' хранится список $\hat{E}(v^*)$ *внеостовных* ребер компоненты G' (т.е. ребер компоненты G', не входящих в ее остовное дерево); $\Delta(v^*) \doteq |\hat{E}(v^*)|$. (Под $\hat{E}(G')$ и $\Delta(G')$ далее будут пониматься $\hat{E}(v^*(G'))$ и $\Delta(v^*(G'))$.) В конце каждой итерации алгоритма $\mathcal{A}_{10.2}$ для каждой компоненты $G' \subseteq G^t$ будем иметь $\Delta(G') \leq 1$. В середине итерации возможна ситуация $\Delta(G') = 2$, когда количество ребер в компоненте G' превышает количество ее вершин, что означает зависимость компоненты G' и графа G^t. (В этом случае в алгоритме $\mathcal{A}_{10.2}$ переходим к п.6).

Алгоритм $\mathcal{A}_{10.2}$ (нахождение базы $E' \subseteq E$)

1. $E^0_R := E;\ E^0_Z := E^0_W := \varnothing;\ \lambda'_e := \lambda_e,\ \forall e \in E;$
   $p(v) := 0,\ \hat{E}(v) := \varnothing,\ \Delta(v) := 0\ \forall v \in V;\ t := 1.$
2. Если $t > \bar{n}$, то **stop**.
3. Выбираем произвольный элемент $e \in E^{t-1}_R$ и полагаем $E^t_R := E^{t-1}_R \setminus \{e\};\ E^t_W := E^{t-1}_W \cup \{e\};\ E^t_Z := E^{t-1}_Z$.
4. Преобразуем информацию о графе G^t (алгоритм $\mathcal{A}^1_{10.2}$).
5. Если множество E^t_W зависимо, то **на** 6 иначе **на** 10.
6. Находим (алгоритмом $\mathcal{A}^2_{10.2}$) минимальный связный двуциклический подграф G^* графа G^t. Находим (алгоритмом $\mathcal{A}^3_{10.2}$) нетривиальную зависимость $\sum_{e \in G^*} \mu_e w_e = 0$.
7. Пересчитываем коэффициенты $\{\lambda'_e \,|\, e \in G^*\}$ по формуле $\lambda'_e := \lambda'_e + \varepsilon\mu_e$, где ε — максимальное из чисел, оставляющих все коэффициенты $\{\lambda'_e\}$ в интервале $[0, 1]$.
8. Находим элемент $e^* \in G^* \subseteq E^t_W$, для которого выполнено $\lambda'_{e^*} \in \{0, 1\}$, и переводим его из E^t_W в E^t_Z.
9. Преобразуем информацию о графе G^t (алгоритм $\mathcal{A}^4_{10.2}$).
10. $t := t + 1$; **на** 2.

Алгоритм $\mathcal{A}_{10.2}$ описан. ■

Вместо графа G в алгоритме $\mathcal{A}_{10.2}$ нам достаточно будет рассматривать его подграф $G^t = (V, E^t_W)$, изменяемый от итерации к итерации. Докажем, что в конце каждой

итерации $t \in \overline{\mathbb{N}}_{\bar{n}}$ алгоритма $\mathcal{A}_{10.2}$ множество E_W^t независимо, т.е. граф G^t распадается на компоненты связности, в каждой из которых имеется не более одного цикла.

Пусть множество E_W^t было независимым после $(t-1)$-й итерации и стало зависимым после добавления ребра e в граф G^{t-1} на шаге 3 итерации t. Если ребро e соединяет две вершины, принадлежащие одной компоненте связности графа G^{t-1}, то в результате в G^t образуется компонента G', в которой число ребер на единицу больше числа вершин. В случае, когда e соединяет две различные компоненты графа G^{t-1}, каждая из этих компонент (в силу независимости графа G^{t-1}) содержит число ребер, не превосходящее числа вершин. Поскольку, однако, добавление e делает граф G^t зависимым, то вновь образуется компонента G', в которой число ребер на единицу больше числа вершин. Такая компонента содержит два цикла. При этом минимальный связный двуцикличе-ский подграф G^* компоненты G' может иметь одну из двух конфигураций (a или b), изображенных на рис. 10.2.

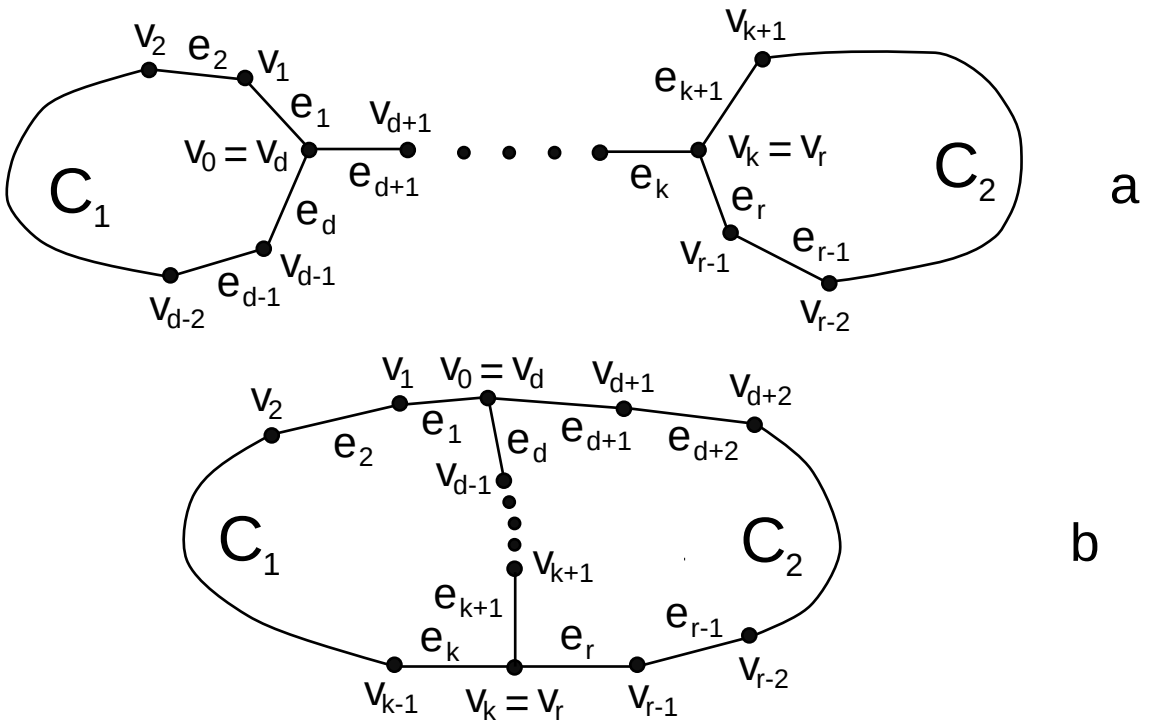


Рис. 10.2: Двуциклический подграф G^*

После удаления на шаге 8 из компоненты G' одного ребра ($e' \in G^*$) она либо остается связной (если e' было циклическим), либо распадается на две компоненты (если e' принадлежало "перешейку" $(v_k \dots v_d)$ в конфигурации a графа G^* — см. рис. 10.2). В первом случае число ребер компоненты G' становится равным числу ее вершин, а следовательно, G' (а вместе с ней — и граф G^t) становятся независимыми. Во втором

случае каждая из двух вновьобразующихся компонент содержит ровно один цикл, а следовательно, является независимой.

Теперь докажем, что в конце каждой итерации $t = 0, \ldots, \bar{n}$ множество ребер E_W^t является базой множества $E_Z^t \cup E_W^t$. (Отсюда будет следовать, что множество $E_W^{\bar{n}}$, вычисляемое по окончании последней итерации алгоритма $\mathcal{A}_{10.2}$, является искомой базой E' множества всех ребер E.) Для этого достаточно доказать, что добавление к E_W^t любого ребра $e \in E_Z^t$ делает множество $E_W^t \cup \{e\}$ зависимым.

Это верно на итерации $t = 0$, когда оба множества E_Z^t и E_W^t пусты. Пусть $t > 0$, и пусть $e = (v', v'')$ — произвольное ребро из E_Z^t. Ребро e было переведено из $E_W^{t'}$ в $E_Z^{t'}$ на шаге 8 итерации $t' \leq t$. Заметим, что если на какой-то итерации $\bar{t}$ вершина $\bar{v}$ принадлежит "циклической" компоненте графа $G^{\bar{t}}$ (т.е. компоненте, содержащей цикл), то на всех последующих итерациях это свойство сохраняется. Таким образом, в графе G^t вершины v' и v'' также принадлежат циклическим компонентам (G' и G''). Если при этом $G' = G''$, то добавление ребра e делает компоненту G' зависимой. Если же $G' \neq G''$, то при добавлении ребра e компоненты G' и G'' сливаются в одну компоненту, содержащую два цикла. В обоих случаях множество $E_W^t \cup \{e\}$ становится зависимым.

Для завершения доказательства леммы 10.6 остается убедиться, что трудоемкость каждой из $\bar{n}$ итераций алгоритма $\mathcal{A}_{10.2}$ не превосходит $O(m)$. Для этого необходимо более детально описать работу подалгоритмов $\mathcal{A}_{10.2}^i$, применяемых на различных шагах алгоритма $\mathcal{A}_{10.2}$.

Алгоритм $\mathcal{A}_{10.2}^1$ (преобразование информации о графе G^t при добавлении в него нового ребра $e = (v^1, v^2)$)

Для каждой из двух концевых вершин v^i нового ребра, используя функцию $p(v)$, найдем цепь $P_i = (v_0^i v_1^i \ldots v_{s_i}^i)$, соединяющую ее с корнем ребрами остовного дерева, где $v_0^i = v^i,\ v_{j+1}^i = p(v_j^i),\ p(v_{s_i}^i) = 0$. Если корни $v_{s_1}^1, v_{s_2}^2$ совпадают, т.е. вершины v^1, v^2 принадлежат одной компоненте G', то ребро e объявляем внеостовным, добавляем его в список $\hat{E}(G')$, $\Delta(G')$ увеличиваем на 1. Если же корни различны, т.е. ребро e соединяет две различные компоненты — G_1 и G_2 (как на рис. 10.3), то берем корень $v^*(G_1)$ в качестве корня новой компоненты (G'), для чего переопределяем функцию $p(v)$ для вершин цепи P_2, полагая $p(v_{i+1}^2) := v_i^2,\ i \in \overline{\mathbb{N}}_{s_2-1};\ p(v^2) := v^1$. Для остальных вершин $v \in G_2$ функция $p(v)$ не изменяется. Полагаем $\hat{E}(G') := \hat{E}(G_1) \cup \hat{E}(G_2);\ \Delta(G') := \Delta(G_1) + \Delta(G_2)$. ■

Алгоритм $\mathcal{A}_{10.2}^2$ (нахождение минимального связного двуциклического подграфа $G^* \subseteq G'$)

Алгоритм $\mathcal{A}_{10.2}^2$ приступает к работе, как только из алгоритма $\mathcal{A}_{10.2}^1$ поступает информация о том, что в графе G^t образовалась компонента связности G' с $\Delta(G') = 2$. Как было сказано выше, минимальный связный двуциклический подграф G^* компоненты G' имеет одну из двух конфигураций (a или b), изображенных на рис. 10.2. Результатом работы алгоритма $\mathcal{A}_{10.2}^2$ является цепь $P^* = (v_0 e_1 v_1 e_2 \ldots e_r v_r)$ (где $v_0 = v_d,\ v_r = v_k$), содержащая все ребра графа G^* (согласно определению цепи по Харари [62], все ее ребра различны).

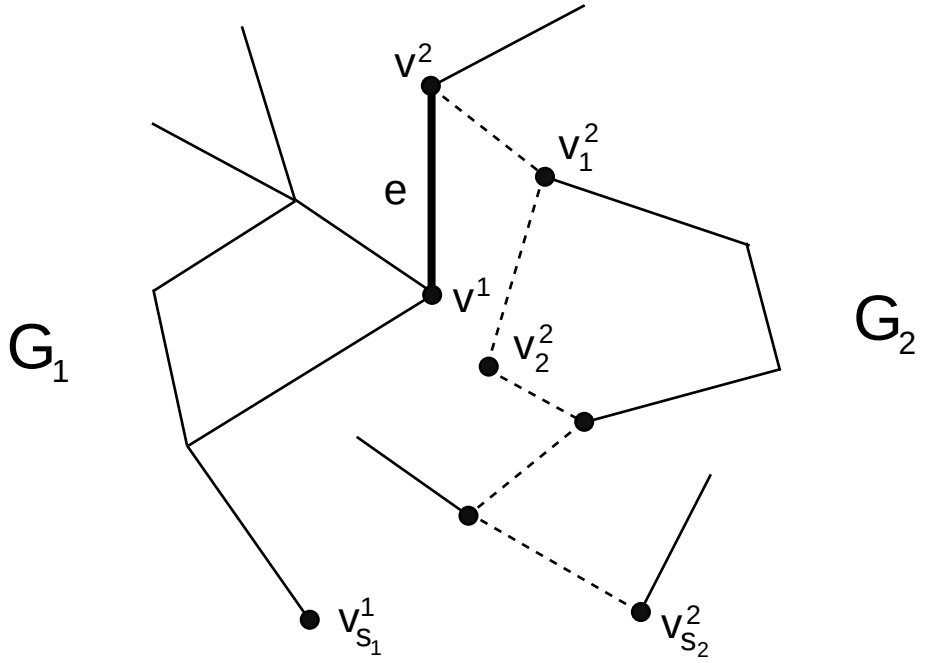


Рис. 10.3: Алгоритм $\mathcal{A}^1_{10.2}$: ребро e соединяет компоненты G_1 и G_2

В начале работы алгоритма нам известны два внеостовных ребра $\{e_1^0, e_2^0\}$ графа G' и значения функции $p(v)\ \forall\ v \in G'$. Для каждой из четырех концевых вершин $v_i^0,\ i \in \mathbb{N}_4$, ребер e_1^0, e_2^0 с помощью функции p найдем простую цепь $P(v_i^0) = (v_i^0 v_i^1 \dots v_i^{s_i})$ из вершины v_i^0 в корень v^*, где $p(v_i^j) = v_i^{j+1}\ \forall\ j < s_i;\ v_i^{s_i} = v^*,\ i \in \mathbb{N}_4;\ p(v^*) = 0$. Пройдем по каждой из четырех цепей в обратном направлении, начиная с корня v^*, и найдем общий для всех цепей участок, а также вершины v_i' $(i = 1, 2, 3)$, начиная с которых цепи P_i расходятся друг от друга. С точностью до нумерации цепей существует два варианта (c и d) расположения вершин $\{v_i'\}$ (см. рис. 10.4; возможны случаи: $v_2' = v_3'$ — в варианте c, а также $v_1' = v_2'$ или $v_1' = v_2' = v_3'$ — в обоих вариантах).

В обоих вариантах граф G^* образуют ребра множества $\{e_1^0, e_2^0\} \cup_{i=1}^4 E(v_i^0) \setminus \cap_{i=1}^4 E(v_i^0)$, где $E(v_i^0)$ — множество ребер цепи $P(v_i^0)$. При этом возможны две топологически различных конфигурации. Если ребра e_1^0, e_2^0 образуются парами вершин $(v_1^0, v_2^0), (v_3^0, v_4^0)$, то имеем конфигурацию a графа G^* (рис. 10.2). В этом случае цепь P^*, проходящая по всем ребрам графа G^* и используемая в алгоритме $\mathcal{A}^3_{10.2}$ для нумерации ребер и вершин, образуется конкатенацией ребер e_1^0, e_2^0 и подцепей цепей $\{P(v_i^0)\}$:

$P^* = (v_2' \dots v_1' \dots v_1^0 v_2^0 \dots v_2' \dots v_3' \dots v_3^0 v_4^0 \dots v_3')$ — в варианте c,
$P^* = (v_2' \dots v_1^0 v_2^0 \dots v_2' \dots v_1' \dots v_3' \dots v_3^0 v_4^0 \dots v_3')$ — в варианте d.

Если же ребра e_1^0, e_2^0 образуются парами $(v_1^0, v_4^0), (v_2^0, v_3^0)$, то имеем конфигурацию b (рис. 10.2), а цепь P^* имеет вид:

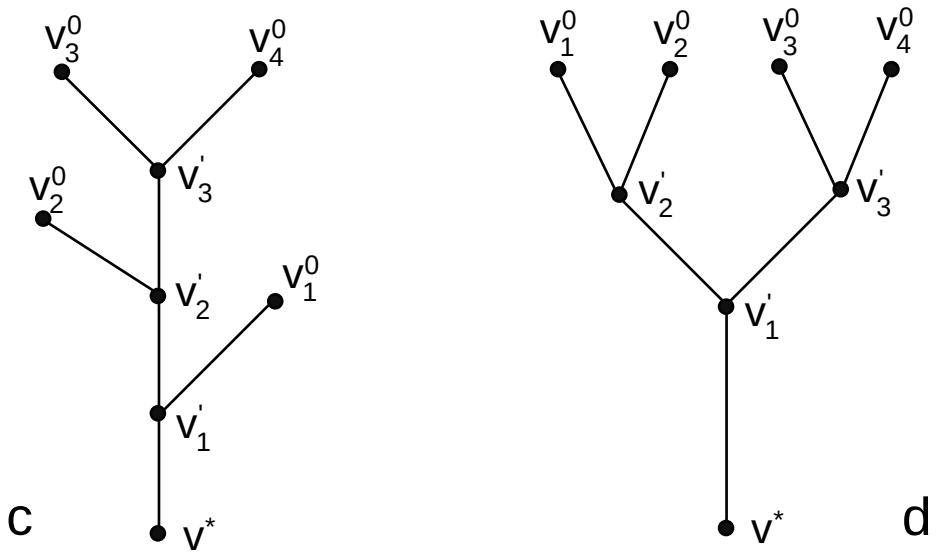


Рис. 10.4: Подграф графа G', составленный из цепей P_i.

$P^* = (v'_2 \dots v^0_2 v^0_3 \dots v'_3 \dots v'_2 \dots v'_1 \dots v^0_1 v^0_4 \dots v'_3)$ — в варианте c,
$P^* = (v'_2 \dots v^0_2 v^0_3 \dots v'_3 \dots v'_1 \dots v'_2 \dots v^0_1 v^0_4 \dots v'_3)$ — в варианте d.

Случай ребер $(v^0_1, v^0_3), (v^0_2, v^0_4)$ получается из предыдущего перенумерацией вершин: $v^0_3 \leftrightarrow v^0_4$. ■

Алгоритм $\mathcal{A}^3_{10.2}$ (нахождение нетривиальной зависимости векторов множества $W(E^t_W)$)

Будем считать, что вершины и ребра графа G^* занумерованы в порядке прохождения их цепью P^*. В случае $k \geq d$ имеем конфигурацию a, а в случае $k \leq d$ — конфигурацию b (рис. 10.2; при $k = d$, очевидно, конфигурации a и b совпадают). Через C_1 и C_2 обозначим циклы $(v_0 v_1 \dots v_{d-1} v_d)$ и $(v_k v_{k+1} \dots v_r)$ соответственно.

Зависимость между векторами $\{w_e \,|\, e \in G^*\}$ в терминах весов $w(e, v)$ выражается системой из $r - 1$ уравнений с r неизвестными $\{y_i \,|\, i \in \mathbb{N}_r\}$ (эти уравнения называем уравнениями “баланса” в вершине v_i):

$$w(e_i, v_i)y_i + w(e_{i+1}, v_i)y_{i+1} = 0, \;\; i \in \mathbb{N}_{r-1} \setminus \{d, k\}, \tag{10.5}$$

$$w(e_d, v_d)y_d + w(e_{d+1}, v_d)y_{d+1} + w(e_1, v_d)y_1 = 0, \tag{10.6}$$

$$w(e_k, v_k)y_k + w(e_{k+1}, v_k)y_{k+1} + w(e_r, v_k)y_r = 0. \tag{10.7}$$

В случае $k = d$ уравнения (10.6) и (10.7) заменяются на:

$$w(e_d, v_d)y_d + w(e_{d+1}, v_d)y_{d+1} + w(e_1, v_d)y_1 + w(e_r, v_d)y_r = 0. \tag{10.8}$$

Заметим, что если системе (10.5)–(10.7) оставить лишь уравнения и переменные, соответствующие вершинам и ребрам цикла C_1, то дискриминант этой подсистемы легко вычисляется по формуле

$$D = \prod_{\nu=1}^{d} w(e_\nu, v_\nu) - \prod_{\nu=1}^{d} w(e_\nu, v_{\nu-1}) \quad (\text{где } v_0 = v_d).$$

Множество ребер цикла C_1 w-зависимо если и только если $D = 0$. Следовательно, w-зависимость цикла C_1 (и аналогично — цикла C_2) устанавливается за время $O(r)$.

Цепь $C = (v_{i_0} e_{j_1} v_{i_1} e_{j_2} \dots e_{j_s} v_{i_s}) \subseteq G^*$ назовем *особой*, если $w(e_{j_1}, v_{i_0}) = w(e_{j_s}, v_{i_s}) = 0$, $w(e_{j_\nu}, v_{i_\nu}) \neq 0,\ w(e_{j_{\nu+1}}, v_{i_\nu}) \neq 0,\ \nu \in \mathbb{N}_{s-1}$; и ни одна из вершин цепи (кроме, быть может, концевых) не проходится цепью более одного раза.

Ясно, что всякая особая цепь C является w-зависимой. При этом коэффициенты $\{y_\nu\}$ нетривиальной w-зависимости ее ребер ($\sum_{e_\nu \in C} y_\nu w_{e_\nu} = 0$) находятся с помощью следующей простой **процедуры** $\mathcal{P}$, имеющей трудоемкость $O(r)$:

значение $y_{j_1} \neq 0$ берем произвольным; затем последовательно вычисляем значения $y_{j_\nu},\ \nu = 2, \dots, s$, из уравнений баланса в вершинах $v_{i_{\nu-1}}$.

(В случае особой цепи уравнения баланса имеют вид

$$w(e_{j_\nu}, v_{i_{\nu-1}}) y_{j_\nu} + w(e_{j_{\nu-1}}, v_{i_{\nu-1}}) y_{j_{\nu-1}} = 0, \tag{10.9}$$

т.е. связывают лишь две переменные.)

Цепь C назовем *полуособой*, если $w(e_{j_s}, v_{i_s}) = 0$, остальные веса ребер, входящих в цепь C, ненулевые, и начальный отрезок $C' = (v_{i_0} e_{j_1} v_{i_1} \dots e_{j_t} v_{i_t})$ цепи C является w-независимым циклом (т.е. $v_{i_t} = v_{i_0}$).

Определим функцию $W(C')$ от ребер цикла C', вычисляемую по формуле

$$W(C') = 1 + (-1)^{t-1} \prod_{\nu=1}^{t} \frac{w(e_{j_\nu}, v_{i_\nu})}{w(e_{j_\nu}, v_{i_{\nu-1}})}.$$

Функция $W(C')$ определена корректно, если ни один из коэффициентов $w(e_{j_\nu}, v_{i_{\nu-1}})$, $\nu \in \mathbb{N}_t$, не равен нулю. (Заметим, что значение функции $W(C')$ зависит от направления обхода цикла C'.)

Легко заметить, что полуособая цепь также является w-зависимой. Для нахождения коэффициентов зависимости ребер полуособой цепи C' применяется описанная выше процедура $\mathcal{P}$. (Хотя уравнение баланса в вершине v_{i_t} связывает три переменные y_{j_ν} (для $\nu \in \{1, t, t+1\}$), к моменту вычисления переменной $y_{j_{t+1}}$ две другие переменные уже известны.)

Опишем алгоритм $\mathcal{A}^3_{10.2}$ нахождения нетривиального (ненулевого) решения системы (10.5)–(10.7) за $O(r)$ операций. Он состоит из четырех шагов.

1. За время $O(r)$ проверяем наличие в графе G^* особой цепи. **Если** такая цепь C находится, **то** {находим коэффициенты $\{y_\nu\}$ нетривиальной w-зависимости ее ребер; $y_\nu := 0\ \forall\ e_\nu \in G^* \setminus C$; **stop**} **иначе** переходим к п. 2.

2. Проверяем зависимость циклов C_1 и C_2 по их дискриминантам. В случае зависимости цикла C_i коэффициенты этой зависимости находятся с помощью процедуры $\mathcal{P}$. (Ясно, что при вычислении y_{j_ν} из уравнения (10.9) не возникает деления на ноль. Действительно, если один из коэффициентов $w(e_{j_\nu}, v_{i_{\nu-1}})$ равен нулю, то из $D=0$ следует, что один из коэффициентов $w(e_{j_\nu}, v_{i_\nu})$ также равен нулю. Но в этом случае C_i содержит особую цепь, и алгоритм должен был прекратить работу на шаге 1.)

3. Проверяем наличие в графе G^* полуособой цепи. **Если** такая цепь найдена, **то** {находим коэффициенты нетривиальной w-зависимости ее ребер с помощью процедуры $\mathcal{P}$; **stop**} **иначе** переходим к п. 4.

4. Находим коэффициенты w-зависимости ребер цепи P^* согласно следующей процедуре:

Для конфигурации a:
значения переменных $y_\nu,\ \nu = 1, \ldots, k$, находим с помощью процедуры $\mathcal{P}$;
значение y_{k+1} находится из уравнения баланса в вершине v_k, которое в этом случае выглядит следующим образом:

$$w(e_k, v_k)y_k + w(e_{k+1}, v_k)W(C_2)y_{k+1} = 0$$

(это вычисление корректно, поскольку $w(e_{k+1}, v_k) \neq 0$, а величина $W(C_2)$ определена и также не равна нулю);
значения $y_\nu,\ \nu = k+2, \ldots, r$, находятся из уравнений баланса, связывающих переменные $y_{\nu-1}$ и y_ν.

Для конфигурации b:
значения переменной $y_1 \neq 0$ берется произвольным, а значение y_{d+1} определяется из уравнения баланса в вершине v_0:

$$w(e_1, v_0)W(C_1)y_1 + w(e_{d+1}, v_0)W(C_2)y_{d+1} = 0;$$

последовательно находятся значения $y_\nu,\ \nu = 2, \ldots, k$, и $y_\nu,\ \nu = d+2, \ldots, r$, из уравнений баланса в вершинах $v_{\nu-1}$, связывающих переменные $y_{\nu-1}$ и y_ν;
находятся значения $y_\nu,\ \nu = k+1, \ldots, d$, из уравнений баланса в вершинах $v_{\nu-1}$.
Алгоритм $\mathcal{A}^3_{10.2}$ описан. ■

Для каждого внеостовного ребра $e_i^0 = (v_j^0, v_k^0)$ однозначно определяется содержащий его цикл $C(e_i^0)$, все ребра которого, кроме e_i^0, являются остовными. Более точно, цикл $C(e_i^0)$ образуют ребра множества $\{e_i^0\} \cup (P(v_j^0) \ominus P(v_k^0))$, где $P' \ominus P'' \doteq (E' \cup E'') \setminus (E' \cap E'')$; E' и E'' — множества ребер цепей P' и P'' соответственно. Справедлива следующая

Лемма 10.7 *Множество всех циклических ребер графа G содержится в объединении циклов $C(e_i^0)$ по всем внеостовным ребрам $\{e_i^0\}$.*

Доказательство. Все внеостовные рёбра являются циклическими, и для них утверждение леммы следует из определения циклов $C(e_i^0)$, поэтому достаточно доказать лемму для остовных циклических ребер.

Предположим, что существует остовное циклическое ребро e и содержащий его цикл C' такие, что $e \notin C(e_i^0)$ ни для какого внеостовного ребра $e_i^0 \in C'$. Пусть среди всех циклов с таким свойством цикл C' содержит минимальное число внеостовных ребер. Пусть $e = (v', v'')$, $p(v') = v''$, и дуга (v'', v') задает "правильное" направление обхода цикла C'. Тогда через $C'(v_1, v_2)$ обозначим отрезок цикла C' от вершины v_1 до вершины v_2 при правильном обходе. Пусть $C'(v'', v_j^0, v_k^0)$ — минимальный отрезок цикла C', содержащий ребро e и некоторое внеостовное ребро $e_i^0 = (v_j^0, v_k^0)$. Ясно, что $e \in P(v_j^0)$. (В противном случае $P(v_j^0) \cup C'(v'', v_j^0) \cup P(v'')$ содержит цикл, образованный только остовными ребрами, что невозможно.) Если при этом $e \notin P(v_k^0)$, то $e \in P(v_j^0) \ominus P(v_k^0)$, т.е. $e \in C(e_i^0)$, что противоречит предположению о цикле C'. Следовательно, $e \in P(v_k^0)$. Но тогда в $P(v_k^0) \ominus C'(v_k^0, v'')$ имеется цикл, содержащий ребро e (так как $e \in P(v_k^0)$ и $e \notin C'(v_k^0, v'')$) и на единицу меньшее число внеостовных ребер, что противоречит определению цикла C'. □

Алгоритм $\mathcal{A}_{10.2}^4$ (преобразование информации о графе G^t при удалении из него ребра $e = (v', v'')$)

Алгоритм переопределяет функцию p в вершинах графа G^*, списки внеостовных ребер и их количество ($\Delta(G_i)$) для вновьобразующихся компонент связности G_i.

Если ребро e — внеостовное (что легко устанавливается посредством нахождения корня $v^*(G')$ компоненты $G' = G'(e)$, содержащей ребро e, и сравнения e со всеми ребрами из списка $\hat{E}(v^*)$), то после удаления e пересчитывать функцию p не нужно. Так как e — циклическое, то после удаления e компонента G' остается связной; исключаем e из списка $\hat{E}(v^*)$ и полагаем $\Delta(v^*) := \Delta(v^*) - 1$.

Пусть e — остовное. Различаем два случая: e — циклическое или, наоборот, ациклическое. Для установления цикличности ребра e, согласно лемме 10.7, достаточно проверить его принадлежность циклам $C(e')$ для всех внеостовных ребер $e' \in G'(e)$, что может быть сделано за время $O(\Delta(G')m)$.

Случай 1 (e — циклическое).

Пусть $e \in C(e_i^0)$, $e_i^0 = (v_j^0, v_k^0)$ — внеостовное ребро компоненты $G'(e)$. После удаления e компонента $G'(e)$ остается связной. Функцию p пересчитываем следующим образом.

Ребро e принадлежит одной из двух цепей $P(v_j^0), P(v_k^0)$. Пусть, для определенности, $e \in P(v_j^0) = (v_j^0 v_j^1 \dots v_j^{s_j})$, где $e = (v_j^\nu, v_j^{\nu+1})$. Полагаем $p(v_j^i) := v_j^{i-1}$ для $i \in \mathbb{N}_\nu$ и $p(v_j^0) := v_k^0$. — При этом внеостовное ребро (v_j^0, v_k^0) становится остовным, исключаем его из списка $\hat{E}(G'(e))$. Полагаем $\Delta(G') := \Delta(G') - 1$.

(Замечаем, что если e принадлежит одновременно нескольким циклам $C(e_i^0)$, то описанный выше пересчет функции p достаточно выполнить лишь для одного из циклов.)

Случай 2 (e — ациклическое).

Пусть $e = (v', v'')$, $p(v') = v''$. При удалении e компонента $G'(e)$ распадается на две меньшие компоненты: $G' \setminus \{e\} = G_1 \cup G_2$, где $v' \in G_1, v'' \in G_2$. Корнем компоненты G_1 назначаем вершину v' (полагаем $p(v') = 0$); вершина v^* остается корнем компоненты G_2. Все внеостовные рёбра $e' \in G'(e)$ остаются внеостовными для вновьобразующихся

компонент G_1 и G_2. Для каждого ребра $e' \in G'(e)$ с помощью функции p устанавливаем его принадлежность одной из компонент G_i и включаем e в соответствующий список $\hat{E}(G_i)$; находим значения $\Delta(G_1), \Delta(G_2)$.

Алгоритм $\mathcal{A}^4_{10.2}$ описан. ■

Предоставляем читателю убедиться, что описанные алгоритмы $\mathcal{A}^i_{10.2}$ действительно выполняют декларируемые в их заголовках функции, и что на каждом шаге алгоритма $\mathcal{A}_{10.2}$ для работы его подалгоритмов имеется вся необходимая информация.

Для подтверждения оценки $O(\bar{n}m)$ трудоемкости алгоритма $\mathcal{A}_{10.2}$ мы лишь заметим, что:

— трудоемкость каждого из алгоритмов $\mathcal{A}^i_{10.2}$ $(i \in \mathbb{N}_4)$ не превосходит $O(|G^*|) \leq O(m)$;

— в п.7 алгоритма $\mathcal{A}_{10.2}$ коэффициенты $\{\lambda'_e\}$ пересчитываются лишь для ребер графа G^*.

Лемма 10.6 доказана. □

Для заданого семейства векторов $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset \mathbb{R}^m$ определим *параллелепипед*

$$P(X) \doteq \left\{ \sum_{x_i \in X} \lambda_i x_i \,|\, \lambda_i \in [0,1],\ i \in \mathbb{N}_n \right\}.$$

Лемма 10.8 *Пусть в пространстве $\mathbb{R}^m$ с нормой s заданы семейство векторов $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset B_{m,s}$ и точка $x = \sum \lambda_i x_i \in P(X)$. Тогда с трудоемкостью $O(nm^2)$ можно найти подмножество $N' \subseteq \mathbb{N}_n$, задающее такую вершину параллелепипеда $x' = \sum_{i \in N'} x_i$, что*

$$\|x' - x\|_s \leq m/2. \tag{10.10}$$

Доказательство. Применяя лемму 10.2, находим базу X' семейства X и чи́сла $\{\lambda'_i \in [0,1]\}$ такие, что $(\sum \lambda'_i x_i = x)$ & $(\{x_i \,|\, \lambda'_i \in (0,1)\} \doteq X'' \subseteq X')$. Обозначим $N^- = \{i \,|\, \lambda'_i \in (0, \frac{1}{2})\}$, $N^+ = \{i \,|\, \lambda'_i \in [\frac{1}{2}, 1)\}$ и положим $N' = \{i \in \mathbb{N}_n \,|\, \lambda'_i \in [\frac{1}{2}, 1]\}$. Тогда для $x' = \sum_{i \in N'} x_i$ будем иметь

$$\|x' - x\|_s = \left\| \sum_{i \in N'} x_i - \sum_{i \in N'} \lambda'_i x_i - \sum_{i \in N^-} \lambda'_i x_i \right\|_s = \left\| \sum_{i \in N^+} (1 - \lambda'_i) x_i - \sum_{i \in N^-} \lambda'_i x_i \right\|_s$$

$$\leq \frac{1}{2} \left(|N^+| + |N^-| \right) = \frac{1}{2} |X''| \leq \frac{m}{2}$$ □

Формулируемая ниже лемма является аналогом леммы 10.8 в случае, когда векторы $\{x_i\}$ не произвольны, а определяются согласно функции $w : E \times V \to R^+$, удовлетворяющей (10.4), где $G = (V, E)$ — двудольный граф. При этом специфика векторов $\{x_i\}$ позволяет не только понизить трудоемкость алгоритма, но и существенно “улучшить качество решения”, что выражается в усилении оценки (10.10).

Лемма 10.9 *Пусть задан двудольный граф $G = (V, E)$, где $V = A \cup B$, A и B — левая и правая доли графа; $m = \min\{|A|, |B|\}$; $\bar{m} = |V|$; $\bar{n} = |E|$. Для каждого ребра $e = (v', v'') \in E$ определены неотрицательные веса $w(e, v'), w(e, v'')$ этого ребра относительно его концевых вершин. Заданы числа $\{\lambda_e \in [0, 1] \,|\, e \in E\}$. Тогда с трудоемкостью $O(\bar{n}m + \bar{m})$ можно найти такое подмножество ребер $E' \subseteq E$, что*

$$\sum_{e \in E(v) \cap E'} w(e, v) - \sum_{e \in E(v)} \lambda_e w(e, v) = \varepsilon_v h_v, \ \forall\ v \in V, \tag{10.11}$$

где $E(v) = \{e \in E \,|\, e$ инцидентно $v\}$, $h_v = \max_{e \in E(v)} w(e, v)$, $\varepsilon_v \in (-1, 1)$.

Доказательство. Искомое подмножество ребер E' находится с помощью описываемого ниже алгоритма $\mathcal{A}_{10.3}$, состоящего из четырех шагов.

Алгоритм $\mathcal{A}_{10.3}$ (нахождения подмножества ребер $E' \subseteq E$)

1. Пусть w_e — $\bar{m}$-мерные векторы, определенные для каждого ребра $e \in E$ согласно (10.4); $x = \sum_{e \in E} \lambda_e w_e$. По лемме 10.6, алгоритмом $\mathcal{A}_{10.2}$ находим базу $E_B \subseteq E$ и коэффициенты $\{\lambda'_e \in [0, 1] \,|\, e \in E\}$, такие что $\left(\sum_{e \in E} \lambda'_e w_e = x\right) \& (E'' \subseteq E_B)$, где $E'' = \{e \in E \,|\, \lambda'_e \in (0, 1)\}$.

Замечаем, что в случае двудольного графа G длина наибольшей простой цепи в G не превосходит $2m$, поэтому трудоемкость первого шага составляет $O(\bar{n}m + \bar{m})$.

2. Удаляем из графа $G^t = (V, E_B)$, образовавшегося по завершении последней итерации ($t = \bar{n}$) алгоритма $\mathcal{A}_{10.2}$, все ребра $e \in E_B \setminus E''$. При этом преобразуем информацию о графе G^t согласно алгоритму $\mathcal{A}^4_{10.2}$. Таким образом, трудоемкость второго шага не превосходит $O(|E_B|m) \leq O(\bar{n}m)$.

3. Для каждой вершины $v \in V$ находим список $E'(v)$ ребер $(v, v') \in G^t$, таких что $p(v') = v$. Такие списки по всем вершинам $v \in V$ находятся за один просмотр множества V с трудоемкостью $O(\bar{m})$: если $p(v) \neq 0$, то включаем ребро $(v, p(v))$ в список $E'(p(v))$.

4. "Округляем" коэффициенты $\{\lambda'_e \,|\, e \in E''\}$ до одного из двух значений $\{0, 1\}$. Для этого дважды просматриваем множество вершин V. Если $p(v) = 0$, т.е. v является корнем некоторой компоненты связности G' графа G^t, то выполняем одну из двух описанных ниже процедур, в зависимости от значения $\Delta(G')$: в случае $\Delta(G') = 1$ делаем "округление по циклу", а в случае $\Delta(G') = 0$ — "округление по дереву G'".

Округление по циклу $C(G')$.
С трудоемкостью $O(m)$ находим цикл $C(G')$ компоненты G': для внеостовного ребра $e' = (v', v'')$ строим цепи $P(v')$ и $P(v'')$ и полагаем $C(G') = \{e'\} \cup (P(v') \ominus P(v''))$.

Переносим корень компоненты G' в вершину цикла $C(G')$, для чего переопределяем функцию p в вершинах цепи $P(v') \cap P(v'') = (v_0 v_1 \dots v_s)$: $p(v_i) := v_{i-1},\ i \in \mathbb{N}_s;\ p(v_0) := 0$.

Начиная с внеостовного ребра $e \in \hat{E}(G')$, последовательно обходим рёбра $e \in C(G')$, и коэффициентам λ'_e поочередно присваиваем значения 1 и 0. (Т.е. $\lambda'_e := 1$ для всех нечетных ребер, и $\lambda'_e := 0$ — для четных ребер цикла $C(G')$.) Удаляем рёбра $e \in C(G')$ из списков $E'(v)$ инцидентных им вершин $v \in C(G')$ и из графа G^t. (Для преобразования информации о графе G^t при удалении ребра используем алгоритм $\mathcal{A}^4_{10.2}$.)

Ясно, что по окончании работы процедуры все вершины $v \in C(G')$ становятся корнями вновьобразованных безцикловых компонент связности.

Трудоемкость описанной процедуры не превосходит $O(n(G'))$, где $n(G')$ — число ребер компоненты G'.

Округление по дереву G'.
Коэффициенты λ'_e "округляем" в порядке просмотра ребер дерева G' методом "поиска в глубину". Поиск начинается в корневой вершине:

$v := v^*(G')$;
1: Если список $E'(v)$ пуст, то
{если $p(v) = 0$, то **stop**;
иначе {удаляем ребро $(v, p(v))$ из списка $E'(p(v))$ и графа G^t; $v := p(v)$; **на** 2}}
иначе **на** 2.
2: Берем последнее ребро $e = (v, v')$ в списке $E'(v)$. Полагаем

$$\lambda'_e := \begin{cases} 1, & \text{если } \delta(v) < 0; \\ 0, & \text{в противном случае,} \end{cases} \tag{10.12}$$

где $\delta(v) \doteq \sum_{e \in E(v)} \lambda'_e w(e, v) - \sum_{e \in E(v)} \lambda_e w(e, v)$;
$v := v'$; **на** 1.

Трудоемкость этой процедуры также не превосходит $O(n(G'))$. Ясно, что после первого цикла просмотра вершин $v \in V$ на шаге 4 в графе G^t не останется циклов, а по окончании второго цикла просмотра граф G^t становится пустым. Последнее означает, что все коэффициенты λ'_e, $e \in E$, имеют значения 0 или 1. Полагаем $E' := \{e \in E \,|\, \lambda'_e = 1\}$.

Алгоритм $\mathcal{A}_{10.3}$ описан. ■

Докажем, что так определенное множество E' удовлетворяет (10.11). Так как $\sum_{e \in E(v) \cap E'} w(e, v) - \sum_{e \in E(v)} \lambda_e w(e, v) = \delta(v)$, то достаточно доказать, что по окончании работы алгоритма величи́ны $\delta(v)$ для всех $v \in V$ представимы в виде

$$\delta(v) = \varepsilon h_v, \text{ где } \varepsilon \in (-1, 1). \tag{10.13}$$

Вершину v назовем *неокругленной*, если λ'_e еще не округлялось ни для какого ребра $e \in E(v)$, и *полуокругленной*, если $\delta(v)$ удовлетворяет (10.13). Очевидно, что $\delta(v) = 0$ для неокругленных вершин $v \in V$.

До начала работы алгоритма все вершины $v \in V$ являются неокругленными. Очевидно, что вершины циклической компоненты G', не принадлежащие циклу $C(G')$, остаются неокругленными вплоть до окончания округления по циклу $C(G')$, а вершины $v \in C(G')$ — вплоть до начала работы этой процедуры. Покажем, что по окончании округления по циклу $C(G')$ вершины $v \in C(G')$ становятся полуокругленными.

Действительно, так как любой цикл в двудольном графе имеет четную длину, то для каждой вершины $v \in C(G')$ и одного из двух инцидентных ей циклических ребер (e_1) соответствующий коэффициент (λ'_{e_1}) получит значение 0, а для второго ребра (e_2)

— значение 1. При этом сумма $\sum_{e\in E(v)} \lambda'_e w(e,v)$ изменится на $\delta(v) = (1-\lambda'_{e_2})w(e_2,v) - \lambda'_{e_1} w(e_1,v)$. Так как веса $w(e_i,v)$ неотрицательны, то $\delta(v)$ удовлетворяет (10.13).

Из доказанного следует, что для любой безцикловой компоненты G' до начала применения к ней процедуры округления по дереву ее корневая вершина $v^*(G')$ является полуокругленной (в частности, она может быть неокругленной), а все некорневые вершины $v \in G'$ являются неокругленными.

Покажем, что при выполнении процедуры "округление по дереву G'" все вершины дерева на каждом шаге процедуры являются полуокругленными (а следовательно, они являются таковыми по окончании работы процедуры и всего алгоритма $\mathcal{A}_{10.3}$, что и требуется доказать). Как было отмечено выше, это утверждение верно к моменту начала выполнения процедуры. Пусть оно верно к моменту округления на ребре $e = (v, v')$, где v — "ближняя", а v' — "дальняя" (т.е. более удаленная от корня) вершина ребра e. Очевидно, что при просмотре ребер методом "поиска в глубину" дальняя вершина текущего ребра всегда является неокругленной. По индукционному предположению, ближняя вершина ребра e является полуокругленной. Ясно однако, что округление λ'_e по формуле (10.12) не нарушает полуокругленности вершины v, как и вершины v'.

Для завершения доказательства леммы остается убедиться в справедливости оценки трудоемкости алгоритма. Она следует из приведенных выше оценок трудоемкости его шагов. Лемма 10.9 доказана. □

Замечание 10.10 *Нетрудно видеть, что если G в лемме 10.9 является произвольным (не обязательно двудольным) графом, а каждое ребро $e = (v', v'') \in E$ имеет произвольные (не обязательно неотрицательные) веса $w(e,v'), w(e,v'')$, то алгоритм $\mathcal{A}_{10.3}$ гарантирует нахождение подмножества ребер $E' \subseteq E$ с аналогичной оценкой*

$$\sum_{e\in E(v)\cap E'} w(e,v) - \sum_{e\in E(v)} \lambda_e w(e,v) = \varepsilon_v h_v,\ \forall\ v \in V,$$

где $\varepsilon_v \in (-2,2)$. □

Из приведенного выше замечания вытекает следующее

Следствие 10.11 *Пусть в лемме 10.8 каждый из векторов $\{x_i \in \mathbb{R}^m \,|\, i \in \mathbb{N}_n\}$ имеет не более двух ненулевых компонент. Тогда с трудоемкостью $O(nm)$ находится такое подсемейство векторов $\{x_i \,|\, i \in N'\},\ N' \subseteq \mathbb{N}_n$, что*

$$\left|\sum_{i\in N'} x_i(j) - \sum_{i\in\mathbb{N}_n} \lambda_i x_i(j)\right| \le 2 \max_{i\in\mathbb{N}_n} |x_i(j)|,\ \forall\ j \in \mathbb{N}_m.$$ □

* * *

Далее рассмотрим следующую оптимизационную геометрическую задачу *о подмножестве векторов с заданной суммой* в пространстве $\mathbb{R}^m$ с нормой s.

Задача ПВЗС$_{m,s}(\boldsymbol{X}, \boldsymbol{x})$. *В векторном пространстве $\mathbb{R}^m$ с нормой s для заданного семейства векторов $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset B_{s,m}$ требуется найти подсемейство*

векторов $\{x_i \,|\, i \in N'\},\ N' \subseteq \mathbb{N}_n$, *с суммой, наиболее близкой к заданному вектору* $x \in \mathbb{R}^m$. *Иначе говоря, требуется найти подмножество индексов* $N' \subseteq \mathbb{N}_n$, *минимизирующее функцию*

$$\xi^s_{X,x}(N') \doteq \left\| x - \sum_{i \in N'} x_i \right\|_s . \qquad \square$$

Следующая задача является непрерывной релаксацией задачи ПВЗС$_{m,s}(X, x)$.

Задача ПВЗС$'_{m,s}(X, x)$. *В векторном пространстве* $\mathbb{R}^m$ *с нормой* s *для заданного семейства векторов* $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset B_{s,m}$ *и вектора* $x \in \mathbb{R}^m$ *требуется найти семейство чисел* $\Lambda = \{\lambda_i \in [0, 1] \,|\, i \in \mathbb{N}_n\}$, *минимизирующее функцию*

$$\hat{\xi}^s_{X,x}(\Lambda) \doteq \left\| x - \sum_{i \in \mathbb{N}_n} \lambda_i x_i \right\|_s . \qquad \square$$

Ясно, что оптимальное решение задачи ПВЗС$'_{m,s}(X, x)$ дает нижнюю оценку задачи ПВЗС$_{m,s}(X, x)$. В случаях, когда норма s допускает "линеаризацию" (т.е. ее единичный шар представи́м в виде пересечения конечного числа замкнутых полупространств), задача ПВЗС$'_{m,s}(X, x)$ допускает эквивалентную ЛП-формулировку, а следовательно, имеет эффективное решение. В частности, в случае нормы l_∞ задача ПВЗС$'_{m,l_\infty}(X, x)$ может быть сформулирована следующим образом.

Минимизировать функцию λ_0 от переменных $\{\lambda_i \,|\, i = 0, \ldots, n\}$ при ограничениях

$$\lambda_0 + \sum_{i \in \mathbb{N}_n} x_i(j)\lambda_i \geq x(j),\ j \in \mathbb{N}_m, \tag{10.14}$$

$$\lambda_0 - \sum_{i \in \mathbb{N}_n} x_i(j)\lambda_i \geq -x(j),\ j \in \mathbb{N}_m, \tag{10.15}$$

$$0 \leq \lambda_i \leq 1,\ i \in \mathbb{N}_n. \tag{10.16}$$

Нетрудно видеть, что в случае, когда вектор x принадлежит параллелепипеду $P(X)$, релаксационная задача ПВЗС$'_{m,s}(X, x)$ имеет тривиальное решение, а ее оптимум в этом случае равен нулю, и в качестве нижней оценки оптимума задачи ПВЗС$_{m,s}(X, x)$ не представляет интереса. В то же время, при $x \in P(X)$ возможны более эффективные (чем в случае произвольного x) алгоритмы приближенного решения задачи ПВЗС$_{m,s}(X, x)$, не требующие решения релаксационной задачи ПВЗС$'_{m,s}(X, x)$. Например, лемма 10.8 при $x \in P(X)$ предоставляет нам эффективный (трудоемкости $O(nm^2)$) алгоритм приближенного решения задачи ПВЗС$_{m,s}(X, x)$ для любой нормы s с гарантированной оценкой абсолютного отклонения от оптимума, не зависящей от числа векторов:

$$\xi^s_{X,x}(N') - \xi^s_{X,x}(N_{\text{opt}}) \leq m/2. \tag{10.17}$$

Ввиду вышесказанного, задачу ПВЗС$_{m,s}(X, x)$ с $x \in P(X)$, представляющую самостоятельный интерес, будем называть *Задачей о ближайшей вершине* и обозначать БВ$_{m,s}(X, x)$.

Заметим, что принадлежность вектора x параллелепипеду $P(X)$ проверяется эффективно: $x \in P(X)$, если и только если задача (10.14)–(10.16) имеет решение $\lambda_0 = 0$.

В случае, когда вектор x произволен, а норма s допускает линеаризацию, для нахождения приближенного решения задачи $\text{ПВЗС}_{m,s}(X,x)$ с оценкой вида (10.17) применяется следующий эффективный алгоритм. Сначала находим точное решение $\{\lambda_i \,|\, i \in \mathbb{N}_n\}$ задачи $\text{ПВЗС}'_{m,s}(X,x)$, а затем решаем задачу $\text{БВ}_{m,s}(X,x')$ для вектора $x' = \sum \lambda_i x_i \in P(X)$ одним из имеющихся эффективных алгоритмов (например, применяя алгоритм из леммы 10.8).

* * *

Представляют интерес случаи задачи $\text{БВ}_{m,s}(X,x')$ для конкретных норм s. Так например, для евклидовой нормы ($s = l_2$) нами было показано [34], что задача $\text{БВ}_{m,s}(X,x)$ решается с трудоемкостью $O(nm^2 + m^4)$ и существенно лучшей (по сравнению с (10.17)) априорной оценкой точности:

$$\xi_{l_2,X}(N') \le \sqrt{m}/2, \tag{10.18}$$

причем оценка (10.18) является наилучшей возможной, поскольку существуют семейство векторов X и точка x, для которых задача $\text{БВ}_{m,s}(X,x)$ имеет оптимальное решение $\xi_{l_2,X}(N'_{\text{opt}}) = \sqrt{m}/2$.

Далее покажем, что для нормы $s = l_\infty$ задача $\text{БВ}_{m,s}(X,x)$ также решается с линейной от n трудоемкостью и лучшей по сравнению с (10.17) априорной оценкой точности. Искомый результат будет вытекать из следующей леммы.

Лемма 10.12 *Для любого $\varepsilon > \frac{1}{2}$ существует такое целое число $m(\varepsilon)$, что для любого $m \ge m(\varepsilon)$, любого заданного семейства векторов $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset B_{l_\infty,m}$ и точки $x = \sum_{i\in\mathbb{N}_n} \lambda_i x_i$ параллелепипеда $P(X)$ ($\lambda_i \in [0,1],\ i \in \mathbb{N}_n$) найдется такая вершина параллелепипеда $y = \sum_{i\in\mathbb{N}_n} a_i x_i$ ($a_i \in \{0,1\},\ i \in \mathbb{N}_n$), что*

$$\|y - x\|_{l_\infty} \le \left(1 + \frac{n}{m}\right)\sqrt{\varepsilon m \ln 2m}. \tag{10.19}$$

При этом искомая вершина y находится с трудоемкостью $O(nm)$.

Доказательство. Поскольку функция $\ln(1+x)$ разлагается в ряд:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots,$$

то при любом $\varepsilon > \frac{1}{2}$ найдется такое $\delta > 0$, что

$$1 + x \ge \exp(x - \varepsilon x^2),\ \forall\ x \in [-\delta, \delta]. \tag{10.20}$$

Через $\delta(\varepsilon)$ обозначим максимальное δ, удовлетворяющее (10.20). Например, имеем:

$$\delta(1) > 0.68, \quad \delta(0.54) > 0.1. \tag{10.21}$$

Очевидно, $\delta(\varepsilon) < 1,\ \forall\ \varepsilon > \frac{1}{2}$.

Существование вершины y, удовлетворяющей оценке (10.19), равносильно существованию таких чисел $w_i,\ i \in \mathbb{N}_n$, принимающих значения $w_i \in \{-\lambda_i, 1-\lambda_i\}$, что для любого $k \in \mathbb{N}_m$ выполняется неравенство

$$\left|\sum_{i\in\mathbb{N}_n} w_i x_i(k)\right| \le \left(1+\frac{n}{m}\right)\sqrt{\varepsilon m \ln 2m},$$

где $x_i = (x_i(1), \ldots, x_i(m))$. Для отыскания искомых чисел w_i воспользуемся алгоритмом, подобным тому, что Бек и Фиала использовали в [76] для решения задачи Integer making. (Формулировку последней см. в разделе 20.)

Алгоритм

Последовательность чисел $w_i,\ i = 1, \ldots, n$ строим по индукции. Пусть числа $w_i,\ i = 1, \ldots, j-1$ уже определены. Обозначим

$$R^{\pm}_{j-1,k} = \prod_{i\le j-1}(1 \pm \beta\, w_i\, x_i(k)), \quad R^{\pm}_{j,k}(y) = R^{\pm}_{j-1,k}\cdot(1 \pm \beta\, y\, x_j(k)),$$

$$Q_{j-1} = \sum_{k=1}^{m}(R^{+}_{j-1,k} + R^{-}_{j-1,k}), \quad Q_j(y) = \sum_{k=1}^{m}(R^{+}_{j,k}(y) + R^{-}_{j,k}(y)),$$

где β определяется из формулы

$$\beta = \sqrt{\frac{\ln 2m}{\varepsilon m}} \tag{10.22}$$

Тогда в качестве w_j выбирается значение $y \in \{-\lambda_j, 1-\lambda_j\}$, минимизирующее функцию $Q_j(y)$. ■

Докажем, что

$$\left|\sum_{i\le n} w_i x_i(k)\right| \le \left(1+\frac{n}{m}\right)\sqrt{\varepsilon m \ln 2m},\ \forall\ k \in \mathbb{N}_m. \tag{10.23}$$

Из определения величин Q_{j-1} и $Q_j(y)$ имеем:

$$Q_j(1-\lambda_j) = Q_{j-1} + \beta(1-\lambda_j)\sum_{k=1}^{m} x_j(k)(R^{+}_{j-1,k} - R^{-}_{j-1,k}),$$

$$Q_j(-\lambda_j) = Q_{j-1} + \beta(-\lambda_j)\sum_{k=1}^{m} x_j(k)(R^{+}_{j-1,k} - R^{-}_{j-1,k}).$$

Отсюда для $Q_j = \min\{Q_j(1-\lambda_j), Q_j(-\lambda_j)\}$ получаем

$$Q_j - Q_{j-1} \le \lambda_j(Q_j(1-\lambda_j) - Q_{j-1}) + (1-\lambda_j)(Q_j(-\lambda_j) - Q_{j-1}) = 0. \tag{10.24}$$

Поскольку $Q_0 = 2m$ (считая пустое произведение равным 1), то из (10.24) следует

$$Q_j = \sum_{k=1}^{m} (R_{j,k}^{+} + R_{j,k}^{-}) \le 2m, \ \forall j. \tag{10.25}$$

Пусть $m(\varepsilon)$ есть наименьшее такое число m', что

$$\varepsilon\delta^2(\varepsilon)m \ge \ln 2m, \ \forall\ m \ge m'.$$

Тогда для любого $m \ge m(\varepsilon)$ и β, определенного согласно (10.22), имеем $\beta \le \delta(\varepsilon) < 1$.

Поскольку все величины w_i и $x_i(k)$ по модулю не превосходят 1, то $R_{j,k}^{\pm} \ge 0, \ \forall\ j,k$. С учетом этого из (10.25) имеем:

$$\prod_{i\le j}(1 \pm \beta\, w_i\, x_i(k)) \le 2m, \ \forall\ j,k.$$

Используя неравенство (10.20) со значениями $x = \pm\beta\, w_i\, x_i(k)$, получаем соотношения

$$2m \ge \prod_{i\le n}(1 \pm \beta\, w_i\, x_i(k)) \ge \exp\left\{\pm\beta\left(\sum_{i\le n} w_i\, x_i(k)\right) - \varepsilon\beta^2\left(\sum_{i\le n} w_i^2\, x_i^2(k)\right)\right\},$$

откуда

$$\pm\sum_{i\le n} w_i\, x_i(k) \le \varepsilon n\beta + \frac{\ln 2m}{\beta}. \tag{10.26}$$

Подставляя в правую часть (10.26) значение β из (10.22), получим соотношение

$$\pm\sum_{i\le n} w_i\, x_i(k) \le \left(1 + \frac{n}{m}\right)\sqrt{\varepsilon m \ln 2m},$$

равносильное (10.23) и справедливое для всех $m \ge m(\varepsilon)$. Остается заметить, что для нахождения каждого числа $w_i, \ i = 1,\ldots,n$ вычисляются значения функции $Q_i(y)$ для каждого из двух допустимых значений y, для чего требуется $O(m)$ вычислительных операций. Таким образом, трудоемкость алгоритма составляет $O(nm)$. Лемма 10.12 доказана. □

Теорема 10.13 *Существует алгоритм приближенного решения задачи* $БВ_{m,l_\infty}$ *с временной сложностью* $O(nm^2)$*, который для любого заданного семейства векторов* $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset B_{s,m}$ *и любой точки* $x = \sum_{i\in\mathbb{N}_n} \lambda_i x_i \in P(X)$ $(\lambda_i \in [0,1], \ i \in \mathbb{N}_n)$ *находит такое подсемейство векторов* $\{x_i \,|\, i \in N'\}, \ N' \subseteq \mathbb{N}_n$*, что*

$$\xi_{X,x}^{l_\infty}(N') = \left\| x - \sum_{i\in N'} x_i \right\|_{l_\infty} \le \begin{cases} k(n,m)\sqrt{m\ln 2m}, & \textit{для всех } m; \\ 0.735\cdot k(n,m)\sqrt{m\ln 2m}, & \textit{для всех } m \ge 1500; \end{cases}$$

$$k(n,m) = \begin{cases} 1 + \frac{n}{m}, & \textit{при } n \le m; \\ 2, & \textit{при } n \ge m. \end{cases} \tag{10.27}$$

Доказательство. Вначале заметим, что задача $\text{БВ}_{m,s}$ с $n \geq m$ векторами сводится к задаче с m векторами. Для этого достаточно воспользоваться алгоритмом из леммы 10.2 и преобразовать точку x к виду $x = \sum_{i\in\mathbb{N}_n} \lambda'_i x_i$, где $\lambda'_i \in \{0,1\}$ для всех индексов i за исключением не более чем m индексов из множества $N_B = \{i \in \mathbb{N}_n \,|\, \lambda'_i \in (0,1)\}$. Такое преобразование выполняется с трудоемкостью $O(nm^2)$, после чего решаем задачу БВ для семейства векторов $X' = \{x_i \,|\, i \in N_B\}$ и точки $x' = \sum_{i\in N_B} \lambda'_i x_i \in P(X')$, используя алгоритм из леммы 10.12.

Поскольку при $\varepsilon = 1$ имеем $m(\varepsilon) = 5$, то справедливость оценки

$$\xi^{l_\infty}_{X,x}(N') \leq k(n,m)\sqrt{m \ln 2m}$$

при всех $m \geq 5$ вытекает из леммы 10.12. Для всех $m \leq 4$ эта оценка проверяется непосредственно, исходя из тривиальной оценки

$$\xi^{l_\infty}_{X,x}(N') \leq \frac{1}{2}\min\{n,m\},$$

достижимой простым округлением коэффициентов $\{\lambda_i\}$.

При $\varepsilon = 0.54$ оценка

$$\xi^{l_\infty}_{X,x}(N') \leq k(n,m)\sqrt{0.54\, m \ln 2m}, \quad m \geq 1500,$$

вытекает непосредственно из леммы 10.12 и оценки $m(0.54) < 1500$.

Теорема 10.13 доказана. □

* * *

Пусть k и q — натуральные числа, $q < k$. Сформулируем оптимизационную задачу о нахождении подмножества операций, которую будем коротко обозначать $\text{ПО}(k,q)$.

Задача ПО(k,q). *Пусть задано множество операций $\mathcal{O} = \{o_i \,|\, i \in \mathbb{N}_{\bar{n}}\}$; $\bar{n} = kn$. Каждая операция o_i принадлежит некоторой работе $J(o_i) \in \{J_j \,|\, j \in \mathbb{N}_n\}$ и выполняется на определенной машине $M(o_i) \in \{M_j \,|\, j \in \mathbb{N}_m\}$ за время p_i. Пусть $D_j = \{o_i \,|\, J(o_i) = J_j\}$ — множество операций работы J_j, $F_j = \{o_i \,|\, M(o_i) = M_j\}$ — множество операций машины M_j, $L_j = \sum_{o_i\in F_j} p_i$ — нагрузка машины M_j, и пусть выполняется $|D_j| = k$, $\forall\, j \in \mathbb{N}_n$. Требуется найти подмножество операций $\mathcal{O}' \subset \mathcal{O}$, минимизирующее функционал*

$$\psi(\mathcal{O}') \doteq \max_{j\in\mathbb{N}_m} \left| \sum_{o_i\in F_j\cap\mathcal{O}'} p_i - \frac{q}{k} L_j \right|$$

при ограничениях

$$|\{D_j \cap \mathcal{O}'\}| = q, \ \forall\, j \in \mathbb{N}_n. \tag{10.28}$$

Следующая лемма предоставляет эффективный алгоритм приближенного решения этой задачи.

Лемма 10.14 *Для любых параметров k, q и множества операций $\mathcal{O}$ существует решение $\mathcal{O}' \subset \mathcal{O}$ задачи ПО(k, q), удовлетворяющее оценкам*

$$\left| \sum_{o_i \in F_j \cap \mathcal{O}'} p_i - \frac{q}{k} L_j \right| \leq p^j_{\max}, \ \forall\ j \in \mathbb{N}_m, \tag{10.29}$$

где $p^j_{\max} = \max_{o_i \in F_j} p_i$. Искомое решение может быть найдено за время $O(\bar{n}m)$[55].

Доказательство. Определим двудольный граф $G = (A, B; E)$, где вершины одной доли ($A = \{a_i \,|\, i \in \mathbb{N}_m\}$) соответствуют машинам $\{M_i \,|\, i \in \mathbb{N}_m\}$, а вершины другой доли ($B = \{b_j \,|\, j \in \mathbb{N}_n\}$) — работам $\{J_j \,|\, j \in \mathbb{N}_n\}$; каждой операции $o_\nu \in \mathcal{O}$ соответствует ребро $(a_i, b_j) \in E$, где $M_i = M(o_\nu),\ J_j = J(o_\nu)$. Следуя работе [84], весовую функцию w на ребрах $e \in E$ определим следующим образом: если ребро $e = (a_i, b_j)$ соответствует операции o_ν, то полагаем $w(e, a_i) = p_\nu,\ w(e, b_j) = 1$. Тогда для коэффициентов $\lambda_e \equiv q/k,\ \forall\ e \in E$, будем иметь

$$\sum_{e \in E(v)} \lambda_e w(e, v) = \begin{cases} q, & \text{если } v \in B, \\ \frac{q}{k} L_i & \text{если } v = a_i \in A. \end{cases} \tag{10.30}$$

Применив алгоритм $\mathcal{A}_{10.3}$ из леммы 10.9, с трудоемкостью $O(\bar{n}m)$ находим подмножество ребер $E' \subseteq E$, такое что выполнено (10.11) с $\varepsilon_v \in (-1, 1),\ \forall\ v \in V$. Тогда для $v \in B$ из (10.11), (10.30), равенства $h_v = 1$ и целочисленности весов $w(e, v)$ следует равенство

$$\sum_{e \in E(v) \cap E'} w(e, v) = |E(v) \cap E'| = q,$$

а для $v = a_j \in A$ из (10.11), (10.30) и $h_v = p^j_{\max}$ — равенство

$$\sum_{e \in E(v) \cap E'} w(e, v) - \frac{q}{k} L_j = \varepsilon_j p^j_{\max}$$

для некоторого $\varepsilon_j \in (-1, 1)$, откуда вытекают равенство (10.28) и неравенство (10.29). □

11 Компактное суммирование векторов

Определение 11.1 Конечное семейство векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ называется s-семейством, если $X \subset B_{s,m}$ и $\Sigma(X) = 0$.

[55] В работе Фиалы [84] такое решение отыскивается для случая $q = k/2$ (k — четное) алгоритмом трудоемкости $O(\bar{n}^2)$.

В работе [131] Штейниц установил следующее свойство s-семейств векторов.

Лемма (Штейница) *Для любого s-семейства векторов $X = \{x_1, \ldots, x_n\}$ существует перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что*

$$\left\| \sum_{j=1}^{k} x_{\pi_j} \right\|_s \leq C, \ k \in \mathbb{N}_n,$$

где C — константа, не зависящая от семейства векторов (C зависит от нормы и размерности пространства).

Иными словами, для всякого конечномерного пространства и любой нормы s существует такая константа C, что для всякого конечного s-семейства векторов существует такой порядок их суммирования, при котором все частичные суммы находятся в шаре радиуса C (с центром в начале координат).

В связи с леммой Штейница возникает естественный вопрос о нахождении минимального значения константы C с этим свойством. Функцию, отражающую зависимость такого минимального значения от размерности пространства, мы называем *функцией Штейница* и для заданной в пространстве нормы s обозначаем через $\varphi_s(m)$. Более подробному исследованию свойств и нахождению оценок функций Штейница посвящен следующий раздел диссертации. Здесь же нас будет интересовать более конструктивный вопрос — об отыскании перестановки π, обеспечивающей суммирование s-семейства векторов в шаре минимального радиуса, или иначе, о *компактном суммировании векторов* из заданного s-семейства.

Задача КСВ$_{m,s}(X)$. *Пусть в пространстве $\mathbb{R}^m$ с нормой s задано s-семейство векторов $X = \{x_1, \ldots, x_n\}$. Требуется найти перестановку $\pi = (\pi_1, \ldots, \pi_n)$, минимизирующую функционал*

$$f_{s,X}(\pi) = \max_{k \in \mathbb{N}_n} \left\| \sum_{j=1}^{k} x_{\pi_j} \right\|_s .$$

Интуитивную очевидность того, что эта задача не может иметь легкого решения, подтверждает теорема 11.4, где устанавливается NP-трудность в сильном смысле задачи КСВ на распознавание уже в одномерном случае (и следовательно, для любой нормы). Такую одномерную задачу назовем задачей *компактного суммирования чисел*, или КСЧ. Для фиксированного параметра $\mu > 0$ через КСЧ(μ) будем обозначать следующую задачу на распознавание.

КСЧ(μ).
УСЛОВИЕ: задано семейство целых чисел $X = \{x_1, \ldots, x_n\} \subset Z$ такое, что $\sum_i x_i = 0$; $K(X) \doteq \max_i |x_i|$.
ВОПРОС: существует ли перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что

$$\left| \sum_{j=1}^{k} x_{\pi_j} \right| \leq \mu K(X), \ k \in \mathbb{N}_n? \tag{11.1}$$

При $\mu < 1/2$ ответ на вопрос задачи КСЧ(μ) всегда отрицателен. При $\mu \geq 1$ ответ всегда положителен, причем перестановка π находится за линейное от n время. При $\mu = 1/2$ справедлива

Лемма 11.2 *Задача* КСЧ($\frac{1}{2}$) *является NP-полной в сильном смысле.* □

Утверждение леммы легко доказывается путем сведения к задаче КСЧ($\frac{1}{2}$) известной NP-полной в сильном смысле задачи 3-РАЗБИЕНИЕ [17, с. 283]. Для доказательства аналогичного результата при любом $\mu \in (\frac{1}{2}, 1)$ нам потребуется сначала доказать NP-полноту в сильном смысле задачи k-РАЗБИЕНИЕ при любом $k \geq 3$.

***k*-РАЗБИЕНИЕ.**
УСЛОВИЕ: заданы множество A, состоящее из km элементов, граница $B \in Z^+$ и такие размеры $s(a) \in Z^+$ всех элементов $a \in A$, что $B/(k+1) < s(a) < B/(k-1)$ и $\sum_{a \in A} s(a) = mB$.
ВОПРОС: можно ли множество A разбить на m непересекающихся подмножеств $A_1, \ldots, A_m$ так, что для любого $i \in \mathbb{N}_m$ выполнено соотношение

$$\sum_{a \in A_i} s(a) = B? \tag{11.2}$$

Заметим, что каждое множество A_i должно содержать ровно k элементов.

Лемма 11.3 *Задача k-РАЗБИЕНИЕ NP-полна в сильном смысле* $\forall\ k \geq 3$.

Доказательство. При $k = 3$ NP-полнота задачи доказана в [86]. Пусть $k > 3$. Сведем к ней задачу 3-РАЗБИЕНИЕ.

Пусть заданы множество A, состоящее из $3m$ элементов, граница $B \in Z^+$ и размеры $\{s(a) \in Z^+ \,|\, a \in A\}$, такие что

$$B/4 < s(a) < B/2,\ \sum_{a \in A} s(a) = mB.$$

Сформируем множество $\tilde{A}$ из km элементов, такое что $\tilde{A} = A \cup A'$, $A \cap A' = \emptyset$, и определим размеры $\{\tilde{s}(a) \in Z^+ \,|\, a \in \tilde{A}\}$:

$$\tilde{s}(a) = \begin{cases} C + s(a) & \text{при } a \in A, \\ C + B & \text{при } a \in A', \end{cases}$$

где $C = (k-2)B$. Определим границу

$$\tilde{B} = B + 3C + (k-3)(C+B) = (k^2 - k - 2)B = kC + (k-2)B.$$

Предоставляем читателю убедиться, что

$$\tilde{B}/(k+1) < \tilde{s}(a) < \tilde{B}/(k-1),\ \forall a \in \tilde{A},\ \text{и}\ \sum_{a \in \tilde{A}} \tilde{s}(a) = m\tilde{B}.$$

Покажем, что искомое 3-разбиение $\{A_1,\ldots,A_m\}$ множества A существует если и только если существует k-разбиение $\{\tilde{A}_1,\ldots,\tilde{A}_m\}$ множества $\tilde{A}$, удовлетворяющее свойству

$$\sum_{a\in\tilde{A}_i}\tilde{s}(a)=\tilde{B}. \tag{11.3}$$

По 3-разбиению k-разбиение строится просто: произвольно разбиваем множество A' на m подмножеств $A'_1,\ldots,A'_m$ мощности $k-3$ и полагаем $\tilde{A}_i=A_i\cup A'_i$.

Обратно, пусть задано k-разбиение $\{\tilde{A}_1,\ldots,\tilde{A}_m\}$, удовлетворяющее (11.3). Положим $A_i=\tilde{A}_i\cap A,\ A'_i=\tilde{A}_i\cap A'$. Из определения размеров $\tilde{s}(a)$ убеждаемся, что если $|A'_i|<k-3$, то при любом A_i мощности $k-|A'_i|$ будем иметь $\sum_{a\in\tilde{A}_i}\tilde{s}(a)<\tilde{B}$, а если $|A'_i|>k-3$, то $\sum_{a\in\tilde{A}_i}\tilde{s}(a)>\tilde{B}$. Таким образом, возможно лишь $|A'_i|=k-3,\ \forall i\in\mathbb{N}_m$, что автоматически дает 3-разбиение $\{A_1,\ldots,A_m\}$ множества A, удовлетворяющее (11.2).

Поскольку длина записи входной информации задачи k-РАЗБИЕНИЕ равна по порядку длине записи исходной задачи 3-РАЗБИЕНИЕ (при фиксированном k), лемма 11.3 доказана. □

Теорема 11.4 *При любом фиксированном $\mu\in[\frac{1}{2},1)$ задача* КСЧ(μ) *NP-полна в сильном смысле.*

Доказательство. При $\mu=1/2$ это является утверждением леммы 11.2. Пусть $\mu\in(\frac{1}{2},1)$. Положим $k=\max\{3,\lceil\mu/(1-\mu)\rceil\}$ и сведем задачу k-РАЗБИЕНИЕ к задаче КСЧ(μ).

Пусть в задаче k-РАЗБИЕНИЕ задано множество $A=\mathbb{N}_{km}$, граница $B\in Z^+$ и такие размеры $s(a)\in Z^+$ всех элементов $a\in A$, что

$$B/(k+1)<s(a)<B/(k-1) \tag{11.4}$$

и $\sum_{a\in A}s(a)=mB$. Положим $n=(2k+3)m,\ A'=\{km+1,\ldots,(k+2)m\}\doteq\{\beta_1,\ldots,\beta_{2m}\}$, $A''=\{(k+2)m+1,\ldots,n\}\doteq\{\gamma_1,\ldots,\gamma_{(k+1)m}\},\ K=2\lceil B/(2\mu-1)\rceil$ и определим семейство чисел $X=\{x_1,\ldots,x_n\}\subset Z$ следующим образом.

$$x_i=\begin{cases}2s(i)-K & ,i\in A;\\ -\lfloor\mu K\rfloor & ,i\in A';\\ K & ,i\in A''.\end{cases}$$

Убеждаемся, что $K(X)=K$ и

$$\begin{aligned}\sum_{i=1}^{n}x_i &= 2mB-kmK-2m\lfloor\mu K\rfloor+(k+1)mK=\\ &= (2B+K-2\lfloor\mu K\rfloor)m=0.\end{aligned} \tag{11.5}$$

Кроме того, из 11.4 и определения для k,K для любого $i\in A$ имеем

$$\begin{aligned}2s(i) &< \frac{2B}{k-1}\le 2B/\left(\left\lceil\frac{\mu}{1-\mu}\right\rceil-1\right)\le 2B/\left(\frac{\mu}{1-\mu}-1\right)=\frac{2B(1-\mu)}{2\mu-1}\le\\ &\le (1-\mu)K,\end{aligned}$$

откуда

$$x_i = 2s(i) - K < -\mu K, \ \forall i \in A. \tag{11.6}$$

Покажем, что в задаче k-РАЗБИЕНИЕ искомое разбиение $\{A_1, \ldots, A_m\}$ удовлетворяющее (11.2), существует если и только если в задаче КСЧ(μ) существует перестановка $\pi = (\pi_1, \ldots, \pi_n)$, удовлетворяющая (11.1).

Пусть разбиение $\{A_1, \ldots, A_m\}$ множества A удовлетворяет (11.2). Элементы множества A_j обозначим $\{\alpha_1^j, \ldots, \alpha_k^j\}$. Организуем перестановку π следующим образом. Она будет состоять из m сегментов: $\pi = \pi^1 \oplus \pi^2 \oplus \cdots \oplus \pi^m$, где $\oplus$ — конкатенация, а сегмент π^j $(j \in \mathbb{N}_m)$ определен как

$$\pi^j = (\pi_1^j, \ldots, \pi_{2k+3}^j) = (\beta, \gamma, \alpha_1^j, \gamma, \alpha_2^j, \gamma, \ldots, \alpha_k^j, \gamma, \beta),$$

где β — некоторый элемент из A' (напоминаем, что все числа $\{x_i \,|\, i \in A'\}$ равны), γ — некоторый элемент из A''.

Сумма чисел $\{x_i\}$ по любому сегменту π^j равна

$$\begin{aligned}\sum_{\nu=1}^{2k+3} x_{\pi_\nu^j} &= 2x_\beta + \sum_{i \in A_j} x_i + (k+1)x_\gamma = (k+1)K - 2\lfloor \mu K \rfloor + 2B - kK = \\ &= 2B + K - 2\lfloor \mu K \rfloor = 0.\end{aligned}$$

Кроме того, нетрудно проверить, что

$$\left| \sum_{\nu=1}^{j} x_{\pi_\nu^j} \right| \le \lfloor \mu K \rfloor \le \mu K, \ \forall i \in \mathbb{N}_{2k+3}.$$

Из двух последних соотношений следует, что перестановка π удовлетворяет (11.1).

Обратно, пусть имеется перестановка π, удовлетворяющая (11.1). Покажем, что она может быть организована только так, как описано выше, причем разрезание ее на m сегментов, в каждом из которых содержится ровно k элементов $\alpha \in A$, дает разбиение $\{A_1, \ldots, A_m\}$ множества A, удовлетворяющее (11.2).

Действительно, так как в $A \cup A'$ на m элементов больше, чем в A'', то перестановка π должна содержать сегменты (назовем их α-β-*сегментами*), состоящие из двух идущих подряд элементов из $A \cup A'$. (Здесь мы рассматриваем π как циклически замкнутую перестановку элементов; элемент π_1 является "следующим" за π_n.) Ясно, что в π не может идти подряд более двух элементов из $A \cup A'$ и более одного элемента из A'', поэтому π содержит ровно m не смежных между собой α-β-сегментов. Кроме того, из (11.6) и (11.1) приходим к выводу, что оба элемента α-β-сегмента должны быть из A', поэтому уместно называть его β-*сегментом*. Наконец, из (11.1) следует, что для каждого β-сегмента (π_l, π_{l+1}) должно выполняться

$$\sum_{j=1}^{l-1} x_{\pi_j} = \lfloor \mu K \rfloor, \ \sum_{j=1}^{l+1} x_{\pi_j} = -\lfloor \mu K \rfloor,$$

а следовательно, $\sum_{j=1}^{l} x_{\pi_j} = 0$. Замечая, что $\pi_1, \pi_n \in A'$, приходим к выводу, что перестановка π является конкатенацией из m сегментов: $\pi = \pi^1 \oplus \cdots \oplus \pi^m$, таких что:

— каждый из них (π^j) начинается и заканчивается β-элементом и других элементов из A' не содержит;

— остальная часть сегмента представляет собой чередование α- и γ-элементов, причем начинается и заканчивается γ-элементом, т.е. в π^j элементов из A'' на один больше, чем из A, и

$$\sum_{i \in \pi^j} x_i = 0, \ \forall i \in \mathbb{N}_m. \tag{11.7}$$

Пусть $A_j = \pi^j \cap A, \ |A_j| = l$. Тогда из (11.7) получаем

$$\sum_{i \in \pi^j} x_i = (l+1)K + 2\sum_{i \in A_j} s(i) - lK - 2\lfloor \mu K \rfloor = 2\sum_{i \in A_j} s(i) + K - 2\lfloor \mu K \rfloor = 0,$$

откуда, сравнивая с (11.5), получаем $\sum_{i \in A_j} s(i) = B$. Таким образом, $\{A_1, \ \ldots, A_m\}$ является искомым разбиением, удовлетворяющим (11.2).

Теорема 11.4 доказана. □

Поскольку точное решение задачи КСВ представляет некоторую "NP-трудность", мы займемся приближенным ее решением. Следствие 11.8 доказываемой ниже теоремы 11.5 сообщает о возможности приближенного решения задачи КСВ полиномиальным алгоритмом с гарантированной оценкой точности. Сама теорема 11.5 не оперирует какой-либо нормой, а формулируется в терминах компактного суммирования семейства векторов с нулевой суммой в некоторой выпуклой ограниченной области пространства $\mathbb{R}^m$.

Теорема 11.5 *Для любого 0-семейства векторов $\{x_1, \ldots, x_n\} = X \subset \mathbb{R}^m$ и вектора $a \in \mathbb{R}^m$ с трудоемкостью $O(n^2m^2)$ находится перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что*

$$\sum_{i=1}^{k} x_{\pi_i} \in (m-1)B(X) + B_a, \ k \in \mathbb{N}_n, \tag{11.8}$$

где $B(X) = \mathrm{conv}\{x_1, \ldots, x_n\}, \ B_a = \mathrm{conv}\{0, a - \frac{1}{m}B(X)\}$.

Прежде чем перейти к доказательству, сформулируем несколько следствий этой теоремы.

Следствие 11.6 *Пусть $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m, \ \bar{x} = \sum_{i=1}^{n} x_i/n$. Тогда для любого вектора $a \in \mathbb{R}^m$ с трудоемкостью $O(n^2m^2)$ находится перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что*

$$\sum_{i=1}^{k} x_{\pi_i} - (k-m+1)\bar{x} \in (m-1)B(X) + B_a, \ k \in \mathbb{N}_n. \tag{11.9}$$

Доказательство. Пусть $B' = B(X) - \bar{x};\ x'_i = x_i - \bar{x},\ i \in \mathbb{N}_n;\ X' = \{x'_1, \ldots, x'_n\};\ a' = a - \bar{x}/m$. Тогда $B' = B(X')$ и $\sum_{i=1}^n x'_i = 0$. По теореме 11.5 находим перестановку $\pi = (\pi_1, \ldots, \pi_n)$ такую, что

$$\begin{aligned}\sum_{i=1}^k x'_{\pi_i} &\in (m-1)B(X') + \operatorname{conv}\{0, a' - \frac{1}{m}B(X')\} \\ &= (m-1)(B(X) - \bar{x}) + \operatorname{conv}\{0, a' - \frac{1}{m}(B(X) - \bar{x})\} \\ &= (m-1)B(X) - (m-1)\bar{x} + \operatorname{conv}\{0, a - \frac{1}{m}B(X)\}.\end{aligned}$$

Подставляя $\sum_{i=1}^k x'_{\pi_i} = \sum_{i=1}^k x_{\pi_i} - k\bar{x}$, получаем (11.9). □

Ясно, что теорема 11.5 и следствие 11.6 останутся справедливыми, если множество $B(X)$ заменить на включающее его множество Ψ, а B_a — на $\Psi_a = \operatorname{conv}\{0, a - \frac{1}{m}\Psi\}$.

Следствие 11.7 *Пусть векторы $\{x_1, \ldots, x_n\}$ содержатся в некотором выпуклом центрально-симметричном (не обязательно относительно нуля) множестве $\Psi \subset \mathbb{R}^m$ и их сумма равна нулю. Тогда с трудоемкостью $O(n^2m^2)$ находится перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что*

$$\sum_{i=1}^k x_{\pi_i} \in \left(m - 1 + \frac{1}{m}\right)\Psi,\ k \in \mathbb{N}_n.$$

Доказательство. Центрально-симметричное множество Ψ представимо в виде $\Psi = b + \Psi'$, где $\Psi' = -\Psi'$. Отсюда $2b - \Psi = 2b - b - \Psi' = b + \Psi' = \Psi$. Выбрав в теореме 11.5 вектор $a = \frac{2}{m}b$, получим $a - \frac{1}{m}\Psi = \frac{1}{m}(2b - \Psi) = \frac{1}{m}\Psi$, откуда с учетом $0 \in \Psi$ имеем $\Psi_a = \operatorname{conv}\{0, \frac{1}{m}\Psi\} = \frac{1}{m}\Psi$, и $(m-1)\Psi + \Psi_a = (m - 1 + \frac{1}{m})\Psi$. □

В частном случае, когда Ψ — единичный шар некоторой нормы, получаем

Следствие 11.8 *Для задачи* КСВ *в пространстве $\mathbb{R}^m$ с произвольной нормой s существует алгоритм, который для всякого s-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ с трудоемкостью $O(n^2m^2)$ находит перестановку $\pi = (\pi_1, \ldots, \pi_n)$ с оценкой*

$$f_{s,X}(\pi) \le m - 1 + \frac{1}{m}. \quad \square \tag{11.10}$$

Из следствия 11.7 вытекает также

Следствие 11.9 *Пусть векторы $\{x_1, \ldots, x_n\}$ содержатся в некотором выпуклом центрально-симметричном (не обязательно относительно нуля) множестве Ψ; $x = \sum_{i=1}^n x_i/n$. Тогда с трудоемкостью $O(n^2m^2)$ находится перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что*

$$\sum_{i=1}^k x_{\pi_i} - \left(k - m + 1 - \frac{1}{m}\right)x \in \left(m - 1 + \frac{1}{m}\right)\Psi,\ k \in \mathbb{N}_n. \quad \square$$

(Доказательство аналогично доказательству следствия 11.6.)

Теорема 11.10 *Пусть $\mathbb{R}^m$ — нормированное пространство, $m > 1$, Ψ — выпуклое неограниченное множество в $\mathbb{R}^m$, имеющее объемный рецессивный конус $0^+\Psi$. Тогда существует алгоритм $\mathcal{A}$, который для всякого семейства векторов $X = \{x_1, \ldots, x_n\} \subset \Psi$ с трудоемкостью $O(n^2m^2)$ находит перестановку $\pi = (\pi_1, \ldots, \pi_n)$ такую, что*

$$\sum_{i=1}^{k} x_{\pi_i} - (k - m + 1)\bar{x} \in (m-1)\Psi,\ k \in \mathbb{N}_n,$$

где $\bar{x} = \sum_{j=1}^{n} x_j/n$.

Доказательство. Пусть $B(X)$ — выпуклая оболочка семейства X. Т.к. множество $-\frac{1}{m}B(X)$ ограничено, а конус $0^+\Psi$ объемный, то согласно утверждению 9.6 (46) существует вектор a такой, что $B_a \doteq a - \frac{1}{m}B(X) \subset 0^+\Psi$. Согласно следствию 11.6 находим перестановку $\pi = (\pi_1, \ldots, \pi_n)$ такую, что для любого $k \in \mathbb{N}_n$, выполнены соотношения

$$\sum_{i=1}^{k} x_{\pi_i} - (k - m + 1)\bar{x} \in (m-1)B(X) + B_a \subset (m-1)\Psi + 0^+\Psi \subset (m-1)\Psi. \qquad \square$$

В задачах суммирования векторов, возникающих из моделей теории расписаний (см. главу III), семейство векторов X, как правило, содержится в некотором полиэдре Ψ, описываемом системой линейных неравенств:

$$\Psi = \{x \in \mathbb{R}^m \mid (c_i, x) \le \beta_i,\ i \in \mathbb{N}_l\}.$$

Для нас представляет интерес следующий частный случай только что доказанной теоремы.

Следствие 11.11 *Пусть заданы вектор $c \in \mathbb{R}^m$, $c \neq 0$; l векторов $c_i \in \mathbb{R}^m$ таких, что $(c_i, c) < 0$, $i \in \mathbb{N}_l$; l чисел $\{\beta_1, \ldots, \beta_l\}$ и полиэдр $\Psi \subset \mathbb{R}^m$, определяемый системой из l неравенств:*

$$\Psi = \{x \in \mathbb{R}^m \mid (c_i, x) \le \beta_i,\ i \in \mathbb{N}_l\}.$$

Тогда существует алгоритм $\mathcal{A}$, который для всякого семейства векторов $X = \{x_1, \ldots, x_n\} \subset \Psi$ с трудоемкостью $O(n^2m^2)$ находит перестановку $\pi = (\pi_1, \ldots, \pi_n)$ такую, что

$$\sum_{i=1}^{k} x_{\pi_i} - (k - m + 1)\bar{x} \in (m-1)\Psi,\ k \in \mathbb{N}_n,$$

где $\bar{x} = \sum_{j=1}^{n} x_j/n$. $\square$

(Вытекает из теоремы 11.10 и утверждения 9.8, стр. 46.)

Определение 11.12 Пусть B — выпуклое множество в $\mathbb{R}^m$, $0 \in B$, $a \in \mathbb{R}^m$, N — конечное множество, $|N| \geq m$. Семейство векторов $\{u_i \,|\, i \in N\} \subset B$ назовем *B_a-сбалансированной системой*, если существует набор чисел $\{\lambda_i \in \mathbb{R} \,|\, i \in N\}$, удовлетворяющий условиям

$$\lambda_i \in [0,1],\ i \in N, \tag{11.11}$$

$$\sum_{i \in N} \lambda_i = |N| - (m-1), \tag{11.12}$$

$$\sum_{i \in N} \lambda_i u_i \in B_a. \tag{11.13}$$

Доказательство теоремы 11.5 будет существенно опираться на следующую лемму.

Лемма 11.13 *Пусть B — выпуклое множество в $\mathbb{R}^m$, $0 \in B$, $a \in \mathbb{R}^m$, N — конечное множество, $|N| = k \geq m+1$, $\{u_i \,|\, i \in N\}$ — B_a-сбалансированная система векторов, для которой известен набор чисел $\{\lambda_i \,|\, i \in N\}$, удовлетворяющий* (11.11)–(11.13). *Тогда существует индекс $j \in N$ такой, что система векторов $\{u_i\}_{i \neq j}$ вновь является B_a-сбалансированной. Этот индекс j и набор чисел $\{\lambda'_j\}$, соответствующий системе $\{u_i\}_{i \neq j}$, могут быть найдены с трудоемкостью $O(km^2)$.*

Доказательство. Определим $U = \{\bar{u}_i = (u_i, 1) \,|\, i \in N\} \subset \mathbb{R}^{m+1}$. Положим $\lambda'_i = (k-m)\lambda_i/(k-m+1),\ i \in N$. Тогда

$$\lambda'_i \in [0,1],\ i \in N, \tag{11.14}$$

$$\sum_{i \in N} \lambda'_i = k - m, \tag{11.15}$$

$$\sum_{i \in N} \lambda'_i u_i \in B_a. \tag{11.16}$$

Преобразуем коэффициенты $\{\lambda'_i\}$ с сохранением свойств (11.14)–(11.16) так, что один из коэффициентов (λ'_j) обратится в нуль, и значит система векторов $\{u_i\}_{i \neq j}$ будет B_a-сбалансированной. Искомое преобразование потребует не более трех действий, описываемых ниже.

Действие 1. Преобразуем коэффициенты $\{\lambda'_i\}$ с сохранением условия (11.14) и суммы $\sum_{i \in N} \lambda'_i \bar{u}_i$ так, чтобы подмножество векторов $U' \doteq \{\bar{u}_i \in U \,|\, \lambda'_i \in (0,1)\}$ содержалось в базе U^* множества U. Согласно лемме 10.2, искомые коэффициенты $\{\lambda'_i\}$ и база $U^* \subseteq U$ могут быть найдены с трудоемкостью $O(km^2)$.

Если среди коэффициентов $\{\lambda'_i\}$ найдется $\lambda'_i = 0$, то система векторов $\{u_i\}_{i \neq j}$ является B_a-сбалансированной, а набор коэффициентов $\{\lambda'_i\}_{i \neq j}$ — искомым. Пусть

$$\lambda'_i > 0,\ \forall i \in N. \tag{11.17}$$

Обозначим $I = \{i \in N \,|\, \lambda'_i \in (0,1)\}$. Поскольку $|I| = |U'| \le |U^*| \le m+1$, то $|\{i \in N \,|\, \lambda'_i = 1\}| \ge k-m-1$, и из (11.15) и (11.17) следует, что здесь имеет место равенство, откуда

$$|U'| = |U^*| = m+1,\ \sum_{i\in I} \lambda'_i = 1.$$

Покажем, что коэффициенты $\{\lambda'_i \,|\, i \in I\}$ можно преобразовать с сохранением свойств (11.14)–(11.16) так, что один из коэффициентов обратится в нуль. Введем обозначения:

$$c = -\sum_{i\in N\setminus I} u_i,\ \bar{c} = (c,1) \in \mathbb{R}^{m+1},\ b = \sum_{i\in N} \lambda'_i u_i \in B_a.$$

Действие 2. Найдем представление $\bar{c} = \sum_{i\in I} \mu_i \bar{u}_i$. Такое представление существует и единственно, поскольку векторы $\{\bar{u}_i \,|\, i \in I\} = U' = U^*$ образуют базис пространства $\mathbb{R}^{m+1}$. Коэффициенты $\{\mu_i\}$ могут быть найдены, например, методом Гаусса с трудоемкостью $O(m^3)$ (нам не требуется более быстрых методов решения системы линейных уравнений, поскольку уже $O(m^3) \le O(km^2)$). Если

$$I' = \{i \in I \,|\, \mu_i \le 0\} \ne \emptyset, \tag{11.18}$$

то найдем $\varepsilon = \max_{i\in I'}(-\mu_i/(\lambda'_i - \mu_i))$. Очевидно, $\varepsilon \in [0,1)$,

$$\lambda''_i = (1-\varepsilon)\mu_i + \varepsilon\lambda'_i \ge 0,\ \forall i \in I,$$

и существует $j \in I$ такое, что $\lambda''_i = 0$. Из равенств $\sum_{i\in I} \mu_i = 1$, $\sum_{i\in I} \lambda'_i = 1$ следует, что $\sum_{i\in I} \lambda''_i = 1$. Положим

$$\lambda_i = \begin{cases} 1 & , i \in N \setminus I, \\ \lambda''_i & , i \in I \setminus \{j\}. \end{cases}$$

Поскольку

$$\begin{aligned} \sum_{i\in N\setminus\{j\}} \lambda_i u_i &= \sum_{i\in N\setminus I} u_i + \sum_{i\in I} \lambda''_i u_i = -c + \sum_{i\in I}[(1-\varepsilon)\mu_i + \varepsilon\lambda'_i]u_i = \\ &= -c + (1-\varepsilon)c + \varepsilon(c+b) = \varepsilon b \in B_a, \end{aligned}$$

то система векторов $\{u_i\}_{i\ne j}$ является B_a-сбалансированной, а коэффициенты $\{\lambda_i\}_{i\ne j}$ — искомыми.

Пусть (11.18) не выполняется, т.е. $\mu_i \in (0,1),\ \forall i \in I$.

Действие 3. Найдем представление вектора $\bar{a} = (ma, 1) \in \mathbb{R}^{m+1}$ в виде $\bar{a} = \sum_{i\in I} \eta_i \bar{u}_i$. Поскольку $\sum_{i\in I}(m\mu_i + \eta_i) = m+1$, то найдется $i_0 \in I$ такое, что $m\mu_{i_0} + \eta_{i_0} \le 1$. Так как $\mu_{i_0} > 0$, то справедливо $\eta_{i_0} < 1$ и

$$r \doteq \frac{\mu_{i_0}}{1-\eta_{i_0}} \le \frac{1}{m}. \tag{11.19}$$

Вектор $c - ru_{i_0}$ имеет представление

$$c - ru_{i_0} = c - \frac{\mu_{i_0} u_{i_0}}{1 - \eta_{i_0}} = c - \mu_{i_0} u_{i_0} - \frac{\mu_{i_0}\eta_{i_0} u_{i_0}}{1 - \eta_{i_0}} =$$

$$= \sum_{i \in I \setminus \{i_0\}} \mu_i u_i + \mu_{i_0} \sum_{i \in I \setminus \{i_0\}} \eta_i u_i / \sum_{i \in I \setminus \{i_0\}} \eta_i - mra \doteq \sum_{i \in I} \mu'_i u_i - mra,$$

где $\sum_{i \in I} \mu'_i = 1$ и $\mu'_{i_0} = 0$.

Если $\mu'_i \geq 0,\ \forall i \in I$, то, используя (11.19), получим

$$\sum_{i \in N \setminus I} u_i + \sum_{i \in I \setminus \{i_0\}} \mu'_i u_i = -c + c - ru_{i_0} + mra = mr\left(a - \frac{1}{m} u_{i_0}\right) \in B_a,$$

а система векторов $\{u_i\}_{i \in N \setminus \{i_0\}}$ является B_a-сбалансированной.

Пусть $I'' = \{i \in I \,|\, \mu'_i < 0\} \neq \emptyset$. Определим ε по формуле $\varepsilon = \max_{i \in I''}(-\mu'_i/(\mu_i - \mu'_i))$. Тогда $\varepsilon \in (0, 1)$,

$$\lambda''_i \doteq (1 - \varepsilon)\mu'_i + \varepsilon\mu_i \geq 0,\ \forall i \in I,$$
$$\exists j \in I :\ \lambda''_i = 0,$$
$$\textstyle\sum_{i \in I} \lambda''_i = 1.$$

Положим

$$\lambda_i = \begin{cases} 1 & , i \in N \setminus I, \\ \lambda''_i & , i \in I \setminus \{j\}. \end{cases}$$

Тогда

$$\sum_{i \in N \setminus \{j\}} \lambda_i u_i = \sum_{i \in N \setminus I} u_i + \sum_{i \in I} \lambda''_i u_i = -c + \sum_{i \in I} [(1 - \varepsilon)\mu'_i + \varepsilon\mu_i] u_i =$$

$$= -c + (1 - \varepsilon)(c - ru_{i_0} + mra) + \varepsilon c = (1 - \varepsilon) mr\left(a - \frac{1}{m} u_{i_0}\right) \in B_a,$$

и система векторов $\{u_i\}_{i \neq j}$ является B_a-сбалансированной.

Для завершения доказательства леммы 11.13 остается заметить, что трудоемкость действий 2 и 3 есть $O(m^3)$ и не превосходит трудоемкости действия 1, равной $O(km^2)$. Лемма 11.13 доказана. □

Доказательство теоремы 11.5. Индукцией по $k = n, n-1, \ldots, 1$ построим перестановку $\pi = (\pi_1, \ldots, \pi_n)$ такую, что системы векторов $\{x_i \,|\, i \in N_k^\pi = \mathbb{N}_n \setminus \{\pi_{k+1}, \ldots, \pi_n\}\}$ являются B_a-сбалансированными для всех $k = m, \ldots, n$.

Пусть $k = n$. Чтобы убедиться в B_a-сбалансированности системы $\{x_i \,|\, i \in \mathbb{N}_n\}$, достаточно положить $\lambda_i = (n - (m-1))/n,\ \forall i \in \mathbb{N}_n$. Тогда

$$\lambda_i \in [0, 1],\ \sum \lambda_i = n - (m-1),\ \sum \lambda_i x_i = 0 \in B_a.$$

Шаг $k \to k-1 (k > m)$. Пусть индексы $\{\pi_i \,|\, i = k+1, \ldots, n\}$ определены и система $\{x_i \,|\, i \in N_k^\pi\}$ является B_a-сбалансированной. Согласно лемме 11.13 находим индекс $j \in N_k^\pi$, для которого система $\{x_i \,|\, i \in N_k^\pi \setminus \{j\}\}$ остается B_a-сбалансированной, и полагаем $\pi_k = j$. Одновременно находим набор чисел $\{\lambda_i\}$, удовлетворяющий (11.11)–(11.13) для новой системы векторов.

Для $k \in \mathbb{N}_{m-1}$ значения перестановки π доопределяем произвольно.

Из B_a-сбалансированности системы $\{x_{\pi_i} \,|\, i \in \mathbb{N}_k\}$ (при $k \geq m$) следует, что

$$\begin{aligned}\sum_{i\in\mathbb{N}_k} x_{\pi_i} &= \sum_{i\in\mathbb{N}_k} (1-\lambda_{\pi_i}) x_{\pi_i} + \sum_{i\in\mathbb{N}_k} \lambda_{\pi_i} x_{\pi_i} \in \sum_{i\in\mathbb{N}_k} (1-\lambda_{\pi_i}) B + B_a = \\ &= (m-1)B + B_a.\end{aligned}$$

При $k \leq m-1$, очевидно,

$$\sum_{i\in\mathbb{N}_k} x_{\pi_i} \in (m-1)B + B_a,$$

т.е. (11.8) выполняется. Для завершения доказательства теоремы остается заметить, что поскольку трудоемкость каждого шага индукции составляет $O(km^2)$, то трудоемкость всего алгоритма не превосходит $O(n^2m^2)$

Теорема 11.5 доказана. □

12 Оценки и свойства функций Штейница

В задаче КСВ из предыдущего раздела для заданного s-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ мы отыскивали перестановку $\pi = (\pi_1, \ldots, \pi_n)$, минимизирующую функционал $f_{s,X}(\pi)$ (то бишь, радиус шара нормы s, внутри которого укладывается траектория суммирования векторов из X в соответствии с перестановкой π). В связи с этим было бы интересно знать, на какую наилучшую оценку радиуса мы можем рассчитывать в худшем случае? Таким образом, нас интересуют точные значения *функций Штейница*, определяемых для различных норм s в пространстве $\mathbb{R}^m$ по формуле

$$\varphi_s(m) \doteq \sup_{X \subset B_{m,s}} \min_{\pi} f_{s,X}(\pi).$$

Здесь мы должны признаться, что так поставленный вопрос оказался нам **пока** не по зубам. (Причем, "слабоватые зубы" оказались не только у автора диссертации, но и многих, интересующихся этой проблемой, — начиная с 1917 года, когда Гросс для евклидовой нормы в $\mathbb{R}^2$ получил оценку $\varphi_{l_2}(2) \leq \sqrt{2}$ и поставил вопрос о нахождении функции $\varphi_{l_2}(m)$.) На сегодняшний день точные значения функций Штейница при $m \geq 2$ известны (благодаря результатам В. Банащика [70], [71]) лишь в трех случаях:

$$\varphi_{l_2}(2) = \sqrt{5/4}; \tag{12.1}$$

$$\varphi_{l_1}(2) = \varphi_{l_\infty}(2) = 3/2. \tag{12.2}$$

Для других норм и значений размерности пространства известно некоторое количество верхних и нижних оценок функций Штейница и их самые общие свойства. Значительная часть этих результатов была получена автором диссертации и будет приведена ниже. Так например, в [39] было доказано простое, но довольно важное свойство аффинной инвариантности функций Штейница.

Теорема 12.1 *Пусть s — норма в $\mathbb{R}^m$, A — невырожденное аффинное преобразование пространства $\mathbb{R}^m$, s_A — норма с единичным шаром $B_{m,s_A} \doteq A(B_{m,s})$. Тогда* $\varphi_{s_A}(m) = \varphi_s(m)$. □

Как видно из рисунка 12.1, единичные шары B_{2,l_1} и B_{2,l_∞} аффинно подобны, откуда по теореме 12.1 вытекает равенство $\varphi_{l_1}(2) = \varphi_{l_\infty}(2)$.

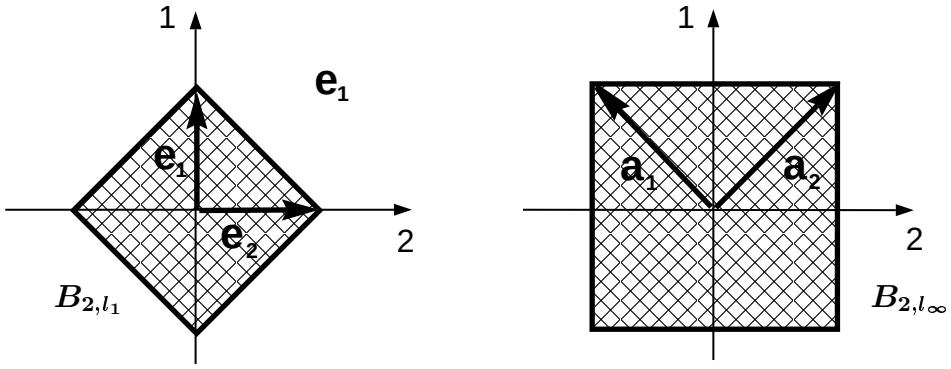


Рис. 12.1: Единичные шары норм l_1 и l_∞ в $\mathbb{R}^2$.

Из следствия 11.8 теоремы 11.5 непосредственно вытекает оценка

$$\varphi_s(m) \le m - 1 + \frac{1}{m}\,, \tag{12.3}$$

справедливая для любого целого $m \ge 1$ и любой нормы s в пространстве $\mathbb{R}^m$.

Первый естественный вопрос, который возникает при взгляде на оценку (12.3): действительно ли функции $\varphi_s(m)$ зависят от размерностей пространства? Быть может, найдутся какая-то норма s и достаточно большая константа C, не зависящая от m, такие что всякое s-семейство ("коротких" — длины не более 1) векторов может быть просуммировано в шаре радиуса C? Мы пока не можем ответить на этот вопрос в общем случае — для произвольных норм s[56]. Тем не менее, нам известно, что для всех

[56]Как будет видно из результатов раздела 20, ответ на этот вопрос зависит от ответа на аналогичный вопрос о величине функции Дворецкого $D_s(m)$. Последний нам представляется более простым, поскольку не требует рассмотрения всевозможных перестановок из n векторов в $\mathbb{R}^m$, а оперирует лишь с **подмножествами** заданного семейства векторов.

норм $l_p,\ p \geq 1$, ответ на этот вопрос отрицателен. Такой вывод следует из приводимых далее нижних оценок функций $\varphi_{l_p}(m),\ p \geq 1$.

Нахождение нижних оценок функций $\varphi_s(m)$ основывается на построении "плохих" s-семейств векторов. Одним из возможных методов построения таких семейств является

Метод 1.

На основе некоторого семейства векторов $\{a_1, \ldots, a_t\} \subset B_{m,s}$ (которое будем называть *основным*) определяется s-семейство $X = \{x_1, \ldots, x_n\}$, где $n = t - 1 + 2N,\ N$ — большое нечетное число:

- $t-1$ векторов $\{x_i \,|\, i = 1, \ldots, t-1\}$ совпадают с $a_1, \ldots, a_{t-1}$;
- N векторов $\{x_i \,|\, i = t, \ldots, t-1+N\}$ равны вектору $b' \doteq -\varepsilon \sum_{i=1}^{t-1} a_i + (1-(t-1)\varepsilon)a_t$, где $\varepsilon = \frac{1}{2N}$;
- N векторов $\{x_i \,|\, i = t+N, \ldots, n\}$ равны вектору $b'' \doteq -\varepsilon \sum_{i=1}^{t-1} a_i - (1-(t-1)\varepsilon)a_t$. ■

Так как вектор b' равен выпуклой комбинации векторов $\{-a_1, \ldots, -a_{t-1}, a_t\} \subset B_{m,s}$, то $b' \in B_{m,s}$; аналогично, $b'' \in B_{m,s}$. Кроме того, очевидно, $\sum x_i = 0$, поэтому X является s-семейством в $\mathbb{R}^m$.

Пусть $\pi = (\pi_1, \ldots, \pi_n)$ — произвольная перестановка индексов из $\mathbb{N}_n$, l — такой минимальный индекс, что $\{\pi_1, \ldots, \pi_l\}$ содержит ровно N индексов $i \geq t$. Тогда частичная сумма $x_\pi^l \doteq \sum_{i=1}^{l} x_{\pi_i}$ имеет вид $x_\pi^l = \frac{1}{2}\sum_{i=1}^{t-1} \lambda_i a_i + k(1-(t-1)\varepsilon)a_t$, где $\lambda_i = \pm 1,\ k$ — нечетное (положительное или отрицательное). Отсюда получаем оценку

$$\varphi_s(m) \geq \sup_{\{a_1,\ldots,a_t\}\subset B_{m,s}} \min_{k\ -\ \text{нечет.}} \min_{\lambda_i=\pm 1} \left\| \frac{1}{2}\sum_{i=1}^{t-1} \lambda_i a_i + k(1-(t-1)\varepsilon)a_t \right\|_s$$

$$= \frac{1}{2} \sup_{\{a_1,\ldots,a_t\}\subset B_{m,s}} \min_{k\in\mathbb{N}} \min_{\lambda_i=\pm 1} \left\| \sum_{i=1}^{t-1} \lambda_i a_i + (4k-2)a_t \right\|_s \tag{12.4}$$

Выбирая в качестве векторов $\{a_i\}$ m единичных орт в $\mathbb{R}^m$, для норм l_p $(p \geq 1)$ из (12.4) получаем оценку

$$\varphi_{l_p}(m) \geq \frac{1}{2}(m - 1 + 2^p)^{1/p}. \tag{12.5}$$

В частности, при $m = 2$ имеем оценку

$$\varphi_{l_p}(2) \geq \frac{1}{2}(1 + 2^p)^{1/p}. \tag{12.6}$$

Нетрудно видеть, что равенство (12.2) является следствием теоремы 12.1, оценки (12.3) при $m = 2,\ s = l_1$ и оценки (12.6) при $p = 1$.

Близость полученной нами нижней оценки (12.5) функции $\varphi_{l_p}(m)$ к верхней оценке (12.3) наблюдается лишь для нормы l_1:

$$\varphi_{l_1}(m) \geq \frac{m+1}{2}.$$

Уже для нормы l_2 получается оценка

$$\varphi_{l_2}(m) \geq \frac{\sqrt{m+3}}{2}, \tag{12.7}$$

отличающаяся по порядку от верхней оценки (12.3), а для нормы l_∞ оценка (12.5) вырождается в тривиальную: $\varphi_{l_\infty}(m) \geq 1$. Что касается оценки (12.7), то нам представляется, что она по порядку близка к точной. (А следовательно, есть необходимость в разработке алгоритма суммирования векторов, лучше учитывающего специфику конкретной нормы и гарантирующего лучшую по порядку оценку точности по сравнению с оценкой (11.10).) В то же время, оценку $\varphi_{l_\infty}(m) \geq 1$ (как мы вскоре убедимся) никак нельзя признать точной. Слабость этой оценки объясняется неудачным выбором исходного семейства векторов $\{a_1, \ldots, a_t\} \subset B_{m,l_\infty}$. Из рисунка 12.1 видно, что в случае пространства $\mathbb{R}^2$ и нормы l_∞ бóльшие значения величины в правой части (12.4) достигаются при выборе m наиболее длинных (в норме l_∞) взаимно ортогональных векторов: $a_1 = (1, -1);\ a_2 = (1, 1)$. Для произвольной l_p-нормы ($p \geq 2$) это должны быть векторы $(1, -1)$ и $(1, 1)$, умноженные на коэффициент $2^{-1/p}$. Для таких векторов $\{a_i\}$ правая часть (12.4) достигает минимума при $k = 1$, и оценка (12.4) превращается в

$$\varphi_{l_p}(2) \geq \frac{1}{2}\left(\frac{3^p+1}{2}\right)^{1/p}. \tag{12.8}$$

Не удивительно, что при изменении p от 1 до ∞ оценка (12.8) ведет себя по сравнению с (12.6) с точностью до наоборот: при $p = 1$ оценка вырождается в тривиальную ($\varphi_{l_1}(2) \geq 1$), затем функция от p в правой части (12.8) монотонно возрастает и дает наилучшую оценку $\varphi_{l_\infty}(2) \geq 3/2$ при $p = \infty$. При $p = 2$ оценки (12.5) и (12.8) "встречаются", т.е. дают одинаковую оценку $\varphi_{l_2}(2) \geq \sqrt{5/4}$. С учетом равенства (12.1), оценки (12.3) при $m = 2$, монотонного убывания функции в правой части (12.6) и монотонного возрастания функции в правой части (12.8) получаем следующий результат.

Теорема 12.2 *Для норм l_p ($p \geq 1$) в пространстве $\mathbb{R}^2$ выполняются следующие оценки функций Штейница:*

$$\varphi_{l_p}(2) \geq \begin{cases} \frac{1}{2}(1+2^p)^{1/p}, & \text{при } p \in [1,2]; \\ \frac{1}{2}\left(\frac{3^p+1}{2}\right)^{1/p}, & \text{при } p \in [2,\infty]. \end{cases}$$

При этом минимум значений $\varphi_{l_p}(2)$ по всем $p \geq 1$ равен $\sqrt{5/4}$ и достигается на евклидовой норме, а максимум значений $\varphi_{l_p}(2)$ по всем $p \geq 1$ равен $3/2$ и достигается на нормах l_1 и l_∞. □

Пусть s — произвольная норма; $R = \max_{x \in B_{m,s}} \|x\|_{l_2}$. В методе 1 выберем в качестве основного семейство из m попарно ортогональных векторов $\{a_1, \ldots, a_m\} \subset B_{m,s}$ так, что $\|a_m\|_{l_2} = R$. Тогда, оценивая s-норму частичной суммы x_π^l неравенством $\|x_\pi^l\|_s \geq \frac{1}{R}\|x_\pi^l\|_{l_2}$, из (12.4) получим оценку

$$\varphi_s(m) \geq \min_{k \in \mathbb{N}} \frac{1}{2R}\left(\sum_{i=1}^{m-1} \|a_i\|_{l_2}^2 + (4k-2)^2 R^2\right)^{1/2} \geq \frac{1}{2}\left(\sum_{i=1}^{m-1} \left\|\frac{a_i}{R}\right\|_{l_2}^2 + 4\right)^{1/2}. \tag{12.9}$$

Далее используем следующий известный результат.

Теорема 12.3 (John [92], Leichtweiss [98]) *Для любой нормы s в $\mathbb{R}^m$ существует такое аффинное преобразование A пространства $\mathbb{R}^m$, что*

$$B_{m,l_2} \subseteq A(B_{m,s}) \subseteq \sqrt{m}\, B_{m,l_2}. \quad \square \tag{12.10}$$

Пусть A — аффинное преобразование, гарантирующее свойство (12.10) для нормы s. Так как по теореме 12.1 имеем $\varphi_s(m) = \varphi_{s_A}(m)$, (где s_A — норма, определяемая шаром $A(B_{m,s})$), то вместо функции $\varphi_s(m)$ достаточно оценивать $\varphi_{s_A}(m)$. Из (12.10) имеем $\|a_i\|_{l_2} \geq 1$ $(i = 1, \ldots, m-1)$ и $R \leq \sqrt{m}$. Подставляя эти оценки в (12.9), получим

$$\varphi_s(m) \geq \frac{1}{2}\left(\frac{m-1}{m} + 4\right)^{1/2} = \frac{1}{2}\sqrt{5 - 1/m}.$$

Из этой оценки при $m = 2$ получаем следующий результат.

Теорема 12.4 *Для любой нормы s в $\mathbb{R}^2$ верна оценка $\varphi_s(2) \geq \sqrt{9/8}$.* $\square$

Из (12.9) мы можем заключить, что нижняя оценка функции $\varphi_s(m)$ тем лучше, чем ближе норма векторов a_i к R. В частности, для нормы l_∞ имеем $R = \max_{x \in B_{m,l_\infty}} \|x\|_{l_2} = \sqrt{m}$, и максимум достигается на векторах x, имеющих координаты $x(j) = \pm 1$, $\forall\, j \in \mathbb{N}_m$. Таким образом, наилучшая оценка для $\varphi_{l_\infty}(m)$ получается при выборе такого основного семейства векторов $\{a_1, \ldots, a_m\} \subset B_{m,l_\infty}$, что векторы a_i попарно ортогональны и $a_i(j) = \pm 1$, $\forall\, i, j$. Семейства векторов с такими свойствами предоставляют нам матрицы Адамара (чьи строки берутся в качестве искомых векторов). Известно однако, что матрицы Адамара (A_n) порядка $n > 2$ существуют лишь для n, кратных 4. Причем до сих пор остается недоказанной гипотеза Адамара о том, что **для любого** n, кратного 4, существует матрица Адамара порядка n. Пусть $n \leq m$ есть максимальный порядок, для которого существует матрица A_n. Тогда из (12.9) получим оценку

$$\varphi_{l_\infty}(m) \geq \varphi_{l_\infty}(n) \geq \frac{1}{2}\sqrt{n+3}. \tag{12.11}$$

Поскольку из (12.4) следует, что при таком выборе основного семейства векторов l_∞-норма частичной суммы x_π^l должна быть кратна $1/2$, то оценку (12.11) можно несколько улучшить:

$$\varphi_{l_\infty}(m) \geq \frac{1}{2}\left\lceil \sqrt{n+3} \right\rceil. \tag{12.12}$$

Для норм l_p $(p \geq 2)$ в качестве основного семейства $\{a_i\}$ берем строки матрицы Адамара A_n с коэффициентом $(n)^{-1/p}$. Тогда $\|a_i\|_{l_2} = R$, $\forall\, i$, и из (12.9) вновь получаем оценку

$$\varphi_{l_p}(m) \geq \frac{1}{2}\sqrt{n+3}. \tag{12.13}$$

* * *

Далее будет рассмотрен другой метод построения “плохих” семейств векторов.

Метод 2.

Пусть A_n — матрица Адамара порядка $n \le m+1$, $\{a_i \,|\, i \in \mathbb{N}_n\}$ — множество строк матрицы A_n. Без ограничения общности можем считать, что $a_i(1) = 1$, $\forall\ i \in \mathbb{N}_n$. (В противном случае вся строка заменяется на противоположную.) Рассмотрим векторы $x_i = (a_i(2), \ldots, a_i(n)) \in \mathbb{R}^{n-1}$, $i \in \mathbb{N}_n$. Обозначим $x = \sum_{i=1}^n x_i$, $a = \sum_{i=1}^n a_i$. Тогда ввиду попарной ортогональности векторов a_i будем иметь

$$(x,x) = (a,a) - a^2(1) = \sum_{i=1}^n (a_i, a_i) - n^2 = n^2 - n^2 = 0,$$

откуда $x = 0$. Таким образом, $\{x_i \,|\, i \in \mathbb{N}_n\}$ является l_∞-семейством векторов в $\mathbb{R}^{n-1}$, которое и берем в качестве искомого “плохого” семейства. ■

Лемма 12.5 *Если для* $n \le m+1$, $m \ge 3$, *существует матрица Адамара* A_n, *то*

$$\varphi_{l_p}(m) \ge \frac{1}{2}\sqrt{n+1+\frac{1}{n-1}},\ p \ge 2, \tag{12.14}$$

$$\varphi_{l_\infty}(m) \ge \left\lceil \frac{1}{2}\sqrt{n+4} \right\rceil. \tag{12.15}$$

Доказательство. Пусть $n \le m+1$ есть максимальный порядок, для которого существует матрица Адамара A_n. Так как $m \ge 3$, то n кратно 4. Сформируем l_∞-семейство векторов $\{x_1, \ldots, x_n\}$ по методу 2 и рассмотрим произвольную перестановку $\pi = (\pi_1, \ldots, \pi_n)$. Обозначим частичные суммы $\tilde{x} = \sum_{i=1}^{n/2} x_{\pi_i}$, $\tilde{a} = \sum_{i=1}^{n/2} a_{\pi_i}$. Тогда

$$\sum_{i=1}^{n-1} \tilde{x}^2(i) = (\tilde{x}, \tilde{x}) = (\tilde{a}, \tilde{a}) - \tilde{a}^2(1) = n^2/2 - n^2/4 = n^2/4. \tag{12.16}$$

Из (12.16) ввиду целочисленности координат вектора $\tilde{x}$ получаем

$$\|\tilde{x}\|_{l_\infty} = \max_i |\tilde{x}(i)| \ge \left\lceil \frac{n}{2\sqrt{n-1}} \right\rceil = \left\lceil \frac{1}{2}\sqrt{n+1+\frac{1}{n-1}} \right\rceil .$$

Так как эти соотношения верны для любой перестановки $\pi = (\pi_1, \ldots, \pi_n)$, то получаем оценку

$$\varphi_{l_\infty}(m) \ge \varphi_{l_\infty}(n-1) \ge \left\lceil \frac{1}{2}\sqrt{n+1+\frac{1}{n-1}} \right\rceil \doteq k.$$

Так как k есть минимальное целое, удовлетворяющее $k \ge \frac{1}{2}\sqrt{n+1+\frac{1}{n-1}}$, или $4k^2 \ge n+1+\frac{1}{n-1}$, а n кратно 4, то $4k^2 \ge n+4$, откуда $k = \left\lceil \frac{1}{2}\sqrt{n+4} \right\rceil$.

Для нормы l_p в качестве "плохого" l_p-семейства векторов берем $\{x'_i = \rho_p x_i \,|\, i \in \mathbb{N}_n\}$, где $\{x_1, \ldots, x_n\}$ — l_∞-семейство векторов, образованных по методу 2 при существовании матрицы A_n, $\rho_p = (n-1)^{-1/p}$. Пусть $R_p = \max_{x \in B_{m,l_p}} \|x\|_{l_2}$. Тогда для норм l_p, $p \geq 2$, имеем $R_p = \rho_p \sqrt{n-1}$. Для произвольной перестановки $\pi = (\pi_1, \ldots, \pi_n)$ и частичных сумм $x' = \sum_{i=1}^{n/2} x'_{\pi_i}$, $\tilde{x} = \sum_{i=1}^{n/2} x_{\pi_i}$ получаем соотношения

$$\|x'\|_{l_p} \geq \frac{1}{R_p}\|x'\|_{l_2} = \frac{\rho_p}{R_p}\|\tilde{x}\|_{l_2} \overset{\text{из (12.16)}}{=} \frac{\rho_p}{R_p} \cdot \frac{n}{2} = \frac{n}{2\sqrt{n-1}} = \frac{1}{2}\sqrt{n+1+\frac{1}{n-1}},$$

откуда $\varphi_{l_p}(m) \geq \varphi_{l_p}(n-1) \geq \frac{1}{2}\sqrt{n+1+\frac{1}{n-1}}$. □

Ясно, что при одном и том же n оценка (12.14) всегда хуже, чем (12.13). Однако если $m = 4k-1$ и существует матрица A_{m+1}, то из (12.14) получаем оценку

$$\varphi_{l_p}(m) \geq \frac{1}{2}\sqrt{m+2+\frac{1}{m}},$$

тогда как из (12.13) в лучшем случае (при существовании матрицы A_{m-3}) вытекает оценка $\varphi_{l_p}(m) \geq \frac{1}{2}\sqrt{m}$.

В то же время, оценка (12.12) при любом m проигрывает оценке (12.15).

Резюме сравнения оценок, полученных при $m \geq 3$, $p \geq 2$ обоими методами с использованием матриц Адамара приводится в следующей теореме.

Теорема 12.6 *Пусть $m \geq 3$, $p \geq 2$. Если при некотором $n \leq m$ существует матрица A_n, то*

$$\varphi_{l_p}(m) \geq \frac{1}{2}\sqrt{n+3}.$$

Если $m = 4k-1$ и существует матрица A_{m+1}, то

$$\varphi_{l_p}(m) \geq \frac{1}{2}\sqrt{m+2+\frac{1}{m}}.$$

Если матрица A_n существует для некоторого $n \leq m+1$, то

$$\varphi_{l_\infty}(m) \geq \left\lceil \frac{1}{2}\sqrt{n+4} \right\rceil. \qquad \square$$

Следствие 12.7 *Если верна гипотеза Адамара и матрицы A_n существуют для всех n, кратных 4, то для всех $m \geq 3$, $p \geq 2$, выполняются оценки*

$$\varphi_{l_p}(m) \geq \frac{1}{2} \cdot \begin{cases} \sqrt{m+3}, & \text{при } m = 4k, \\ \sqrt{m+2}, & \text{при } m = 4k+1, \\ \sqrt{m+1}, & \text{при } m = 4k+2, \\ \sqrt{m+2+\frac{1}{m}}, & \text{при } m = 4k+3. \end{cases}$$

$$\varphi_{l_\infty}(m) \geq \left\lceil \frac{1}{2}\sqrt{m+2} \right\rceil. \quad \square \tag{12.17}$$

Лемма 12.8 *Пусть* $n_1 = \max\{n \,|\, n \le m+1,\ n = 4i,\ \exists\ A_n\}$, $n_2 = \max\{n \,|\, n \le m+1,\ n = 4i^2,\ \exists\ A_n\}$. *Тогда оценки*

$$\varphi_{l_\infty}(m) \ge \left\lceil \frac{1}{2}\sqrt{n_1+4} \right\rceil,\ \varphi_{l_\infty}(m) \ge \left\lceil \frac{1}{2}\sqrt{n_2+4} \right\rceil,$$

вытекающие из теоремы 12.6, совпадают.

Доказательство. Пусть $n_2 = 4i^2$. Из определения n_1 и n_2 имеем $n_1 \le 4(i+1)^2 - 4$, откуда

$$\left\lceil \frac{1}{2}\sqrt{n_1+4} \right\rceil \le i+1 = \left\lceil \sqrt{i^2+1} \right\rceil = \left\lceil \frac{1}{2}\sqrt{n_2+4} \right\rceil.$$

С учетом очевидного неравенства $n_2 \le n_1$ получаем требуемое. □

Таким образом, оценка (12.17) может быть получена при выполнении ослабленной гипотезы Адамара.

Следствие 12.9 *Если матрица A_n существует для максимального $n \le m+1$ вида $n = 4i^2$, то справедлива оценка (12.17).* □

Следует отметить, что результаты теоремы 12.6 и следствий 12.7, 12.9 носят условный характер, поскольку оценки функций Штейница зависят от выполнения предположений о существовании матриц Адамара определенных порядков. Далее будут получены следствия этих результатов, имеющие безусловный характер.

Прежде всего, из результатов, приведенных в [134], [135], следует факт существования матриц A_{4i^2} при всех $i \le 32$. Таким образом, с учетом следствия 12.9 оценка (12.17) справедлива по крайней мере для всех $m \le 4 \cdot 33^2 - 2 = 4354$.

Для всех n вида $n = 2^k$ известны конкретные конструкции матриц A_n (так называемые *матрицы Сильвестра*). Таким образом, для любого m доказано существование матриц A_n порядка $n \in [(m+2)/2, m+1]$ и порядка $n \in [(m+1)/2, m]$. С учетом приведенных выше результатов получаем следующее

Следствие 12.10 *Для всех $m \ge 3,\ p \ge 2$ справедливы оценки*

$$\varphi_{l_p}(m) \ge \frac{1}{2\sqrt{2}}\sqrt{m+7} > 0.3535\sqrt{m+7}, \tag{12.18}$$

$$\varphi_{l_\infty}(m) \ge \left\lceil \frac{1}{2\sqrt{2}}\sqrt{m+10} \right\rceil > \left\lceil 0.3535\sqrt{m+10} \right\rceil. \qquad \square$$

Из оценки (12.18) и неравенств

$$\varphi_{l_p}(m) \ge \frac{1}{2}\sqrt{m+3},\ \forall\ p \in [1,2],$$

вытекающих из (12.5) и монотонности функции $(m-1+2^p)^{1/p}$, получаем следующий результат.

Теорема 12.11 $\inf_{p\geq 1}\varphi_{l_p}(m) \underset{m\to\infty}{\longrightarrow} \infty$. □

Хотя гипотеза Адамара в общем виде пока не доказана, на сегодняшний день известно, что множество целых n, для которых существуют матрицы A_n, является плотным на целочисленной решетке.

Теорема 12.12 (см. [65], стр. 194) *Для любого $\varepsilon > 0$ существует n_0 такое, что для всякого $n \geq n_0$ существует матрица $A_{n'}$ порядка $n' \in ((1-\varepsilon)n, n]$.* □

Из теорем 12.6, 12.12 и оценки (12.5) вытекает следующий результат.

Теорема 12.13 *Для любого $\varepsilon > 0$ существует такое целое m_0, что для всякого $m \geq m_0$ выполнена оценка*

$$\varphi_{l_p}(m) \geq \left(\frac{1}{2} - \varepsilon\right)\sqrt{m},\ \forall\ p \geq 1.$$ □

13 Нестрогое суммирование векторов на плоскости

В этом разделе вводится понятие нестрогого суммирования векторов в ограниченной области пространства. В двух теоремах, доказываемых конструктивно, устанавливаются достаточные условия для области G на плоскости, при которых всякое s-семейство в $\mathbb{R}^2$ может быть нестрого просуммировано в G. При этом перестановка, задающая нестрогое суммирование векторов, находится эффективно. (Этот факт играет решающую роль при построении эффективных алгоритмов с оценками для задач теории расписаний, описываемых в главах 1, 4.)

Определение 13.1 Пусть $G \subset \mathbb{R}^m;\ X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m;\ \Sigma(X) = 0$. Будем говорить, что перестановка $\pi = (\pi_1, \ldots, \pi_n)$ задает *нестрогое суммирование векторов* из X в области G, если при любом $k \in \mathbb{N}_n$ из $x_\pi^{k-1} \notin G$ следует $x_\pi^k \in G$, где $x_\pi^k \doteq \sum_{j=1}^{k} x_{\pi_j}$.

Таким образом, "нестрогость" суммирования векторов в какой-либо области G заключается в том, что частичным суммам разрешается выходить из области G, но для любых двух последовательных частичных сумм хотя бы одна должна лежать в области G.

Определение 13.2 Пусть G — выпуклое множество в $\mathbb{R}^m$, l — прямая в $\mathbb{R}^m$. Тогда $h = l \cap G$ будем называть *хордой* множества G. Хорду, проходящую через начало координат (точку O), назовем *O-хордой* множества G.

Для любого вектора $x \in \mathbb{R}^m,\ x \neq 0$, O-хорду множества G, параллельную вектору x, будем обозначать $h_G(x)$. Пусть в $\mathbb{R}^m$ задана норма, тогда естественным образом определяется понятие "длины" хорды. Так как множество G выпукло, то ясно, что всякая его хорда $h = l \cap G$ является либо отрезком, либо лучом, либо совпадает со всей прямой l. В двух последних случаях хорда имеет бесконечную длину, а в первом случае, если $h = <a, b>$ (включая или исключая точки $\{a, b\}$), полагаем $\|h\| = \|a - b\|$.

Лемма 13.3 *Пусть в пространстве* $\mathbb{R}^m$ *заданы норма* $||\cdot||$, *выпуклое неограниченное множество* G, $0 \in G$, *и две параллельные прямые* $l = \{a + tb \,|\, t \in R\}$ *и* $l' = \{tb \,|\, t \in R\}$, *определяемые векторами* $a, b \in \mathbb{R}^m$, $b \neq 0$, *такие что* $l \cap 0^+G \neq \emptyset$. *Тогда для хорд* $h = l \cap G$ *и* $h' = l' \cap G$ *выполнено соотношение*

$$\|h\| \geq \|h'\|. \tag{13.1}$$

Доказательство. Из свойства $0 \in G$ следует, что $0^+G \subset G$, откуда $h \cap 0^+G = l \cap G \cap 0^+G = l \cap 0^+G \neq \emptyset$. Возьмем произвольное $c \in h \cap 0^+G$.

Предположим, что (13.1) не выполнено, т.е. $\|h\| < \|h'\|$. Тогда, во-первых, хорда h является интервалом. (Точнее, $h = <d, e> = <d, d + \mu b>$, $\mu \geq 0$, где принадлежность точек d и e интервалу h несущественна.) Во-вторых, $c \neq 0$. (В противном случае имели бы $0 \in h \subset l$, откуда $l = l'$ и $\|h\| = \|h'\|$.) И в-третьих, для некоторго $\varepsilon > 0$ имеет место соотношение $\|h'\| - \|h\| > 2\varepsilon\|b\|$.

Возьмем произвольное $\delta \in (0, \varepsilon)$ и определим точки $d_1 = d - \delta b$, $e_1 = e + \delta b$. Поскольку $d_1, e_1 \in l$ и $d_1, e_1 \notin h = l \cap G$, то $d_1, e_1 \notin G$. Отсюда $f \doteq d_1 - c \notin G$ и $g \doteq e_1 - c \notin G$. (В противном случае, если, например, $f \in G$, из $c \in 0^+G$ и определения рецессивного конуса имели бы $f + Q(c) \subset G$. Но $d_1 = f + c \notin G$.) Кроме того, из $c \in <d, e> \subset [d_1, e_1]$ имеем $0 \in [d_1, e_1] - c = [f, g]$ и $[f, g] \subset l'$. Отсюда $h' \doteq l' \cap G \subset [f, g]$. (Действительно, если бы существовала точка $x \in l' \cap G$, лежащая вне отрезка $[f, g]$, то из $0 \in [f, g]$ вытекало бы, что $f \in [0, x]$ либо $g \in [0, x]$. Но это противоречит соотношениям $f \notin G$, $g \notin G$, $[0, x] \subset G$.) Таким образом, получаем соотношения

$$\|h'\| \leq \|g - f\| = \|e_1 - d_1\| = (\mu + 2\delta)\|b\| < \|h\| + 2\varepsilon\|b\| < \|h'\|.$$

Полученное противоречие завершает доказательство леммы 13.3. □

Далее в этом разделе будет рассматриваться двумерное пространство $\mathbb{R}^2$. В случае $\mathbb{R}^2$ всякий вектор $a \in \mathbb{R}^2$, $a \neq 0$, определяет "левую" и "правую" замкнутые полуплоскости $L(a)$ и $R(a)$, т.е. полуплоскости, лежащие слева и справа от прямой $l(a)$ при движении по лучу $Q(a)$ в бесконечность. Обозначим также

$$L^0(a) = \mathbb{R}^2 \setminus R(a), \;\; R^0(a) = \mathbb{R}^2 \setminus L(a);$$

$$L'(a) = L^0(a) \cup Q(a), R'(a) = R^0(a) \cup Q(a).$$

При рассмотрении пространства $\mathbb{R}^2$ возможны два зеркально-симметричных варианта нумерации базисных векторов e_1, e_2. В первом варианте выполняется соотношение $e_1 \in L^0(e_2)$, во втором — соотношение $e_1 \in R^0(e_2)$. В дальнейших рассмотрениях пространства $\mathbb{R}^2$ мы будем считать, что принят первый вариант. Тогда полуплоскости $L(a)$ и $R(a)$ определяются соотношениями:

$$L(a) = \{x \in \mathbb{R}^2 \,|\, x_1a_2 - x_2a_1 \geq 0\}; \;\; R(a) = \{x \in \mathbb{R}^2 \,|\, x_1a_2 - x_2a_1 \leq 0\}. \tag{13.2}$$

Сформулируем несколько простых соотношений, которые далее будем использовать.

$$L(a) = R(-a); \;\; L^0(a) = R^0(-a);$$

$$X(-a) = -X(a);\ X^0(-a) = -X^0(a);\ X'(-a) = -X'(a),\ \text{для } X \in \{L, R\};$$
$$a \in L(b) \Longleftrightarrow b \in R(a);$$
$$a \in L^0(b) \Longleftrightarrow b \in R^0(a);$$
$$a \in L'(b) \Longleftrightarrow b \in R'(a),\ (a \neq 0,\ b \neq 0).$$

Из (13.2) также получаем

$$a \in L(b) \Longrightarrow -a \in L(-b); \tag{13.3}$$
$$a \in L(b) \Longrightarrow a \in L(a+b). \tag{13.4}$$

Утверждение 13.4 *Если выпуклый конус $A \subset \mathbb{R}^2$ отличен от $\mathbb{R}^2$, то он целиком содержится в некоторой полуплоскости $L(a)$.* □

(Следует из теоремы Хана-Банаха и импликации "из $A \subset L(a) + c$ следует $A \subset L(a)$".)

Утверждение 13.5 *Выпуклый замкнутый конус $A \subset \mathbb{R}^2$ представи́м в виде $A = A(a,b) \doteq (L^0(a) \cap R^0(b)) \cup Q(a) \cup Q(b)$, где $b \in L(a),\ a \neq 0,\ b \neq 0$, если и только если он отличен от прямой и не совпадает с $\mathbb{R}^2$.* □

На основании утверждения 13.5 рецессивный конус 0^+G (кроме двух оговоренных выше случаев) мы также будем представлять в виде $0^+G = A(a,b)$.

Отмечаем, что $A(a,b) \subseteq L(a) \cap R(b)$, причем множества в левой и правой частях не совпадают лишь в случае $a = \lambda b,\ \lambda > 0$.

Утверждение 13.6 *Если $a \neq 0,\ b \neq 0,\ \sigma \neq 0,\ b \in L(a),\ \sigma \in -A(a,b)$, то $L(\sigma) \subset R(a) \cup R(b)$.*

Доказательство. Предположим противное, т.е. для некоторого $x \in L(\sigma)$ выполнено $x \notin R(a) \cup R(b)$, или $x \in L^0(a) \cap L^0(b)$. Тогда $\sigma \in R(x);\ a, b \in R^0(x)$, т.е. три вектора σ, a, b лежат в одной полуплоскости $R(x)$. При этом, по условию, $b \in R(\sigma)$, $\sigma \in R(a)$, $a \in R(b)$, что возможно лишь в случае одинаковой направленности всех трех векторов (т.е. $\sigma \in Q(a) = Q(b) = A(a,b)$). Но это противоречит условию $\sigma \in -A(a,b)$. □

Утверждение 13.7 *Если a, b, σ — ненулевые векторы в $\mathbb{R}^2$,*

$$b \in L(a), \tag{13.5}$$

и $\sigma \in -A(a,b)$, то $L(\sigma) \setminus A(a,b) \subseteq L(\sigma) \setminus L'(a)$.

Доказательство. Пусть $x \in L(\sigma)$ и

$$x \notin A(a,b) = (L^0(a) \cap R^0(b)) \cup Q(a) \cup Q(b). \tag{13.6}$$

Докажем,что $x \notin L'(a)$. Допустим противное, т.е.

$$x \in L'(a). \tag{13.7}$$

Тогда из (13.6),

$$x \notin R^0(b) \cup Q(b) = R'(b). \tag{13.8}$$

По утверждению 13.6 имеем $x \in L(\sigma) \subset R(a) \cup R(b)$, или с учетом (13.8), $x \in R(a) \cup Q(-b)$. Так как из (13.5) следует $Q(-b) \subset R(a)$, с учетом (13.7) получаем $x \in R(a) \cap L'(a) = Q(a)$, что противоречит (13.6). □

Теорема 13.8 *Пусть в пространстве $\mathbb{R}^2$ с нормой s задано выпуклое замкнутое неограниченное множество G, содержащее точку 0 и такое, что все его O-хорды имеют s-норму, не меньшую единицы (см. рис. 13.1А). Тогда для всякого s-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ существует перестановка $\pi = (\pi_1, \ldots, \pi_n)$, задающая нестрогое суммирование векторов X в G. Если при этом известен рецессивный конус $A = A(a,b) \neq \emptyset$ множества G, то искомая перестановка π находится за $O(n \log n)$ операций.*

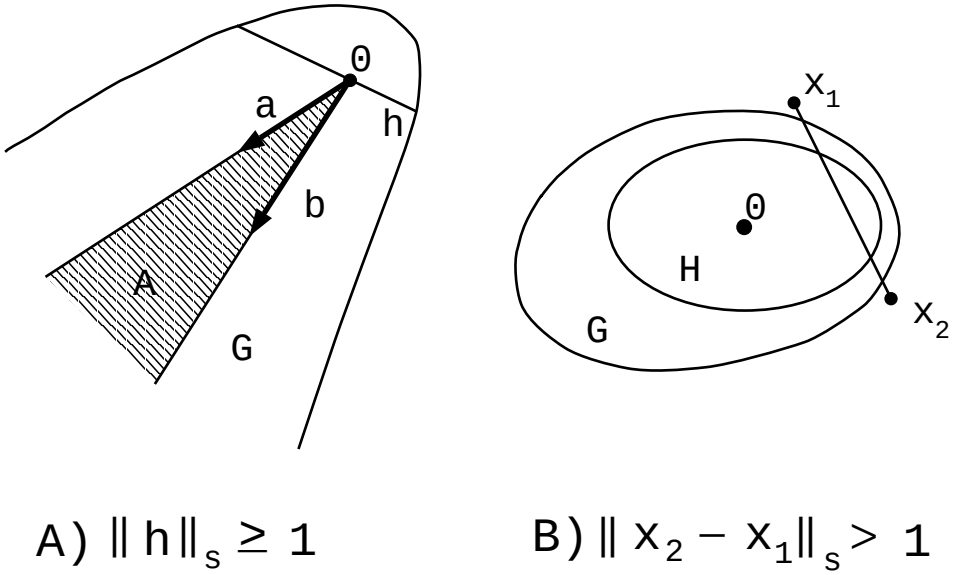


Рис. 13.1: Иллюстрации к теоремам 13.8 и 13.21

Доказательство. Опишем алгоритм $\mathcal{A}_{13.1}$ нахождения перестановки π. Он состоит из трех этапов, которые будем называть *предварительным, начальным* и *основным*.

Алгоритм $\mathcal{A}_{13.1}$

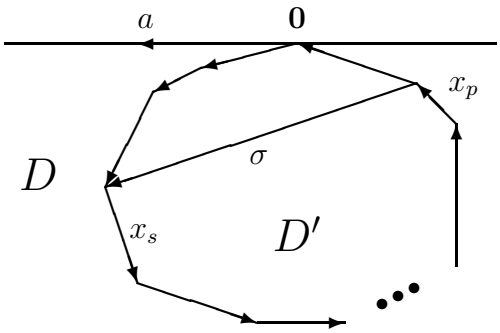


Рис. 13.2: Траектория суммирования векторов семейства X.

Предварительный этап. Занумеруем векторы X против часовой стрелки, начиная от вектора a. Пусть $\{x_1, \dots, x_n\}$ — полученная нумерация.

Начальный этап. Если $X \cap A \neq \emptyset$, то в качестве первых $j = |X \cap A|$ номеров перестановки π возьмем номера $1, 2, \dots, j$. Получим сумму $\sigma_j \doteq \sum_{i=1}^{j} x_{\pi_i} \in A$, причем ясно, что вся траектория суммирования будет лежать в A.

Основной этап. Последующие номера перестановки π будут определяться таким образом, чтобы на k-м шаге $(k = j, \dots, n)$ выполнялось следующее

Свойство (†)**.** *Если $\pi_1, \pi_2, \dots, \pi_k$ уже определены, то множество еще не использованных в π номеров из $\{1, 2, \dots, n\}$ образует связный отрезок $\{s_k, s_k + 1, \dots, p_k\}$, $p_k - s_k + 1 = n - k$.*

Пусть свойство (†) выполнено на шаге с номером $k < n$. Тогда для обеспечения этого свойства на последующих шагах алгоритма очередной, $(k+1)$-й вектор траектории суммирования будем выбирать из двух претендентов: x_{s_k} и x_{p_k}, которые для краткости будем обозначать x_s ("successor") и x_p ("predecessor"); этимология этих обозначений становится понятной из рис.13.2. Индекс k при написании суммы σ_k будем, как правило, опускать.

Принцип выбора очередного вектора прост. Пусть $\sigma \in G$. Тогда если $\sigma + x_s \in G$, то полагаем $\pi_{k+1} := s_k$. Если $\sigma + x_s \notin G$, $\sigma + x_p \in G$, то полагаем $\pi_{k+1} := p_k$. Наконец, если $\sigma + x_s \notin G, \sigma + x_p \notin G$ (из условия $\Sigma(X) = 0 \in G$ следует, что это не может произойти на шаге $k = n - 1$), то полагаем $\pi_{k+1} := s_k, \pi_{k+2} := p_k$ (или наоборот, что не играет роли). Если при этом $\sigma_{k+2} \notin G$, то STOP.

Алгоритм $\mathcal{A}_{13.1}$ описан.

Ясно, что алгоритм отвечает требуемой оценке трудоемкости, так как сложность исходной нумерации векторов есть $O(n \log n)$, а последующее построение перестановки π выполняется за $O(n)$ операций.

Для обоснования того, что полученная с помощью алгоритма $\mathcal{A}_{13.1}$ перестановка π обеспечивает нестрогое суммирование в G, достаточно доказать, что в алгоритме не возникает ситуации "STOP".

Это очевидно в случае $G = \mathbb{R}^2$. Если $G \neq \mathbb{R}^2$ (и следовательно, $A = 0^+G \neq \mathbb{R}^2$), то в силу утверждения 13.4 имеем $A \subseteq L(a)$. Если $A = L(a)$, то ситуации "STOP" не возникает, поскольку вычисляемая алгоритмом перестановка $\pi = (1, 2, \ldots, n)$ задает строгое суммирование векторов X в $A \subseteq G$. Далее предполагаем выполненным строгое включение $A \subset L(a)$. Обозначим через $X_k = \{x_{s_k}, x_{s_k+1}, \ldots, x_{p_k}\}$ множество векторов из X, не вошедших в сумму σ_k.

Наряду с алгоритмом $\mathcal{A}_{13.1}$ будем рассматривать класс алгоритмов $\mathcal{A}(\dagger)$, отличающихся от алгоритма $\mathcal{A}_{13.1}$ тем, что выбор вектора $x_{\pi k+1}$ из двух претендентов $\{x_{s_k}, x_{p_k}\}$ на $(k+1)$-м шаге осонвного этапа $(k = j, \ldots, n-1)$ происходит произвольным образом. Ясно, что любой алгоритм $\mathcal{A} \in \mathcal{A}(\dagger)$ удовлетворяет свойству $(\dagger)$, и что $\mathcal{A}_{13.1} \in \mathcal{A}(\dagger)$. Доказываемые в нижеследующих леммах 13.9–13.13 свойства справедливы для любого алгоритма $\mathcal{A} \in \mathcal{A}(\dagger)$.

Лемма 13.9 *На любом шаге k $(k < n)$ алгоритма справедливы соотношения: $x_s \in L(\sigma), x_p \in R(\sigma)$.*

Доказательство. Просуммировав векторы из X в порядке нумерации, установленной на предварительном этапе, получим траекторию суммирования, ограничивающую выпуклый многоугольник D (см. рис.13.2). В силу свойства $(\dagger)$ на любом шаге алгоритма вектор $\sigma = \sigma_k$ соответствует хорде этого многоугольника и вместе с векторами из X_k образует выпуклый многоугольник D', целиком лежащий слева от σ при $\sigma \neq 0$ (впрочем, это справедливо по отношению к любой из сторон многоугольника в силу упорядочения векторов против часовой стрелки), т.е. $D' \subset L(\sigma)$. Отсюда $x_s \in L(\sigma), x_p \in R(\sigma)$. □

Лемма 13.10 *На каждом шаге k $(k < n)$ алгоритма выполнены импликации: $\sigma \in R(a) \Rightarrow x_s \in L(a)$; $\sigma \in L(a) \Rightarrow x_p \in R(a)$.*

Доказательство. Пусть $\sigma \in R(a)$. Если $x_s \in \mathbb{R}^2 \setminus L(a) = R^0(a)$, то по правилу нумерации векторов и в силу свойства $(\dagger)$ имеем $X_k \subset R^0(a)$. Но тогда $\sigma + \Sigma(X_k) \neq 0$, что противоречит свойству $\Sigma(X) = 0$. Вторая импликация доказывается аналогично. □

Лемма 13.11 *Если $A \subset L(a)$, то на любом шаге алгоритма имеем*

$$\sigma \notin -A \setminus \{0\}. \tag{13.9}$$

Доказательство. Из $A \neq L(a)$ следует, что $A \cap (-A \setminus \{0\}) = \emptyset$, поэтому (13.9) выполняется на любом шаге начального этапа, когда $\sigma \in A$. Предположим, что (13.9) нарушено на шаге k основного этапа. Тогда $\sigma \in -A \subset R^0(a) \cup Q(-a)$, и

$$x_s \notin A. \tag{13.10}$$

Так как по лемме 13.9 имеем $x_s \in L(\sigma)$, с учетом (13.10) и утверждения 13.7 выполняется $x_s \notin L'(a)$, т.е. $x_s \in R^0(a) \cup Q(-a)$. Но тогда, согласно правилу нумерации векторов и в силу свойства $(\dagger)$, получаем $X_k \subset R^0(a) \cup Q(-a)$. Следовательно, $\sigma + \Sigma(X_k) \neq \emptyset$, что противоречит $\Sigma(X) = 0$. □

Обозначим: $\tilde{G}_1 = R^0(b)\setminus G,\ \tilde{G}_2 = L^0(a)\setminus G$. Замечаем, что $\tilde{G}_1 \subseteq R^0(b)\setminus A = R^0(b)\setminus\{(L^0(a)\cap R^0(b))\cup Q(a)\cup Q(b)\} = R^0(b)\setminus L'(a) =$ (так как $-a \in L(b)$) $= R^0(b)\setminus\{L'(a)\cup Q(-a)\} = R^0(b)\setminus L(a)$, откуда

$$\tilde{G}_1 \subseteq R^0(a)\cap R^0(b). \tag{13.11}$$

Аналогично доказывается

$$\tilde{G}_2 \subseteq L^0(a)\cap L^0(b).$$

Лемма 13.12 *Если $\sigma \in G$, то*

$$\sigma + x_s \notin \tilde{G}_1, \tag{13.12}$$

$$\sigma + x_p \notin \tilde{G}_2, \tag{13.13}$$

Доказательство. Пусть (13.12) нарушено. Положим $\sigma_{k+1} = \sigma + x_s$. Тогда с учетом (13.11)

$$\sigma_{k+1} \in \tilde{G}_1 \subseteq R^0(a). \tag{13.14}$$

Ясно, что $(k+1)$-й шаг выполняется не на начальном этапе (когда $\sigma_k \in A \subset L(a)$ при любом k), а на основном. Следовательно, $x_s \notin A$. Из (13.14) и лемм 13.9, 13.10 получаем $x_{s_{k+1}} \in L(\sigma_{k+1})\cap L(a) = A(a,-\sigma_{k+1})$. Отсюда в соответствии с правилом нумерации векторов вытекает, что

$$x_s \in A(a, x_{s_{k+1}}) \subseteq A(a,-\sigma_{k+1}) \subset L(a). \tag{13.15}$$

Из (13.14) и (13.15) следует, что

$$\sigma = \sigma_{k+1} - x_s \in R^0(a). \tag{13.16}$$

Из (13.15) при некоторых $\lambda \geq 0,\ \mu > 0$ имеем представление $x_s = \lambda a - \mu\sigma_{k+1} = \lambda a - \mu(\sigma + x_s)$, или $x_s = \frac{\lambda}{1+\mu}a - \frac{\mu}{1+\mu}\sigma$. Отсюда с учетом $\sigma \in G$, $a \in 0^+G$ получаем

$$\sigma_{k+1} = \sigma + x_s \in \left[\sigma, \sigma + \frac{1+\mu}{\mu}x_s\right] = \left[\sigma, \frac{\lambda}{\mu}a\right] \subset G,$$

что противоречит предположению $\sigma_{k+1} \in \tilde{G}_1 = R^0(b)\setminus G$.

Отношение (13.13) доказывается аналогично. □

Лемма 13.13 *Пусть $\sigma \in G,\ \sigma + x_s \notin G,\ \sigma + x_p \notin G$. Тогда $\sigma + x_s + x_p \in G$.*

Доказательство. Из лемм 13.11,13.12 имеем $\sigma + x_s \notin G \cup \tilde{G}_1 \cup (-A)$. Поэтому $\sigma + x_s \in \tilde{G}_2$. Аналогично, $\sigma + x_p \in \tilde{G}_1$. Докажем, что

$$\sigma + x_s + x_p \notin \tilde{G}_1. \tag{13.17}$$

Допустим противное, т.е.

$$\sigma + x_s + x_p \in \tilde{G}_1. \tag{13.18}$$

Обозначим $\sigma_{k+1} = \sigma + x_s,\ \sigma_{k+2} = \sigma_{k+1} + x_p$, и докажем, что

$$[\sigma_{k+1}, \sigma_{k+2}] \cap A \neq \emptyset. \tag{13.19}$$

Так как $\sigma_{k+1} \in \tilde{G}_2 \subset L^0(a)$, согласно лемме 13.10 имеем $x_p \in R(a)$, а из леммы 13.9 получаем $x_p \in R(\sigma_{k+1})$. Таким образом, справедливо представление $x_p = \lambda a - \mu\sigma_{k+1}$, где $\lambda \geq 0,\ \mu > 0$, причем из (13.18) следует, что $\mu > 1$. Отсюда $\sigma_{k+1} + (1/\mu)x_p = (\lambda/\mu)a \in A \cap [\sigma_{k+1}, \sigma_{k+2}]$, что доказывает (13.19).

Из (13.19) и условий $\sigma_{k+1} \notin G, \sigma_{k+2} \notin G$ следует, что хорда $h \doteq \{\nu\sigma_{k+1} + (1-\nu)\sigma_{k+2} \,|\, \nu \in R\} \cap G$ строго содержится в отрезке $[\sigma_{k+1}, \sigma_{k+2}]$. А так как эта хорда замкнута, то $\|h\| < \|x_p\| \leq 1$. Но по лемме 13.3, хорда h не короче параллельной ей O-хорды, и значит, ее длина не меньше 1. Противоречие доказывает (13.17). Аналогично доказывается

$$\sigma + x_s + x_p \notin \tilde{G}_2. \tag{13.20}$$

Из (13.17),(13.20) и леммы 13.11 следует

$$\sigma + x_s + x_p \in \mathbb{R}^2 \setminus (\tilde{G}_1 \cup \tilde{G}_2 \cup (-A)) \subseteq G$$

Лемма 13.13 доказана.

Из последней леммы следует, что в алгоритме $\mathcal{A}_{13.1}$ не возникает ситуации “STOP”. Таким образом, алгоритм $\mathcal{A}_{13.1}$ обеспечивает нестрогое суммирование векторов X в области G.

Теорема 13.8 доказана.

Замечание 13.14 *В алгоритме мы нигде не пользовались тем, что конус A соответствует совокупности* **всех** *направлений, по которым множество G удаляется в бесконечность. Для определения и обоснования алгоритма, обеспечивающего нестрогое суммирование в G, достаточно иметь хотя бы одно такое направление $a \in \mathbb{R}^m$.*

Замечание 13.15 *Для существования перестановки π, задающей нестрогое суммирование конкретного s-семейства векторов $X = \{x_1, \ldots, x_n\}$ в области G, не обязательно, чтобы длина* **каждой** *O-хорды множества G была не меньше единицы (в норме s). Достаточно, во-первых, потребовать выполнения этого условия лишь для O-хорд $h_G(x_j)$, параллельных векторам $x_j \in X$, и во-вторых, сравнивать $\|h_G(x_j)\|_s$ не с единицей, а с $\|x_j\|_s$. Таким образом, для существования нестрогого суммирования семейства X в G достаточно выполнения условия*

$$\|h_G(x_j)\|/\|x_j\| \geq 1,\ \forall x_j \in X,\ x_j \neq 0,$$

которое не зависит от нормы.

Замечание 13.16 *Из лемм 13.11 и 13.12 следует, что в тех случаях, когда при работе алгоритма $\mathcal{A}_{13.1}$ частичная сумма вынуждена покинуть на один шаг множество G (ситуация леммы 13.13), мы можем по своему выбору оставить ее в $\tilde{G}_2$, выбрав $x_{\pi_{k+1}} = x_s$, или — в $\tilde{G}_1$, выбрав $x_{\pi_{k+1}} = x_p$. Таким образом, всегда выбирая в такой ситуации вектор x_s, мы получаем такое нестрогое суммирование в G, которое является строгим по отношению к области $G \cup \tilde{G}_2 \subset G \cup L(a)$. Аналогично, выбирая вектор x_p, получаем строгое суммирование в области $G \cup \tilde{G}_1 \subset G \cup R(b)$.*

Замечание 13.17 *Ограничение снизу (равное 1) на s-норму любой O-хорды множества G в теореме 13.8 не может быть ослаблено.*

Для обоснования этого замечания мы не ограничимся построением соответствующего контрпримера, а докажем более сильный результат. Он состоит в том, что при сколь угодно малом ослаблении ограничения на длину O-хорд множества G проверка существования нестрогого суммирования в G заданного s-семейства векторов становится NP-трудной проблемой в сильном смысле. Это обстоятельство проявляется уже в случае, когда все векторы из X коллинеарны вектору $x \in \mathbb{R}^m$ и $\|h_G(x)\| < 1$. При этом свойство множества G быть неограниченным в каких-то направлениях, отличных от x, оказывается несущественным, задача становится, по-существу, одномерной, и вместо векторов можно говорить о числах.

При заданном значении параметра $\lambda \in (0,1)$ сформулируем задачу распознавания, которую назовем задачей *нестрогого суммирования чисел*, или НСЧ1(λ).

НСЧ1($\boldsymbol{\lambda}$).
УСЛОВИЕ: заданы число $D \in \mathcal{Z}^+$; семейство чисел $X = \{x_1, \ldots, x_n\} \subset \mathcal{Z}$ такое, что $|x_j| \le D,\ \Sigma(X) = 0$; число $d \in \mathcal{Z}^+$ такое, что $d \in [\lambda D, D)$.
ВОПРОС: существует ли перестановка $\pi = (\pi_1, \ldots, \pi_n)$ такая, что

$$x_\pi^{k-1} \in [0,d] \vee x_\pi^k \in [0,d],\ \forall k \in \mathbb{N}_n,$$

где $x_\pi^k = \sum_{j=1}^k x_{\pi_j}$?

Теорема 13.18 *При любом фиксированном рациональном $\lambda \in (0,1)$ задача* НСЧ1*(λ) NP-полна в сильном смысле.*

Доказательство. При заданном λ к задаче НСЧ1(λ) будет сведена при соответствующем k задача k-РАЗБИЕНИЕ, сформулированная в разделе 11.

Пусть задано рациональное число $\lambda \in (0,1)$. Возьмем $k = \lceil 1/(1-\lambda) \rceil + 1$ и предположим, что задан вход задачи k-РАЗБИЕНИЕ, т.е. такое число $B \in \mathcal{Z}^+$ и такие размеры $s(j)$ элементов множества $A = \{1, 2, \ldots, km\}$, что $\sum_{j \in A} s(j) = Bm$ и $B/(k+1) < s(j) < B/(k-1)$. Зададим вход задачи НСЧ1 (λ):

$$n := (k+2)m;\ l := \left\lceil \frac{\lambda}{(1-\lambda)^2 B} \right\rceil;\ B' := lB;\ d := B';\ D := \lfloor B'/\lambda \rfloor;$$

$$x_j = \begin{cases} s(j)l & \text{при } j \in A; \quad (A - \text{числа}) \\ D - d \doteq \beta & \text{при } j \in \{km+1, \ldots, km+m\}; \quad (\beta - \text{числа}) \\ -D \doteq \gamma & \text{при } j \in \{(k+1)m+1, \ldots, n\}. \quad (\gamma - \text{числа}) \end{cases}$$

Нетрудно убедиться, что определенный таким образом вход удовлетворяет условию задачи НСЧ1 (λ). Кроме того, выполняются неравенства

$$x_j < \beta, \ \forall j \in A. \tag{13.21}$$

Действительно, так как $x_j < B'/(k-1)$ при $j \in A$, достаточно установить, что $\beta = \lfloor B'/\lambda \rfloor - B' \geq B'/(k-1)$. Для этого (с учетом определения k) достаточно доказать неравенство

$$\lambda_{-1}B' - 1 - B' \geq B'/(\lceil 1/(1-\lambda) \rceil)^{-1},$$

или

$$B' \geq (\lambda^{-1} - 1 - (\lceil 1/(1-\lambda) \rceil)^{-1})^{-1}.$$

Последнее вытекает из неравенства

$$B' \geq \lambda(1-\lambda)^{-2} = (\lambda^{-1} - 1 - (1/(1-\lambda))^{-1})^{-1}.$$

Покажем далее, что искомая перестановка π в задаче НСЧ1 (λ) существует, если и только если в задаче k-РАЗБИЕНИЕ существует такое разбиение $\{A_1, \ldots, A_m\}$ множества A, что

$$\sum_{i \in A_j} s(i) = B, \ j \in \mathbb{N}_m.$$

Если разбиение $\{A_1, \ldots, A_m\}$, удовлетворяющее указанному свойству, существует, то перестановку π определяет последовательность номеров

$$\tilde{A}_1\beta_1\gamma_1\tilde{A}_2\beta_2\gamma_2 \cdots \tilde{A}_m\beta_m\gamma_m,$$

где $\tilde{A}_i$ — любая перестановка номеров из множества A_i, $(\beta_1, \ldots, \beta_m)$ — любая перестановка номеров β-чисел, $(\gamma_1, \ldots, \gamma_m)$ — любая перестановка номеров γ-чисел. Очевидно, что перестановка π — искомая в задаче НСЧ1(λ).

Пусть теперь перестановка π задает нестрогое суммирование чисел $\{x_1, \ldots, x_n\}$ в отрезке $[0, d]$. Последовательно соединив звеньями ломаной точки, соответствующие суммам $x_\pi^0, x_\pi^1, \ldots, x_\pi^n$, получим замкнутую траекторию суммирования. Звенья, соответствующие γ-, β- и A-числам, будем называть γ-, β- и A-звеньями соответственно.

Для всякого γ-числа x_{π_j} ровно одна из двух сумм $\{x_\pi^{j-1}, x_\pi^j\}$ (ровно одна из концевых вершин γ-звена) лежит в отрезке $[0, d]$, причем точно на его границе, т.е. совпадает либо с 0, либо с d. В противном случае расстояние от другого конца γ-звена ($x_\pi^{j^*}$) до отрезка $[0, d]$ превосходит $D - d = \beta$ и нестрогое суммирование невозможно. Таким образом, расстояние от $x_\pi^{j^*}$ до $[0, d]$ в точности равно β. Поэтому в случае $j^* = j - 1$ (т.е. при $x_\pi^{j^*} = D$) число $x_{\pi_{j-1}}$, ввиду (13.21), является β-числом, а в случае $j^* = j$ (когда, как легко понять, $x_\pi^{j^*} = d - D$) таковым является число $x_{\pi_{j+1}}$. В обоих случаях вершина траектории суммирования, совпадающая с суммой $x_\pi^{j^*}$, инцидентна паре звеньев, одно из

которых является γ-, а другое — β-звеном. Таким образом, каждое γ-звено инцидентно хотя бы одному β-звену.

Покажем теперь, что каждое β-звено не может быть инцидентно двум γ-звеньям. Допустим противное, т.е. $x_{\pi_j} = \beta,\ x_{\pi_{j-1}} = \gamma,\ x_{\pi_{j+1}} = \gamma$. Тогда если $x_\pi^{j-1} \notin [0,d]$, то $x_\pi^j = 0,\ x_\pi^{j+1} = -D$, и нестрогое суммирование в $[0,d]$ невозможно. Если же $x_\pi^{j-2} \notin [0,d]$, то $x_\pi^{j-2} = D$ (следовательно, $x_{\pi_{j-2}} = \beta$), $x_\pi^{j-1} = 0,\ x_\pi^j = \beta,\ x_\pi^{j+1} = \beta - D < 0$. Поэтому $x_\pi^{j+1} = d - D,\ x_{\pi_{j+2}} = \beta,\ x_\pi^{j+2} = 0$. (Отсюда получаем соотношение $x_\pi^{j+1} = \beta - D = d - D$, т.е. $\beta = d$, и $D = 2d$.) Отрезок траектории из пяти звеньев (от $x_{\pi_{j-2}}$ до $x_{\pi_{j+2}}$) начинается β-звеном в вершине $x_\pi^{j-3} = d$ и кончается β-звеном в вершине 0. Поэтому крайним β-звеньям уже не могут быть инцидентны другие γ-звенья (предшествующие звену $x_{\pi_{j-2}}$ или следующие за $x_{\pi_{j+2}}$). Удалим из траектории суммирования все такие пятерки звеньев, содержащие три β- и два γ-звена. В оставшейся (несвязной) части траектории количество β-звеньев будет строго меньше количества γ-звеньев; при этом каждое γ-звено инцидентно одному или двум β-звеньям, а каждое β-звено — не более чем одному γ-звену. Но это невозможно, так как в двудольном $\gamma - \beta$-графе инцидентности суммы степеней в обеих долях должны совпадать.

Это же свойство траектории позволяет сделать вывод, что каждое γ-звено инцидентно в точности одному β-звену, а каждое β-звено — точно одному γ-звену. Таким образом, в траектории суммирования все γ- и β-звенья объединены в m непересекающихся пар, промежутки между которыми заполнены A-звеньями, что дает разбиение множества A на m подмножеств $\{A_1, \ldots, A_m\}$. Так как каждая $\gamma - \beta$-пара начинается в вершине d и кончается в вершине 0, то сумма A-чисел из каждого множества A_i в точности равна d, т.е. разбиение $\{A_1, \ldots, A_m\}$ является искомым для исходной задачи k-РАЗБИЕНИЕ.

Для завершения доказательства теоремы остается заметить, что при любом фиксированном λ величина k является константой, а длина записи входной информации задачи НСЧ1 (λ) и задачи k-РАЗБИЕНИЕ по порядку равны, что обосновывает полиномиальность процедуры сведения.

Теорема 13.18 доказана.

Из теоремы 13.18 следует, что если в теореме 13.8 ограничение $\|h_G(x)\| \geq 1$ заменить на $\|h_G(x)\| \geq 1 - \varepsilon$ при произвольном $\varepsilon > 0$, то происходит качественное изменение сложности задачи о нестрогом суммировании в G заданного семейства векторов. Именно, справедливо

Следствие 13.19 *Пусть задано число $\lambda \in (0,1)$, и пусть в пространстве $\mathbb{R}^2$ с нормой s задано выпуклое замкнутое неограниченное множество G, содержащее начало координат. Если длины (в норме s) всех O-хорд множества G не меньше λ, то проблема распознавания существования нестрогого суммирования в G произвольного s-семейства векторов является NP-трудной в сильном смысле.*

Определение 13.20 Будем говорить, что три вектора $y_1, y_2, y_3 \in \mathbb{R}^2$ образуют *нормальную тройку*, если существуют числа $\lambda_i \geq 0,\ i = 1,2,3$, такие, что $\sum \lambda_i y_i = 0,\ \sum \lambda_i > 0$.

Легко видеть, что три вектора на плоскости образуют нормальную тройку, если и только если они не лежат в одной открытой полуплоскости.

Для каждого вектора $x \in \mathbb{R}^2$ определим его угловую координату $u(x) \in [0, 2\pi)$. Для определенности будем считать, что угол возрастает при повороте вектора против часовой стрелки (как это принято изображать на комплексной плоскости). Считаем также, что $u(0) = 0$.

Для любых двух векторов a и b таких, что $u(a) \neq u(b),\ a \neq 0,\ b \neq 0$, через $C(a, b)$ обозначим сектор плоскости от луча $Q(a)$ против часовой стрелки до луча $Q(b)$. Более формально:

$$C(a,b) = \begin{cases} L(a) \cap R(b), & \text{если } b \in L(a), \\ L(a) \cup R(b), & \text{если } b \in R(a). \end{cases}$$

В случае $u(a) = u(b)$ полагаем $C(a, b) = C(b, a) = Q(a) = Q(b)$.

Через $C^0(a, b)$ обозначим открытый сектор $C(a, b) \setminus (Q(a) \cup Q(b))$, а через $\angle C(a, b)$ — величину угла сектора $C(a, b)$, т.е.

$$\angle C(a,b) = \begin{cases} u(b) - u(a) & , \text{ если } u(b) \geq u(a), \\ u(b) - u(a) + 2\pi & , \text{ если } u(b) < u(a). \end{cases}$$

Теорема 13.21 *Пусть на плоскости заданы норма s с единичным шаром H и выпуклое множество G такое, что*

$$\text{если } x_1 \notin G,\ x_2 \notin G \text{ и } [x_1, x_2] \cap H \neq \emptyset, \text{ то } \|x_2 - x_1\| > 1 \tag{13.22}$$

(см. рис. 13.1В на стр. 91). Тогда для любого s-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ существует перестановка $\pi = (\pi_1, \ldots, \pi_n)$, задающая нестрогое суммирование векторов X в G. Искомая перестановка может быть найдена с трудоемкостью $O(n \log n)$.

Замечание 13.22 *Из свойства (13.22) и выпуклости G следует, что $H \subset G$ и всякая хорда множества G, пересекающая H, имеет длину не меньше 1. Наоборот, если $H \subset G$ и всякая хорда множества G, пересекающая H, имеет длину не меньше 1, то с добавлением условия замкнутости G получаем свойство (13.22).*

Мы, однако, не будем использовать замкнутость G при доказательстве теоремы 13.21, а будем исходить из более общего свойства (13.22).

Доказательство теоремы 13.21. Опишем алгоритм $\mathcal{A}^*$ нахождения искомой перестановки π.

Алгоритм $\mathcal{A}^*$

Этап 1 (Нумерация векторов и построение дерева поиска).

Нумеруем векторы $x_i \in X$ по возрастанию их угловой координаты $u(x_i)$, при равенстве угловых координат — по возрастанию длины, наконец, при равенстве обеих координат — по возрастанию исходных номеров векторов. Таким образом, все нулевые

векторы получат наименьшие номера (до k_0 включительно, где $k_0 \geq 0$). Трудоемкость алгоритма нумерации — $O(n \log n)$ операций.

Далее согласно Тарьяну [132] строим сбалансированное по глубине двоичное дерево поиска (T_0) на множестве вершин $I_0 = \{1, \ldots, n\}$ по ключу $u(x_i),\ i \in I_0$. Напомним, что такое дерево поиска обладает следующим полезным свойством: для каждой внутренней (не концевой) вершины i дерева T_0 все вершины $\{j\}$ в ее левом поддереве имеют ключи $u(x_j) \leq u(x_i)$, а все вершины в правом поддереве — ключи $u(x_j) \geq u(x_i)$. "Сбалансированность по глубине" означает, что расстояние от корня дерева до любого из его листьев не превосходит $O(\log n)$.

При построении такого дерева мы для каждой вершины i определим ее ранг $r(i)$ как максимальное целое ν такое, что i делится на 2^ν. Таким образом, ранги принимают значения $0, 1, \ldots, \lfloor \log_2 n \rfloor$, причем максимальный ранг $r = \lfloor \log_2 n \rfloor$ достигается на единственной вершине $i_0 = 2^r$, которую принимаем за корень дерева T_0. Для каждой вершины $i \in \mathbb{N}_n$, отличной от корня i_0, в качестве ее родителя $p(i)$ в дереве T_0 выбираем одну из двух вершин $i + 2^{r(i)},\ i - 2^{r(i)}$. Выбор осуществляется следующим образом:

если $i + 2^{r(i)} > n$, то $p(i) := i - 2^{r(i)}$,
иначе если $r(i + 2^{r(i)}) = r(i) + 1$, то $p(i) := i + 2^{r(i)}$,
иначе $p(i) := i - 2^{r(i)}$.

Так как в любом из случаев имеем $r(p(i)) \geq r(i) + 1$, то глубина дерева не превосходит максимального ранга, т.е. $\lfloor \log_2 n \rfloor$. (Предоставляем читателю убедиться, что так определенное дерево является двоичным, т.е. у каждой вершины может быть не более двух детей.)

Этап 2 (Построение перестановки π).

Перестановку π строим по шагам $k = 1, \ldots, n$. После шага k известны первые k значений $\pi_1, \ldots, \pi_k$ перестановки π и множество I_k номеров векторов, еще не включенных в частичную сумму $\sigma_k \doteq \sum_{i=1}^{k} x_{\pi_i}$. Известно также дерево поиска T_k со множеством вершин I_k, причем глубина дерева не превосходит величины r.

Для $k = 1, \ldots, k_0$ полагаем $\pi_k = k$, т.е. первыми в искомом порядке суммирования идут нулевые векторы. Исключив номера $\{1, \ldots, k_0\}$ из множества I_0, получим множество I_{k_0}, и удалив соответствующие вершины из дерева T_0, получим дерево T_{k_0}. (Удаление любой вершины из n-вершинного дерева поиска требует $O(\log n)$ операций [132, стр.46].)

Пусть выполнено k шагов этапа 2; $k_0 \leq k \leq n - 2$.

Шаг $k + 1$. Находим номера $p_k, s_k \in I_k$ векторов $x_i \in X_k$, имеющих ближайшие к $u = u(-\sigma_k)$ значения угловой координаты (снизу и сверху соответственно). Пусть $I_k^-(u) \doteq \{i \in I_k \,|\, u(x_i) \leq u\}, I_k^+(u) \doteq \{i \in I_k \,|\, u(x_i) \geq u\}$. Положим

$$p_k = \begin{cases} \max\{j \in I_k^-(u)\} & \text{, если } I_k^-(u) \neq \emptyset, \\ \max\{j \in I_k\} & \text{в противном случае;} \end{cases}$$

$$s_k = \begin{cases} \min\{j \in I_k^+(u) \setminus \{p_k\}\} & \text{, если } I_k^+(u) \setminus \{p_k\} \neq \emptyset, \\ \min\{j \in I_k\} & \text{в противном случае.} \end{cases}$$

(Вместо p_k, s_k и σ_k там, где это не вызывает недоразумений, будем для краткости писать p, s и σ.) Нетрудно предложить алгоритм поиска вершин p_k, s_k в дереве поиска T_k, требующий $O(\log n)$ действий.

В качестве значения π_{k+1} будем брать один из двух номеров $\{p, s\}$. (Подробное правило выбора будет описано ниже.) Исключив (за $O(\log n)$ действий) вершину π_{k+1} из дерева T_k, приходим к дереву T_{k+1}.

Алгоритм описан.

Утверждение теоремы вытекает из трех лемм, доказываемых ниже.

Лемма 13.23 *На каждом шаге $k = 1, \ldots, n$ алгоритма $\mathcal{A}^*$ тройка векторов $\langle \sigma, x_p, x_s \rangle$ является нормальной.*

Доказательство. По определению векторов x_p, x_s на каждом шаге k имеем

$$-\sigma_k \in C(x_p, x_s), \tag{13.23}$$

$$X_k \subset C(x_s, x_p). \tag{13.24}$$

Отсюда $\angle C(x_s, x_p) \geq \pi$, или

$$\angle C(x_p, x_s) \leq \pi. \tag{13.25}$$

В противном случае при $\sigma_k \in C(x_s, x_p)$ с учетом (13.24) это противоречило бы равенству $\Sigma(X) = 0$, а при $\sigma_k \in C(x_p, x_s)$ с учетом (13.23) получаем, что σ_k и все векторы из X_k лежат в одной полуплоскости, причем в предположении $\angle C(x_s, x_p) < \pi$ векторы x_p и x_s неколлинеарны, что также противоречит равенству $\Sigma(X) = 0$.

Из (13.23) и (13.25) вытекает нормальность тройки $\langle \sigma, x_p, x_s \rangle$.

Лемма 13.23 доказана.

Лемма 13.24 *Пусть $\lambda_\sigma, \lambda_p, \lambda_s$ — коэффициенты, удостоверяющие нормальность тройки $\langle \sigma, x_p, x_s \rangle$, т.е.*

$$\lambda_\sigma \sigma + \lambda_p x_p + \lambda_s x_s = 0;$$

$$\lambda_i \geq 0 \text{ при } i \in \{\sigma, p, s\}; \ \sum \lambda_i > 0.$$

Пусть также

$$\sigma + x_p \notin H \text{ и } \sigma + x_s \notin H. \tag{13.26}$$

Тогда

$$\lambda_\sigma < \max\{\lambda_p, \lambda_s\}. \tag{13.27}$$

Доказательство. Пусть $\lambda_\sigma \geq \max\{\lambda_p, \lambda_s\}$. Тогда $\lambda_\sigma > 0$.

В случае $\lambda_p \geq \lambda_s$ имеем

$$\sigma + x_p = \left(1 - \frac{\lambda_p}{\lambda_\sigma}\right) x_p + \frac{\lambda_s}{\lambda_\sigma}(-x_s) = \gamma_p x_p + \gamma_s(-x_s),$$

где $\gamma_p \geq 0$, $\gamma_s \geq 0$, $\gamma_p + \gamma_s \leq 1$, $x_p \in H$, $-x_s \in H$ (в силу симметричности H). Следовательно, $\sigma + x_p \in \operatorname{conv}\{x_p, -x_s, 0\} \subset H$, что противоречит (13.26).

В случае $\lambda_p \le \lambda_s$ аналогично имеем

$$\sigma + x_s = \frac{\lambda_p}{\lambda_\sigma}(-x_p) + \left(1 - \frac{\lambda_s}{\lambda_\sigma}\right) x_s \in H,$$

что противоречит (13.26).

Лемма 13.24 доказана.

Теперь мы имеем возможность более подробно описать работу алгоритма $\mathcal{A}^*$ на шаге $k+1$.

Шаг $k+1$.

- Если $\sigma + x_p \in H$, то $\pi_{k+1} := p$.
- Если $\sigma + x_p \notin H$ и $\sigma + x_s \in H$, то $\pi_{k+1} := s$.
- Если $\sigma + x_p \notin H$ и $\sigma + x_s \notin H$, то находим неотрицательные коэффициенты $\lambda_\sigma, \lambda_p, \lambda_s$, удостоверяющие нормальность тройки $\langle \sigma, x_p, x_s \rangle$. По лемме 13.24 выполнено (13.27). Поэтому возможны лишь следующие три варианта соотношений между коэффициентами $\lambda_\sigma, \lambda_p, \lambda_s$.

  Вариант 1 $(\lambda_s \le \lambda_\sigma < \lambda_p)$: $\pi_{k+1} := p$.

  Вариант 2 $(\lambda_p \le \lambda_\sigma < \lambda_s)$: $\pi_{k+1} := s$.

  Вариант 3 $(\lambda_\sigma < \min\{\lambda_p, \lambda_s\})$:

  A) если $\sigma + x_p \in G$, то $\pi_{k+1} := p$,

  B) если $\sigma + x_p \notin G, \sigma + x_s \in G$, то $\pi_{k+1} := s$,

  C) если $\sigma + x_p \notin G, \sigma + x_s \notin G$, то $\pi_{k+1} := p,\ \pi_{k+2} := s$.

Шаг $k+1$ алгоритма $\mathcal{A}^*$ описан.

Как видно из описания шага $k+1$, в случае C варианта 3 мы выполняем сразу два шага алгоритма.

Лемма 13.25 *Пусть на шаге $t = k$ $(k_0 \le k \le n-2)$ выполнены соотношения*

$$\sigma_t \in G, \tag{13.28}$$

$$(\sigma_t + Q(-x')) \cap H \ne \emptyset, \tag{13.29}$$

где x' — один из двух векторов x_{p_t}, x_{s_t}. Тогда (13.28),(13.29) выполняются либо на шаге $t = k+1$, либо на шаге $t = k+2$.

Доказательство. Заметим, что с учетом $\Sigma(X) = 0$ равенство в (13.25) достигается лишь в том случае, если σ_k и все векторы из X_k коллинеарны. Поскольку доказательство утверждения леммы 13.25 в этом случае тривиально (в качестве вектора x_{π_t} на

каждом шаге $t = k+1, \ldots, n$ выбирается вектор x_{p_t}, противоположно направленный с σ_t), то далее будем предполагать, что

$$\angle C(x_p, x_s) < \pi. \tag{13.30}$$

Если $\sigma + x_p \in H$ или $\sigma + x_s \in H$, то полагаем соответственно $\pi_{k+1} := p$ или $\pi_{k+1} := s$, и соотношения (13.28), (13.29) выполнены при $t = k+1$. Далее считаем, что имеет место соотношение (13.26). Для определенности будем также считать, что (13.29) при $t = k$ выполняется для $x' = x_s$, т.е.

$$(\sigma_t + Q(-x_s)) \cap H \neq \emptyset. \tag{13.31}$$

(В случае $x' = x_p$ доказательство полностью аналогично.) Поскольку выполнено (13.26), согласно описанию алгоритма находим коэффициенты $\lambda_\sigma, \lambda_p, \lambda_s$, удостоверяющие нормальность тройки $\langle \sigma, x_p, x_s \rangle$. Замечаем, что из (13.30) следует $\lambda_\sigma > 0$. Кроме того, $\lambda_p \neq 0$, так как в противном случае вектор x_s противоположно направлен вектору σ, и ввиду (13.31), $\sigma \in H,\ \sigma + x_s \in H$, что противоречит (13.26). При $\lambda_s = 0$ вектор x_p противоположно направлен σ, и для суммы $\sigma_{k+1} = \sigma + x_p$ соотношения (13.28), (13.29) выполнены при $t = k+1$. Поэтому далее считаем, что

$$\lambda_i > 0 \text{ при } i \in \{\sigma, p, s\}. \tag{13.32}$$

Рассмотрим три возможных варианта соотношений между коэффициентами $\lambda_\sigma, \lambda_p, \lambda_s$.

Вариант 1 ($\lambda_s \leq \lambda_\sigma < \lambda_p$).
Имеем $\pi_{k+1} = p,\ \sigma_{k+1} = \sigma + x_p$. С учетом (13.30), (13.32) получаем

$$-\sigma_{k+1} = -\sigma - x_p = \left(\frac{\lambda_p}{\lambda_\sigma} - 1\right) x_p + \frac{\lambda_s}{\lambda_\sigma} x_s \in C^0(x_p, x_s), \tag{13.33}$$

откуда $x_{s_{k+1}} = x_s$. Кроме того, имеем

$$y_1 \doteq -\frac{\lambda_s}{\lambda_\sigma} x_s \in H, \tag{13.34}$$

$$\sigma_{k+1} = \sigma + x_p = \sigma - \frac{\lambda_\sigma}{\lambda_p}\sigma - \frac{\lambda_s}{\lambda_p} x_s = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right)\sigma + \frac{\lambda_\sigma}{\lambda_p} y_1. \tag{13.35}$$

Из (13.34) и (13.35) вытекает соотношение (13.28) для $t = k+1$. Докажем (13.29).

Из (13.31) следует существование точки $z = \sigma - \gamma x_s \in H$, где $\gamma \geq 0$. Обозначив $z_1 = \frac{\lambda_s}{\lambda_\sigma \gamma + \lambda_s} z$, будем иметь $z_1 \in H$ и

$$\begin{aligned} z_1 &= \frac{\lambda_s}{\lambda_\sigma \gamma + \lambda_s}(\sigma - \gamma x_s) = \frac{\lambda_s}{\lambda_\sigma \gamma + \lambda_s}\sigma + \frac{\lambda_\sigma \gamma}{\lambda_\sigma \gamma + \lambda_s} y_1 \\ &= (1 - \lambda')\sigma + \lambda' y_1 \in [y_1, \sigma], \end{aligned} \tag{13.36}$$

где $\lambda' \doteq \frac{\lambda_\sigma \gamma}{\lambda_\sigma \gamma + \lambda_s}$. Если $\frac{\lambda_\sigma}{\lambda_p} \geq \lambda'$, то из (13.35) и (13.34) имеем $\sigma_{k+1} \in [y_1, z_1] \subset H$, что противоречит условию (13.26). Таким образом, $\frac{\lambda_\sigma}{\lambda_p} < \lambda' = \frac{\lambda_\sigma \gamma}{\lambda_\sigma \gamma + \lambda_s}$, откуда $\gamma - \frac{\lambda_\sigma}{\lambda_p}\gamma - \frac{\lambda_s}{\lambda_p} \doteq \alpha > 0$. Но тогда

$$\sigma_{k+1} - \alpha x_{s_{k+1}} = \sigma_{k+1} - \alpha x_s = \sigma - \frac{\lambda_\sigma}{\lambda_p}\sigma - \frac{\lambda_s}{\lambda_p}x_s - \left(\gamma - \frac{\lambda_\sigma}{\lambda_p}\gamma - \frac{\lambda_s}{\lambda_p}\right) x_s$$

$$= \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right)\sigma - \gamma x_s \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right) = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right)(\sigma - \gamma x_s) = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right) z$$

$$\in H \cap (\sigma_{k+1} + Q(-x_{s_{k+1}})),$$

что доказывает соотношение (13.29) при $t = k + 1$.

Вариант 2 ($\lambda_p \leq \lambda_\sigma < \lambda_s$).

Заметим, что ввиду (13.31) этот вариант несимметричен варианту 1.

Полагаем $\pi_{k+1} = s,\ \sigma_{k+1} = \sigma + x_s$. Пусть $z = \sigma - \gamma x_s,\ (\gamma \geq 0)$, — точка из H, существование которой вытекает из (13.31). Тогда

$$\sigma_{k+1} = \sigma + x_s \in [\sigma - \gamma x_s,\ \sigma + \frac{\lambda_s}{\lambda_\sigma} x_s] = [\sigma - \gamma x_s,\ -\frac{\lambda_p}{\lambda_\sigma} x_p] \subset H,$$

откуда вытекают соотношения (13.28), (13.29) при $t = k + 1$.

Вариант 3 ($\lambda_\sigma < \min\{\lambda_p, \lambda_s\}$).

Случай А ($\boldsymbol{\sigma + x_p \in G}$).

Имеем $\pi_{k+1} = p,\ \sigma_{k+1} = \sigma + x_p \in G$, т.е. соотношение (13.28) при $t = k + 1$ выполнено. Для доказательства (13.29) определим векторы y_1, z, z_1 и величины α, γ, λ', как в варианте 1. Тогда согласно (13.33) имеем $-\sigma_{k+1} \in C^0(x_p, x_s)$, откуда $x_{s_{k+1}} = x_s$ и $x_{p_{k+1}} \in C(-x_s, x_p)$, или

$$-x_{p_{k+1}} \in C(x_s, -x_p). \tag{13.37}$$

Если $\frac{\lambda_\sigma}{\lambda_p} \geq \lambda'$, то из (13.35) и (13.36) имеем

$$z_1 \in [\sigma_{k+1}, \sigma] \subset \sigma_{k+1} + Q(-x_p),$$

откуда

$$(\sigma_{k+1} + Q(-x_p)) \cap H \neq \emptyset. \tag{13.38}$$

Кроме того, $\alpha \leq 0$ и

$$\sigma_{k+1} - \alpha x_s = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right) z \in H \cap (\sigma_{k+1} + Q(x_s)). \tag{13.39}$$

Из (13.37)–(13.39) с учетом (13.30) получаем

$$(\sigma_{k+1} + Q(-x_{p_{k+1}})) \cap H \neq \emptyset,$$

т.е. (13.29) выполнено для $t = k + 1$.

Если же $\frac{\lambda_\sigma}{\lambda_p} < \lambda'$, то $\alpha > 0$ и

$$\sigma_{k+1} - \alpha x_{s_{k+1}} = \sigma_{k+1} - \alpha x_s = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right) z \in H \cap (\sigma_{k+1} + Q(-x_{s_{k+1}})),$$

т.е. (13.29) выполнено при $t = k + 1$.

Случай B ($\boldsymbol{\sigma + x_s \in G}$).

Имеем $\pi_{k+1} = s,\ \sigma_{k+1} = \sigma + x_s \in G$, т.е. (13.28) выполнено при $t = k + 1$. Докажем (13.29).

Ввиду (13.30), (13.32) имеем

$$-\sigma_{k+1} = \frac{\lambda_p}{\lambda_\sigma} x_p + \left(\frac{\lambda_s}{\lambda_\sigma} - 1\right) x_s \in C^0(x_p, x_s),$$

откуда с учетом (13.30),

$$\angle C(-x_s, -\sigma_{k+1}) < \pi, \tag{13.40}$$

$x_{s_{k+1}} \in C(x_s, -x_p) \subset C(x_s, \sigma_{k+1})$, или

$$-x_{s_{k+1}} \in C(-x_s, -\sigma_{k+1}). \tag{13.41}$$

Так как $(\sigma_{k+1} + Q(-\sigma_{k+1})) \cap H \neq \emptyset$ и ввиду (13.31)

$$(\sigma_{k+1} + Q(-x_s)) \cap H \supset (\sigma + Q(-x_s)) \cap H \neq \emptyset,$$

с учетом (13.41) и (13.40) получаем $(\sigma_{k+1} + Q(-x_{s_{k+1}})) \cap H \neq \emptyset$, что доказывает наследование свойства (13.29) при $t = k + 1$.

Случай C ($\boldsymbol{\sigma + x_p \notin G,\ \sigma + x_s \notin G}$).

Имеем $\pi_{k+1} = p,\ \pi_{k+2} = s,\ \sigma_{k+2} = \sigma + x_p + x_s$ и $\sigma_{k+2} \in C^0(x_p, x_s)$, откуда с учетом (13.30) получаем $x_{p_{k+2}} \in C(-x_s, x_p),\ x_{s_{k+2}} \in C(x_s, -x_p)$, или

$$-x_{p_{k+2}} \in C(x_s, -x_p), \tag{13.42}$$

$$-x_{s_{k+2}} \in C(-x_s, x_p). \tag{13.43}$$

Определив z_1, y_1, λ' и α как в варианте 1, замечаем, что случай $\sigma_{k+1} \in [z_1, \sigma]$ невозможен (вытекало бы, что $\sigma_{k+1} \in G$). Таким образом, $z_1 \in (\sigma_{k+1}, \sigma]$, откуда $\lambda' < \frac{\lambda_\sigma}{\lambda_p}$ и $\alpha < 0$.

Рассмотрим случаи $\alpha \geq -1$ и $\alpha < -1$.

Случай 1: $\boldsymbol{\alpha \in [-1, 0)}$.

Поскольку $\sigma_{k+1} - \alpha x_s \in [\sigma_{k+1}, \sigma_{k+2}]$ и $\sigma_{k+1} \notin G$, с учетом (13.22) и $\|\sigma_{k+2} - \sigma_{k+1}\| = \|x_s\| \leq 1$ получаем (13.28) для $t = k + 2$.

Покажем, что при $t = k + 2$ имеет место (13.29) для $x' = x_{s_{k+2}}$. Действительно,

$$\sigma_{k+2} - (1 + \alpha) x_s = \sigma_{k+1} - \alpha x_s = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right) z \in (\sigma_{k+2} + Q(-x_s)) \cap H. \tag{13.44}$$

Кроме того,

$$\sigma_{k+2} + \frac{\lambda_p(1+\alpha)}{\lambda_s + \lambda_\sigma\gamma} x_p = \sigma + x_s + \frac{\lambda_s + \lambda_\sigma\gamma + \lambda_p + \lambda_p\alpha}{\lambda_s + \lambda_\sigma\gamma} x_p$$
$$= \sigma + x_s - \frac{\lambda_p + \lambda_p\gamma}{\lambda_s + \lambda_\sigma\gamma} \cdot \frac{\lambda_\sigma}{\lambda_p}\sigma - \frac{\lambda_p + \lambda_p\gamma}{\lambda_s + \lambda_\sigma\gamma} \cdot \frac{\lambda_s}{\lambda_p} x_s$$
$$= \frac{\lambda_s + \lambda_\sigma\gamma - \lambda_\sigma - \lambda_\sigma\gamma}{\lambda_s + \lambda_\sigma\gamma}\sigma + \frac{\lambda_s + \lambda_\sigma\gamma - \lambda_s - \lambda_s\gamma}{\lambda_s + \lambda_\sigma\gamma} x_s$$
$$= \frac{\lambda_s - \lambda_\sigma}{\lambda_s + \lambda_\sigma\gamma} \cdot \sigma - \frac{\lambda_s - \lambda_\sigma}{\lambda_s + \lambda_\sigma\gamma} \cdot \gamma x_s = \frac{\lambda_s - \lambda_\sigma}{\lambda_s + \lambda_\sigma\gamma} \cdot (\sigma - \gamma x_s)$$
$$= \frac{\lambda_s - \lambda_\sigma}{\lambda_s + \lambda_\sigma\gamma} \cdot z \doteq \gamma' z \doteq z_2, \tag{13.45}$$

где $\gamma' \in (0,1)$. Поэтому

$$(\sigma_{k+2} + Q(x_p)) \cap H \neq \emptyset. \tag{13.46}$$

Из (13.43), (13.44), (13.46) с учетом (13.30) получаем

$$(\sigma_{k+2} + Q(-x_{s_{k+2}})) \cap H \neq \emptyset,$$

т.е. (13.29) выполняется при $t = k+2$.

Случай 2: $\boldsymbol{\alpha < -1}$.

Из (13.45) следует, что $z_2 = \sigma_{k+2} + \lambda_2 x_p \in H$, где $\lambda_2 = \frac{\lambda_p(1+\alpha)}{\lambda_s+\lambda_\sigma\gamma} \in (-1,0)$. Так как $[\sigma + x_s, \sigma_{k+2}] \cap H \neq \emptyset$, с учетом соотношений $\|\sigma_{k+2} - \sigma - x_s\| = \|x_p\| \leq 1$, $\sigma + x_s \notin G$ и свойства (13.22) получаем $\sigma_{k+2} \in G$, т.е. (13.28) выполнено для $t = k+2$. Покажем, что при $t = k+2$ имеет место (13.29) для $x' = x_{p_{k+2}}$.

Так как $z_2 \in \sigma_{k+2} + Q(-x_p)$, то

$$(\sigma_{k+2} + Q(-x_p)) \cap H \neq \emptyset. \tag{13.47}$$

Кроме того, из (13.44) следует, что

$$\sigma_{k+2} - (1+\alpha)x_s = \left(1 - \frac{\lambda_\sigma}{\lambda_p}\right) z \in (\sigma_{k+2} + Q(x_s)) \cap H. \tag{13.48}$$

Из (13.42), (13.47), (13.48) с учетом (13.30) получаем

$$(\sigma_{k+2} + Q(-x_{p_{k+2}})) \cap H \neq \emptyset,$$

т.е. (13.29) выполняется при $t = k+2$.

Лемма 13.25 доказана.

Вернемся к доказательству теоремы 13.21.

Ясно, что трудоемкость алгоритма на этапе 1 не превосходит $O(n \log n)$, поскольку как нумерация векторов, так и построение исходного дерева поиска T_0 могут быть выполнены с такой трудоемкостью. С этой же трудоемкостью мы реализуем этап 2,

поскольку на каждом шаге $k = 1, \ldots, n$ алгоритма построения перестановки π поиск вершин p_k, s_k и последующее удаление одной из них из дерева T_k могут быть выполнены за $O(\log n)$ действий. При этом предполагаем, что проверка включений $x \in H$ и $x \in G$, осуществляемая на каждом шаге этапа 2 для векторов $x = \sigma + x_p,\ x = \sigma + x_s$, может быть выполнена за константу действий. Это безусловно верно в тех частных случаях применения теоремы 13.21, которые будут рассмотрены в следствиях 14.7–14.9. Однако сформулированная в теореме оценка трудоемкости останется справедливой и в том случае, если каждая такая проверка потребует $O(\log n)$ действий.

Для завершения доказательства теоремы 13.21 остается показать, что получаемая алгоритмом $\mathcal{A}^*$ перестановка π задает нестрогое суммирование векторов X в области G. Указанное свойство перестановки легко доказывается индукцией по шагам $t = 0, 1, \ldots, n$. Базис индукции основывается на соотношениях $\sigma_t = 0 \in H \subset G$ $(t = 0, 1, \ldots, k_0)$, из которых следуют соотношения (13.28), (13.29) при $t = 0, 1, \ldots, k_0$, а индукционный шаг обеспечивается леммой 13.25.

Теорема 13.21 доказана.

14 Экстремальные задачи о нестрогом суммировании векторов в семействах полупространств

Определение 14.1 Будем говорить, что перестановка π задает нестрогое суммирование векторов из $X \subset \mathbb{R}^m$ в семействе областей $\mathcal{P} = \{G_i \mid i \in \mathbb{N}_\nu\}$, и обозначать этот факт $(X, \pi) \in S_m\mathcal{P}$, если π задает нестрогое суммирование в каждой из областей $G_i \in \mathcal{P},\ i \in \mathbb{N}_\nu$.

Для любых областей $G_i \subset \mathbb{R}^m$ справедливо

Замечание 14.2 *Перестановка π, задающая нестрогое суммирование векторов из X в пересечении областей $G = \cap_{i=1}^k G_i$, задает и нестрогое суммирование этих векторов в семействе областей $\{G_i \mid i \in \mathbb{N}_k\}$.* □

Обратное утверждение неверно. Действительно, при нестрогом суммировании в пересечении G областей заданного семейства как минимум каждая вторая частичная сумма должна находиться в G, в то время как при нестрогом суммировании в семействе областей область G может не содержать ни одной из n последовательных частичных сумм векторов из X (как показывает пример на рис. 14.1 с двумя полуплоскостями). Более того, область G может быть пустой.

В $\mathbb{R}^m$ зададим норму $\hat{s}$, единичный шар которой с центром в нуле определяется соотношением

$$B_{m,\hat{s}} = \{x \in \mathbb{R}^m \mid |x(i_1)| \le 1,\ |x(i_1) - x(i_2)| \le 1,\ \forall i_1, i_2 \in \mathbb{N}_m\}, \tag{14.1}$$

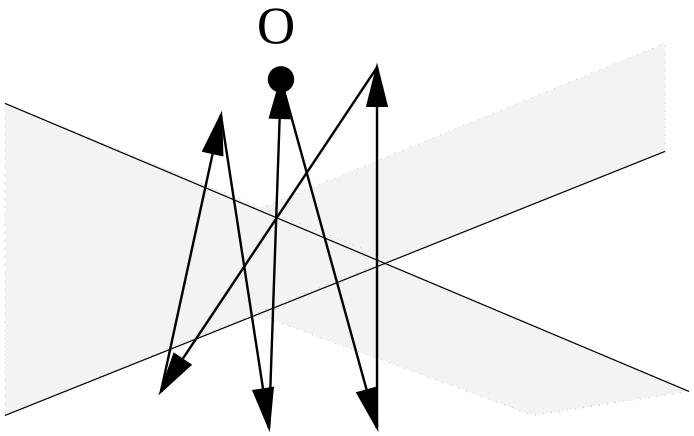


Рис. 14.1: Нестрогое суммирование в двух полуплоскостях, которое не является нестрогим суммированием в их пересечении

где $x = (x(1), \ldots, x(m))$. Норма $\hat{s}$ играет важную роль в приложениях нестрогого суммирования векторов к задачам теории расписаний (см. главы 1,4). Более наглядное представление о конструкции единичного шара нормы $\hat{s}$ дает следующая

Лемма 14.3 *$B_{m,\hat{s}} = \mathrm{conv}(B \cup (-B))$, где $B = [0,1]^m$.*

Доказательство. Очевидно включение $B \subseteq B_{m,\hat{s}}$. Так как $B_{m,\hat{s}}$ симметрично, то $-B \subseteq B_{m,\hat{s}}$. Так как $B_{m,\hat{s}}$ выпукло, то $B' \doteq \mathrm{conv}(B \cup (-B)) \subseteq B_{m,\hat{s}}$. Докажем обратное включение.

Пусть $z \in B_{m,\hat{s}}$. Докажем, что $z \in B'$. Обозначим $z_{\min} = \min_i z(i), z_{\max} = \max_i z(i)$, $\bar{e} = (1, 1, \ldots, 1) \in \mathbb{R}^m$. Тогда $-1 \leq z_{\min} \leq z_{\max} \leq 1$ и $z_{\max} - z_{\min} \leq 1$.

Если $z_{\min} \geq 0$ или $z_{\max} \leq 0$, то $z \in B$ или $z \in -B$ соответственно, откуда в обоих случаях имеем $z \in B'$. Пусть $z_{\min} < 0 < z_{\max}$. Определим векторы $z' = z - z_{\max}\bar{e}$, $z'' = z - z_{\min}\bar{e}$. Тогда для любого $i \in \mathbb{N}_m$ имеем $z'(i) = z(i) - z_{\max} \in [-1, 0]$ и $z''(i) = z(i) - z_{\min} \in [0, 1]$. Следовательно, $z' \in -B$ и $z'' \in B$. Таким образом,

$$B' = \mathrm{conv}(B \cup (-B)) \ni \frac{-z_{\min}}{z_{\max} - z_{\min}} z' + \frac{z_{\max}}{z_{\max} - z_{\min}} z''$$

$$= \left(\frac{-z_{\min}}{z_{\max} - z_{\min}} + \frac{z_{\max}}{z_{\max} - z_{\min}} \right) z + \left(\frac{z_{\max} z_{\min}}{z_{\max} - z_{\min}} - \frac{z_{\max} z_{\min}}{z_{\max} - z_{\min}} \right) \bar{e} = z.$$

Лемма 14.3 доказана.

Для заданного вектора $c \in \mathbb{R}^m$, $c \neq 0$ и числа β через $P(c, \beta)$ обозначим замкнутое полупространство $\{x \in \mathbb{R}^m \,|\, (c, x) \leq \beta\}$, а через $P^0(c, \beta)$ — открытое полупространство $\{x \in \mathbb{R}^m \,|\, (c, x) < \beta\}$ пространства $\mathbb{R}^m$.

Мы будем рассматривать оптимизационные задачи о нестрогом суммировании векторов в пространстве $\mathbb{R}^m$, каждая из которых характеризуется четырьмя компонентами:

— размерностью пространства (m);
— "арностью" целевой функции (l);
— набором векторов $C = \{c_1, \dots, c_l\} \subset \mathbb{R}^m$;
— целевой функцией $\theta(\beta_1, \dots, \beta_l)$ от l действительных переменных $\beta_i \in \mathbb{R},\ i \in \mathbb{N}_l$.

Все задачи будем рассматривать на минимум целевой функции.

Соответствующую массовую задачу будем обозначать НСВ(m, l, C, θ). Каждая индивидуальная задача $x \in$ НСВ(m, l, C, θ) однозначно определяется каким-либо конечным 0-семейством векторов $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$. Допустимым решением задачи x (будем называть ее также задачей X) является пара $(\pi(X); b(X))$, где $\pi(X) = (\pi_1, \dots, \pi_n)$ — перестановка чисел $\{1, 2, \dots, n\}$, а $b(X) = (\beta_1, \dots, \beta_l)$ — набор l действительных чисел, таких что $\pi(X)$ задает нестрогое суммирование векторов X в семействе полупространств $\mathcal{P} = \{P(c_1, \beta_1), \dots, P(c_l, \beta_l)\}$. Поскольку для любого конечного семейства векторов $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$ и любой перестановки $\pi(X) = (\pi_1, \dots, \pi_n)$ нетрудно найти числа $\beta_1, \dots, \beta_l$, для которых все частичные суммы $\{x_\pi^k \doteq \sum_{j=1}^k x_{\pi_j} \,|\, k \in \mathbb{N}_n\}$ содержатся внутри полупространств семейства $\mathcal{P}$, то для любой задачи X ее область допустимых решений $\mathrm{sol}(X)$ непуста.

Целью является нахождение допустимого решения $(\pi(X); b(X)) \in \mathrm{sol}(X)$, доставляющего минимум целевой функции $\theta(b(X))$.

Почему целевая функция зависит только от b и не зависит от π ? На самом деле, как мы сейчас убедимся, дело обстоит совсем не так, а прямо противоположным образом. Действительно, все функции θ, которые мы будем рассматривать, будут обладать свойством монотонной неубываемости по каждому из аргументов β_i. С другой стороны, свойство допустимости пары $(\pi(X); b(X))$ также монотонно неубывает по каждому из аргументов β_i, — в том смысле, что если определить характеристическую функцию $\chi(\pi(X); b(X))$, равную 0, если пара $(\pi(X); b(X))$ недопустима и 1 — в противном случае, то функция χ будет монотонно неубывающей по каждому из аргументов β_i. Чтобы убедиться в справедливости последнего утверждения, достаточно доказать существование таких значений $\bar\beta_1, \dots, \bar\beta_l$, что $\chi(\pi, \beta_1, \dots, \beta_l) = 1$, если и только если $\beta_i \geq \bar\beta_i,\ \forall\ i \in \mathbb{N}_l$.

Действительно, для каждого $i \in \mathbb{N}_l$ значение $\bar\beta_i$ определяется как

$$\bar\beta_i = \max_{k \in \mathbb{N}_n} \min\{(c_i, x_\pi^{k-1}), (c_i, x_\pi^k)\}. \tag{14.2}$$

Ясно, что при любом $\beta_i < \bar\beta_i$ перестановка π не обеспечивает нестрогого суммирования векторов из X в полупространстве $P(c_i, \beta_i)$, т.к. сразу две последовательные частичные суммы $x_\pi^{k^*-1}$ и $x_\pi^{k^*}$ лежат вне полупространства $P(c_i, \beta_i)$ (где k^* — значение k, на котором достигается максимум в правой части 14.2). С другой стороны, при $\beta_i \geq \bar\beta_i$ для любого k хотя бы одна из двух частичных сумм $\{x_\pi^{k-1}, x_\pi^k\}$ лежит в $P(c_i, \beta_i)$, что по-определению означает нестрогую суммируемость X в $P(c_i, \beta_i)$.

Наконец, по определению нестрогой суммируемости в семействе областей, имеем

$$\chi(\pi, \beta_1, \dots, \beta_l) = 1 \Leftrightarrow (X, \pi) \in S_m\{P(c_i, \beta_i)\},\ \forall\ i \in \mathbb{N}_l$$

$$\Leftrightarrow \beta_i \geq \bar\beta_i,\ \forall\ i \in \mathbb{N}_l$$

Т.о., минимум целевой функции $\theta(\beta_1,\ldots,\beta_l)$ будет достигаться на минимальных значениях $\{\beta_i\}$, которые все еще обеспечивают существование нестрогого суммирования векторов X в полупространствах семейства $\{P(c_i,\beta_i)\}$, — иначе говоря, на значениях $\{\bar\beta_i\}$, определенных согласно (14.2). Но нетрудно видеть, что эти значения однозначно определяются по паре (X,π), причем находятся простым алгоритмом линейной от n трудоемкости. Таким образом, логичнее было бы писать $\theta(b(X,\pi))$ (или при заданном X: $\theta(b(\pi))$), — отражая истинную зависимость функции θ от перестановки π. Мы, однако, будем по-прежнему использовать запись $\theta(b)=\theta(\beta_1,\ldots,\beta_l)$, предпочитая отражать более ясную математическую зависимость функции θ от величин $\{\beta_i\}$, чем ее "спрятанную" алгоритмическую зависимость от перестановки π, как то: $\theta_1(b)=\sum_{i=1}^{m+1}\beta_i$, $\theta_4(b)=\max\{\beta_1+\beta_2,\beta_2+\beta_3\}$, и т.п.

Сформулируем несколько задач о нахождении нестрогого суммирования векторов $X\subset\mathbb{R}^m$ в экстремальных семействах полупространств пространства $\mathbb{R}^m$. Будем обозначать их НСВi(m) для $i=1,2,\ldots$, где под каждым номером i будет скрываться задача НСВ(m,l,C,θ) с конкретным набором векторов C и конкретной целевой функцией θ. Обозначение НСВi без параметра m означает, что данная задача определена лишь в пространстве $\mathbb{R}^2$.

Задача НСВ1(m). *Для заданного 0-семейства векторов $X=\{x_1,\ldots,x_n\}\subset\mathbb{R}^m$ требуется найти перестановку $\pi=(\pi_1,\ldots,\pi_n)$ и набор вещественных чисел $b=(\beta_1,\ldots,\beta_{m+1})$ такие, что выполнено $(X,\pi)\in S_m\mathcal{P}_1(m,b)$ для*

$$\mathcal{P}_1(m,b)=\{P(e_1-e_2,\beta_1),\ldots,P(e_{m-1}-e_m,\beta_{m-1}),P(e_m,\beta_m),P(-e_1,\beta_{m+1})\},$$

а функционал

$$\theta_1(b)=\sum_{i=1}^{m+1}\beta_i$$

принимает минимальное возможное значение.

Задача НСВ2(m). *Для заданного 0-семейства векторов $X=\{x_1,\ldots,x_n\}\subset\mathbb{R}^m$ требуется найти перестановку $\pi=(\pi_1,\ldots,\pi_n)$ и набор вещественных чисел $b=(\beta_1,\ldots,\beta_m)$ такие, что выполнено $(X,\pi)\in S_m\mathcal{P}_2(m,b)$ для*

$$\mathcal{P}_2(m,b)=\{P(e_1-e_2,\beta_1),\ldots,P(e_{m-1}-e_m,\beta_{m-1}),P(e_m,\beta_m)\},$$

а функционал

$$\theta_2(b)=\sum_{i=1}^{m}\beta_i$$

принимает минимальное возможное значение.

Пусть $N_6\doteq\{1,\ldots,6\}$. На множестве N_6 введем циклический порядок $1\to2\to\cdots\to6\to1$, определив для каждого элемента $i\in N_6$ его предыдущий ($p(i)$) и последующий ($s(i)$) элементы. Через I обозначим множество "несоседних" пар из N_6, т.е.

$$I\doteq\{(i,j)\,|\,i,j\in N_6;\ i\neq j;\ i\neq s(j);\ i\neq p(j)\}.$$

Задача НСВ3. *Для заданного 0-семейства векторов* $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ *и векторов* $a_1 = e_1$, $a_2 = e_2$, $a_3 = e_2 - e_1$, $a_4 = -e_1$, $a_5 = -e_2$, $a_6 = e_1 - e_2$ *требуется найти перестановку* $\pi = (\pi_1, \ldots, \pi_n)$ *и набор вещественных чисел* $b = (\beta_1, \ldots, \beta_6)$ *такие, что выполнено* $(X, \pi) \in S_2\mathcal{P}_3(2, b)$ *для* $\mathcal{P}_3(2, b) = \{P(a_1, \beta_1), \ldots, P(a_6, \beta_6)\}$, *а функционал*

$$\theta_3(b) = \max_{(i,j)\in I}(\beta_i + \beta_j)$$

принимает минимальное возможное значение.

Задача НСВ4. *Для заданного 0-семейства векторов* $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ *требуется найти перестановку* $\pi = (\pi_1, \ldots, \pi_n)$ *и набор вещественных чисел* $b = (\beta_1, \beta_2, \beta_3)$ *такие, что выполнено* $(X, \pi) \in S_2\mathcal{P}_4(2, b)$ *для* $\mathcal{P}_4(2, b) = \mathcal{P}_1(2, b)$, *а функционал*

$$\theta_4(b) = \max\{\beta_1 + \beta_2,\ \beta_2 + \beta_3\}$$

принимает минимальное возможное значение.

Задача НСВ5(m). *Для заданного 0-семейства векторов* $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ *требуется найти перестановку* $\pi = (\pi_1, \ldots, \pi_n)$ *и набор вещественных чисел* $b = (\beta_1, \ldots, \beta_m)$ *такие, что выполнено* $(X, \pi) \in S_m\mathcal{P}_5(m, b)$ *для* $\mathcal{P}_5(m, b) = \{P(e_1, \beta_1), \ldots, P(e_m, \beta_m)\}$, *а функционал*

$$\theta_5(b) = \max_{i\in\mathbb{N}_m} \beta_i$$

принимает минимальное возможное значение.

Задача НСВ6. *Для заданного 0-семейства векторов* $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ *требуется найти перестановку* $\pi = (\pi_1, \ldots, \pi_n)$ *и набор вещественных чисел* $b = (\beta_1, \ldots, \beta_4)$ *такие, что выполнено* $(X, \pi) \in S_2\mathcal{P}_6(2, b)$ *для* $\mathcal{P}_6(2, b) = \{P(e_1, \beta_1), P(e_2, \beta_2), P(e_2 - e_1, \beta_3), P(e_1 - e_2, \beta_4)\}$ *а функционал*

$$\theta_6(b) = \max\{\beta_1 + \beta_3,\ \beta_2 + \beta_4\}$$

принимает минимальное возможное значение.

Вместе с задачами о нестрогом суммировании векторов в семействах полупространств мы будем рассматривать их аналоги, которые будем обозначать СВi, и которые будут отличаться от соответствующих задач НСВi заменой требования о нестрогом суммировании на строгое.

А сейчас вернемся к задачам НСВi и докажем лемму, утверждающую возможность сведения задачи НСВ2 в пространстве размерности m к задаче НСВ1 в пространстве размерности $m - 1$. Для этого в пространстве $\mathbb{R}^m$ определим гиперплоскость $H = \{x \in \mathbb{R}^m \,|\, x(1) = 0\}$. Пусть x' обозначает ортогональную проекцию вектора $x \in \mathbb{R}^m$ на гиперплоскость H, т.е., $x' = x - x(1)e_1$. Для заданного семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ через X_H будем обозначать семейство проекций $\{x'_1, \ldots, x'_n\}$ векторов $x_j \in X$ на гиперплоскость H.

Лемма 14.4 *Пусть для некоторого $m \geq 2$ имеется алгоритм $\mathcal{A}'$, который по всякому входу $X' = \{x'_1, \ldots, x'_n\} \subset \mathbb{R}^{m-1}$ задачи* НСВ1$(m-1)$ *с трудоемкостью $\mathcal{T}_{\mathcal{A}'}(n)$ находит ее приближенное решение $(\pi'(X'); b'(X'))$. Тогда существует алгоритм $\mathcal{A}$, который по всякому входу $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ задачи* НСВ2(m) *с трудоемкостью $O(\mathcal{T}_{\mathcal{A}'}(n))$ находит ее решение $(\pi(X); b(X))$ с оценкой целевой функции*

$$\theta_2(b(X)) \leq \theta_1(b'(X_H)). \tag{14.3}$$

Доказательство. Задачи НСВ2(m) и НСВ1$(m-1)$ обозначим для краткости через A и B. Согласно общей схеме сведения (стр. 42), нам достаточно определить функции $f: I_A \to I_B$ и $g: I_A \times \mathrm{sol}_B \to \mathrm{sol}_A$.

Отождествляя каждую индивидуальную задачу $x \in I_A$ с некоторым конечным 0-семейством векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$, определим функцию f, которая семейству векторов X ставит в соответствие семейство X_H их проекций на гиперплоскость H. Т.к. X_H сохраняет свойство быть 0-семейством, то X_H является входом задачи B в пространстве H, т.е. $X_H \in I_B$.

Чтобы определить функцию g, рассмотрим некоторый пример $X \in I_A$ задачи A и некоторое решение $(\pi'(X_H); b'(X_H)) \in \mathrm{sol}_B X_H$ для соответствующего примера X_H задачи B, где $b'(X_H) \doteq (\beta'_2, \beta'_3, \ldots, \beta'_m, \beta'_1)$. Найдем допустимое решение $(\pi(X); b(X))$ задачи $X \in I_A$, удовлетворяющее оценке (14.3).

По определению задачи B в пространстве H, перестановка π' обеспечивает нестрогое суммирование векторов из X_H в семействе полупространств

$$\mathcal{P}' = \{P'(e_2 - e_3, \beta'_2), P'(e_3 - e_4, \beta'_3), \ldots, P'(e_{m-1} - e_m, \beta'_{m-1}), P'(e_m, \beta'_m), P'(-e_2, \beta'_1)\}$$

$\doteq \{P'(a_i, \beta'_i) \,|\, i \in \mathbb{N}_m\}$ пространства H, т.е.

$$(X_H, \pi') \in S_{m-1}\mathcal{P}'. \tag{14.4}$$

Пусть $y_\nu = \sum_{j=1}^{\nu} x_{\pi'_j}$ — ν-я вершина траектории суммирования векторов $x_j \in X$ согласно перестановке π'. Найдем такое k, что $y_k(1) = \max_{\nu \in \mathbb{N}_n} y_\nu(1)$, и в качестве искомой перестановки $\pi(X) = (\pi_1, \ldots, \pi_n)$ возьмем циклический сдвиг перестановки π' на k элементов:

$$\pi_j = \begin{cases} \pi'_{j+k} & \text{при } j + k \leq n, \\ \pi'_{j+k-n} & \text{при } j + k > n. \end{cases}$$

Чтобы определить вторую половину решения — вектор b, каждому полупространству $P'(a, \beta) \subset H$ при $a \in H$ поставим в соответствие полупространство $P(a, \beta) \subset \mathbb{R}^m$ с теми же значениями вектора a и числа β, и через $\mathcal{P}$ обозначим семейство полупространств $P \subset \mathbb{R}^m$, соответствующих полупространствам $P' \in \mathcal{P}'$. Обозначим $y'_\nu = \sum_{j=1}^{\nu} x'_{\pi'_j}$. Поскольку каждый вектор x'_j отличается от x_j на вектор, коллинеарный орту e_1, имеем $y_\nu - y'_\nu = \alpha_\nu e_1$ для некоторого $\alpha_\nu \in \mathbb{R}$. Отсюда для любого $a \in H$ получаем: $(y_\nu, a) = (y'_\nu, a) + \alpha_\nu (e_1, a) = (y'_\nu, a)$. Если $y'_\nu \in P'(a_i, \beta'_i) \in \mathcal{P}'$, то $y_\nu \in P(a_i, \beta'_i) \in \mathcal{P}$. Таким образом, (14.4) влечет

$$(X, \pi') \in S_m \mathcal{P}, \tag{14.5}$$

т.е. нестрогую суммируемость векторов исходного семейства X в семействе полупространств $\mathcal{P}$.

Обозначим $z_\nu = \sum_{j=1}^{\nu} x_{\pi_j}$. Тогда с учетом $\sum_{j=1}^{n} x_j = 0$ имеем

$$z_\nu = \begin{cases} y_{k+\nu} - y_k & \text{при } k+\nu \le n, \\ y_{k+\nu-n} - y_k & \text{при } k+\nu > n. \end{cases}$$

Отсюда $z_\nu(1) \le 0$, или

$$z_\nu \in P(e_1, 0),\ \nu \in \mathbb{N}_n. \tag{14.6}$$

Относительно семейства X циклический сдвиг перестановки π' на k элементов равносилен переносу начала координат в k-ю вершину траектории суммирования векторов $\{x_j\}$ (т.е. в точку y_k). При этом траектория суммирования совпадает с исходной траекторией с точностью до параллельного переноса. В результате свойство нестрогой суммируемости векторов X выполнится (при суммировании согласно перестановке π) относительно полупространств семейства $\mathcal{P} + y \doteq \{P + y \,|\, P \in \mathcal{P}\}$, где $y \doteq -y_k$, т.е. $(X, \pi) \in S_m\{\mathcal{P} + y\}$. Поскольку

$$x \in P(a_i, \beta_i') + y \Leftrightarrow (a_i, x - y) \le \beta_i' \Leftrightarrow (a_i, x) \le \beta_i' + (a_i, y) \doteq \beta_i \Leftrightarrow x \in P(a_i, \beta_i),$$

для так определенных чисел β_i имеем $\mathcal{P} + y = \{P(a_i, \beta_i) \,|\, i \in \mathbb{N}_m\}$. Покажем, что смещение всех полупространств семейства $\mathcal{P}$ на один и тот же вектор y не меняет значения целевой функции θ_1. Действительно,

$$\sum \beta_i = \sum \beta_i' + \left(\sum_{i=1}^{m} a_i, y\right) = \sum \beta_i' + ((e_2 - e_3) + (e_3 - e_4) + \cdots$$

$$+(e_{m-1} - e_m) + e_m - e_2,\ y) = \sum \beta_i'.$$

Наконец, из (14.6) следует что если $z_\nu \in P(a_1, \beta_1) \doteq P(-e_2, \beta_1)$, то

$$z_\nu \in P(e_1, 0) \cap P(-e_2, \beta_1) \subset P(e_1 - e_2, \beta_1).$$

Это означает, что перестановка π обеспечивает нестрогую суммируемость векторов X в полупространстве $P(e_1 - e_2, \beta_1)$. Заменяя в семействе $\mathcal{P} + y$ полупространство $P(-e_2, \beta_1)$ на полупространство $P(e_1 - e_2, \beta_1)$, приходим к выводу, что перестановка π обеспечивает нестрогое суммирование векторов из X в семействе полупространств

$$\{P(e_1 - e_2, \beta_1), P(e_2 - e_3, \beta_2), \ldots, P(e_{m-1} - e_m, \beta_{m-1}), P(e_m, \beta_m)\}.$$

Таким образом, для $b(X) \doteq (\beta_1, \ldots, \beta_m)$ пара $(\pi(X); b(X))$ является искомым решением задачи НСВ2(m) со значением целевой функции

$$\theta_2(b(X)) = \sum_{i=1}^{m} \beta_i = \sum_{i=1}^{m} \beta_i' = \theta_1(b'(X_H)).$$

Тудоемкость нахождения решения $(\pi(X); b(X))$ складывается из (не менее чем линейной от n) трудоемкости алгоритма $\mathcal{A}'$ нахождения решения $(\pi'(X_H); b'(X_H))$ задачи B и линейной от n трудоемкости последующей процедуры вычисления вектора y, набора чисел b и перестановки π, что в сумме дает оценку трудоемкости $O(\mathcal{T}_{\mathcal{A}'}(n))$.

Лемма 14.4 доказана.

Если норму векторов исходной совокупности $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$, ограничить сверху единицей (как мы делали это в задаче компактного суммирования векторов), то появляется возможность решения сформулированных выше задач HCBi с гарантированными верхними оценками на значение целевой функции. В частности, из леммы 14.4 может быть получено следующее

Следствие 14.5 *Пусть для некоторого $m \geq 2$ имеется алгоритм $\mathcal{A}'$, который для всякого $\hat{s}$-семейства векторов $X' = \{x'_1, \ldots, x'_n\} \subset \mathbb{R}^{m-1}$ с трудоемкостью $\mathcal{T}_{\mathcal{A}'}(n)$ находит приближенное решение $(\pi'(X'); b'(X'))$ задачи* HCB1$(m-1)$ *с оценкой $\theta_1(b') \leq \theta^*$. Тогда существует алгоритм трудоемкости $O(\mathcal{T}_{\mathcal{A}'}(n))$, который по всякому $\hat{s}$-семейству векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$, поданному на вход задачи* HCB2(m)*, находит ее приближенное решение с оценкой*

$$\theta_2(b) \leq \theta^*.$$

Доказательство. Пусть на входе задачи HCB2(m) задано $\hat{s}$-семейство векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$. Из определения нормы $\hat{s}$ (стр. 107) следует, что $\hat{s}$-норма проекции x'_j каждого вектора x_j на гиперплоскость H также не превосходит 1. И т.к. $\sum x'_j = 0$, то X_H является $\hat{s}$-семейством в пространстве H. Согласно алгоритму $\mathcal{A}'$ находим решение $(\pi'(X_H); b'(X_H))$ задачи $X_H \in$HCB1$(m-1)$ с оценкой $\theta_1(b'(X_H)) \leq \theta^*$. По нему, согласно лемме 14.4, находим решение $(\pi(X); b(X))$ задачи $X \in$HCB2(m) с оценкой $\theta_2(b(X)) \leq \theta_1(b'(X_H))$, откуда следует искомая оценка $\theta_2(b(X)) \leq \theta^*$. □

Нетрудно понять, что лемма 14.4 останется справедливой, если условия нестрогого суммирования в заданных полупространствах заменить на строгое. В результате приходим к следующему результату.

Лемма 14.6 *Пусть для некоторого $m \geq 2$ имеется алгоритм $\mathcal{A}'$, который по всякому входу $X' = \{x'_1, \ldots, x'_n\} \subset \mathbb{R}^{m-1}$ задачи* CB1$(m-1)$ *с трудоемкостью $\mathcal{T}_{\mathcal{A}'}(n)$ находит ее приближенное решение $(\pi'(X'); b'(X'))$. Тогда существует алгоритм $\mathcal{A}$, который по всякому входу $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ задачи* CB2(m) *с трудоемкостью $O(\mathcal{T}_{\mathcal{A}'}(n))$ находит ее решение $(\pi(X); b(X))$ с оценкой целевой функции*

$$\theta_2(b(X)) \leq \theta_1(b'(X_H)). \quad \square \tag{14.7}$$

Далее будем рассматривать задачи HCBi в пространстве $\mathbb{R}^2$.

Пусть $G_i(b)$ обозначает пересечение областей нестрогого суммирования в задаче HCBi:

$$G_i(b) = \bigcap_{P \in \mathcal{P}_i(2,b)} P.$$

Приведем следствия из теорем 13.8 и 13.21, позволяющие эффективно находить приближенные решения задач НСВi в пространстве размерности 2. Теоремы 13.8 и 13.21 применяем к областям $G_i(b)$ при соответствующих значениях векторов b.

Прежде всего заметим, что область $G_1(b) = G_4(b)$ при $b = (1, 1, 1)$ является треугольником, описанным вокруг единичного шара $B_{2,\hat{s}}$, а область $G_3(b')$ в задаче НСВ3 при $b' = (5/4, \ldots, 5/4)$ является шаром радиуса $5/4$ (см. рис. 14.2). Выполнив линейное преобразование пространства $\mathbb{R}^2$, при котором единичные орты e_1, e_2 переходят в единичные векторы с углом 120° между ними, мы получим такое представление векторного пространства, в котором единичный шар $B_{2,\hat{s}}$ нормы $\hat{s}$ будет представлен правильным шестиугольником с единичными сторонами, область $G_3(b')$ становится правильным шестиугольником со стороной 5/4, а область $G_1(b)$ — правильным треугольником, описанным вокруг единичного шара $B_{2,\hat{s}}$ (см. рис. 14.2В).

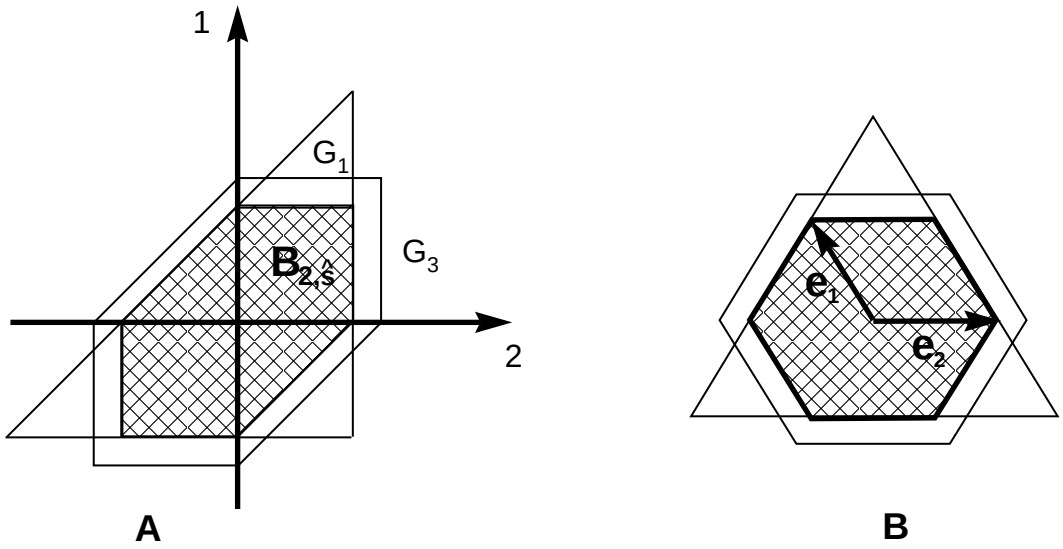


Рис. 14.2: Области G_1 (G_4) и G_3

Поскольку каждая из областей $G_1(b), G_3(b')$ замкнута, содержит $B_{2,\hat{s}}$ и всякая хорда области, пересекающая $B_{2,\hat{s}}$, имеет длину (в норме $\hat{s}$) не меньше единицы, то в силу замечания 13.22 условие (13.22) теоремы 13.21 выполнено. Это обеспечивает возможность нестрогого суммирования векторов всякого $\hat{s}$-семейства X в каждой из указанных областей. Отсюда, согласно замечанию 14.2, вытекает нестрогая суммируемость векторов из X в соответствующих семействах полуплоскостей. При этом искомые перестановки векторов находятся эффективно, а целевые функции задач НСВ1(2), НСВ3 и НСВ4 принимают значения:

$$\theta_1(b) = 3,\ \theta_3(b') = 2.5,\ \theta_4(b) = 2.$$

С учетом вышесказанного, мы можем сформулировать следующие следствия из теоремы 13.21.

Следствие 14.7 *Существует алгоритм трудоемкости $O(n\log n)$, который для всякого входа задачи* НСВ1(2) *с n векторами находит ее приближенное решение с оценкой*

$$\theta_1(b) \le 3.$$

Следствие 14.8 *Существует алгоритм трудоемкости $O(n\log n)$, который для всякого входа задачи* НСВ3 *с n векторами находит ее приближенное решение с оценкой*

$$\theta_3(b) \le 2.5.$$

Следствие 14.9 *Существует алгоритм трудоемкости $O(n\log n)$, который для всякого входа задачи* НСВ4 *с n векторами находит ее приближенное решение с оценкой*

$$\theta_4(b) \le 2.$$

Перейдем к рассмотрению задач НСВ2(2), НСВ5(2) и НСВ6.

Легко устанавливается следующее свойство областей G_2, G_5, G_6.

Утверждение 14.10 *Пусть заданы векторы $b' = (\beta'_1, \beta'_2)$ $b'' = (\beta''_1, \beta''_2)$ и $b''' = (\beta'''_1, \ldots, \beta'''_4)$ с неотрицательными компонентами. Тогда для любой O-хорды h' множества $G_2(b')$, любой O-хорды h'' множества $G_5(b'')$ и любой O-хорды h''' множества $G_6(b''')$ справедливы неравенства:*

$$\|h'\|_{\hat{s}} \ge \beta'_1 + \beta'_2;$$

$$\|h''\|_{\hat{s}} \ge (\sqrt{\beta''_1} + \sqrt{\beta''_2})^2;$$

$$\|h'''\|_{\hat{s}} \ge \min\{\beta'''_3 + \beta'''_4,\ \beta'''_1 + \beta'''_3,\ \beta'''_2 + \beta'''_4,\ \beta'''_1 + \beta'''_2 + 2\sqrt{\beta'''_1\beta'''_2}\}$$ □

С учетом утверждения 14.10 легко выбрать значения компонент $\{\beta'_i\}$, $\{\beta''_i\}$, $\{\beta'''_i\}$ векторов b', b'', b''', при которых всякая O-хорда h' множества $G_2(b')$, всякая O-хорда h'' множества $G_5(b'')$ и всякая O-хорда h''' множества $G_6(b''')$ имеют длину в норме $\hat{s}$ не меньше 1. В частности, это свойство O-хорд выполняется при $b' = (\beta, 1-\beta)$, $\forall\beta \in [0,1]$ и при $b'' = (1/4, 1/4)$, $b''' = (1/2, \ldots, 1/2)$ (см. области $G_5(b'')$ и $G_6(b''')$ на рис. 14.3 и 14.4), что обеспечивает выполнение условий теоремы 13.8.

Из теоремы 13.8 и замечания 14.2 получаем следующие следствия.

Следствие 14.11 *Существует алгоритм трудоемкости $O(n\log n)$, который для всякого входа задачи* НСВ5(2) *с n векторами находит ее приближенное решение с неулучшаемой оценкой*

$$\theta_5(b) \le 1/4 \doteq \theta_5^*.$$

Следствие 14.12 *Существует алгоритм трудоемкости $O(n\log n)$, который для всякого входа задачи* НСВ6 *с n векторами находит ее приближенное решение с неулучшаемой оценкой*

$$\theta_6(b) \le 1 \doteq \theta_6^*.$$

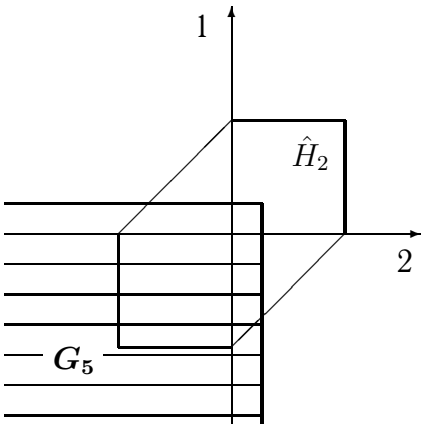


Рис. 14.3: Область G_5

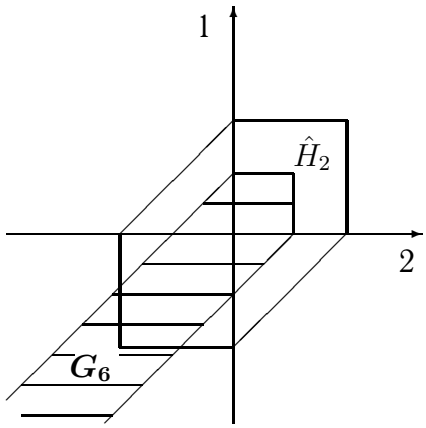


Рис. 14.4: Область G_6

Для задачи НСВ2 из теоремы 13.8 и замечания 14.2 с учетом замечания 13.16 получаем следующее следствие, используемое при доказательстве леммы 39.7 (стр. 237).

Следствие 14.13 *Для любого $\beta \in [0,1]$ и любого $\hat{s}$-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ с трудоемкостью $O(n \log n)$ находятся перестановки π' и π'' индексов $\{1, 2, \ldots, n\}$, обладающие следующими свойствами.*

- *Перестановка π' задает такой порядок суммирования векторов из X, который обеспечивает их строгое суммирование в полуплоскости $P(e_1 - e_2, \beta)$ и нестрогое — в полуплоскости $P(e_2, 1 - \beta)$;*
- *Перестановка π'' задает такой порядок суммирования векторов из X, который обеспечивает их нестрогое суммирование в полуплоскости $P(e_1 - e_2, \beta)$ и строгое — в полуплоскости $P(e_2, 1 - \beta)$.*

Доказательство. Действительно, для любого $\beta \in [0,1]$ теорема 13.8 гарантирует нахождение перестановки π, обеспечивающей нестрогое суммирование векторов исходного семейства в области $G \doteq P(e_1 - e_2, \beta) \cap P(e_2, 1 - \beta)$.

Определим векторы $a = -e_1 - e_2$ и $b = -e_1$. Тогда согласно (13.2) имеем $b \in L(a)$, и значит конус $A(a,b)$ не пуст. Нетрудно проверить, что рецессивный конус 0^+G полиэдра G совпадает с конусом $A(a,b)$. Действительно, для любого вектора $x = \lambda a + \mu b \in A(a,b),\ \lambda \geq 0,\ \mu \geq 0$, луч $Q(x)$ целиком лежит в G, и наоборот: если для вектора $x = \lambda a + \mu b \neq 0$ выполнено $Q(x) \subset G$, то $\lambda \geq 0$ и $\mu \geq 0$, т.е. $x \in A(a,b)$.

Из (13.2) также получаем:

$$L(a) = P(e_1 - e_2, 0) \subseteq P(e_1 - e_2, \beta),\ R(b) = P(e_2, 0) \subseteq P(e_2, 1 - \beta),$$

откуда $G \cup L(a) \subseteq P(e_1 - e_2, \beta),\ G \cup R(b) \subseteq P(e_2, 1-\beta)$. Реализуя в алгоритме $\mathcal{A}_{13.1}$ стратегию выбора вектора x_s в ситуации леммы 13.13, получаем перестановку π', обеспечивающую (согласно замечанию 13.16, стр. 96) строгое суммирование в области $G \cup L(a)$, а значит — и в области $P(e_1 - e_2, \beta)$. Наоборот, выбирая каждый раз в ситуации леммы 13.13 вектор x_p, получим перестановку π'', обеспечивающую строгое суммирование векторов в области $G \cup R(b)$, а значит — в области $P(e_2, 1-\beta)$. □

Неулучшаемость оценок, сформулированных в следствиях 14.11, 14.12, вытекает из доказываемой ниже теоремы 14.14.

Пусть задано рациональное число $\lambda \in (0,1)$. Сформулируем две задачи на распознавание, которые назовем НСВ5$(2,\lambda)$ и НСВ6(λ).

НСВ5(2,λ).
УСЛОВИЕ: заданы число $D \in \mathcal{Z}^+$ и семейство векторов $X = \{x_1, \ldots, x_n\} \subset \mathcal{Z}^2$ такие, что $\|x_j\|_{\hat{s}} \le D,\ \Sigma(X) = 0$.
ВОПРОС: существуют ли перестановка $\pi = (\pi_1, \ldots, \pi_n)$ и вектор b такие, что $(X, \pi) \in S_2\mathcal{P}_5(2, b)$ и $\theta_5(b) \le \lambda\theta_5^* D$?

НСВ6(λ).
УСЛОВИЕ: то же.
ВОПРОС: существуют ли перестановка $\pi = (\pi_1, \ldots, \pi_n)$ и вектор b такие, что $(X, \pi) \in S_2\mathcal{P}_6(2, b)$ и $\theta_6(b) \le \lambda\theta_6^* D$?

Теорема 14.14 *При любом фиксированном рациональном числе $\lambda \in (0,1)$ задачи* НСВ5$(2,\lambda)$ *и* НСВ6(λ) *являются NP-полными в сильном смысле.*

Доказательство. Пусть задано число $\lambda \in (0,1)$. Положим $t = \left\lceil \frac{3\lambda}{2(1-\lambda)} \right\rceil$, $k = 2t+1$, и будем к каждой из задач НСВ5$(2,\lambda)$,НСВ6(λ) сводить уже известную нам NP-полную в сильном смысле задачу k-РАЗБИЕНИЕ при данном k. (Очевидно, $k \ge 3$.)

Пусть задан вход задачи k-РАЗБИЕНИЕ, т.е. число $B \in \mathcal{Z}^+$, множество $A = \{1, \ldots, km\}$ и размеры $s(i) \in \mathcal{Z}^+$ всех элементов $i \in A$ такие, что $\sum_{i \in A} s(i) = Bm$ и

$$B/(k+1) < s(i) < B/(k-1). \tag{14.8}$$

Положим $n = km+3m+3+2t,\ B' = 2tB,\ D' = \lceil B'/\lambda \rceil$. Тогда $\lceil D'\lambda/2 \rceil \ge \lceil (B'/\lambda)(\lambda/2) \rceil = \lceil B'/2 \rceil = B'/2$. С другой стороны, $\lceil D'\lambda/2 \rceil \le \lceil (B'/\lambda + 1)\lambda/2 \rceil = \lceil B'/2 + \lambda/2 \rceil = B'/2$. Таким образом,

$$\lceil D'\lambda/2 \rceil = B'/2. \tag{14.9}$$

Определим семейство чисел $Z = \{z_1, \ldots, z_n\}$:

$$z_i = \begin{cases} s(i) \cdot 2t, & i = 1, \ldots, km; \\ & (A - \text{числа}) \\ \lfloor (D - B')/2 \rfloor \doteq \alpha', & i = km+1, \ldots, (k+1)m+1; \\ & (\alpha' - \text{числа}) \\ \lceil (D - B')/2 \rceil \doteq \alpha'', & i = (k+1)m+2, \ldots, (k+2)m+2; \\ & (\alpha'' - \text{числа}) \\ -D \doteq \gamma, & i = (k+2)m+3, \ldots, (k+3)m+3; \\ & (\gamma - \text{числа}) \\ B \doteq \delta, & i = (k+3)m+4, \ldots, n. \\ & (\delta - \text{числа}) \end{cases}$$

Убеждаемся, что

$$z_i < B \le \alpha', \ \forall i \in A. \tag{14.10}$$

(Действительно, из определения t имеем $t \ge \frac{3\lambda}{2(1-\lambda)}$, откуда

$$2t + 3 \le 2t/\lambda, \ (2t+3)B \le \lceil 2tB/\lambda \rceil, \ 3B \le D' - B', \ B \le \lfloor (D' - B')/3 \rfloor \le \alpha'.$$

Неравенства $z_i < B$ для $i \in A$ следуют из (14.8).) Таким образом, имеем $|z_i| \le D'$ для любого $i \in \mathbb{N}_n$ и $\sum z_i = 0$.

Семейство векторов $X = \{x_i = (z_i, -z_i) \,|\, i \in \mathbb{N}_n\}$ удовлетворяет условию задачи НСВ5$(2, \lambda)$ при $D = 2D'$. Покажем, что положительный ответ на вопрос в задаче НСВ5$(2, \lambda)$ имеет место, если и только если в исходной задаче k-РАЗБИЕНИЕ существует такое разбиение $\{A_1, \ldots, A_m\}$ множества A, что

$$\sum_{i \in A_j} s(i) = B, \ j \in \mathbb{N}_m.$$

Пусть существует искомое разбиение $\{A_1, \ldots, A_m\}$. Определим перестановку π^5 номеров $\{1, 2, \ldots, n\}$:

$$\pi^5 = (\delta_1 \cdots \delta_t \alpha''_1 \gamma_1 \alpha'_1 A_1 \alpha''_2 \gamma_2 \alpha'_2 A_2 \cdots \alpha''_m \gamma_m \alpha'_m A_m \alpha''_{m+1} \gamma_{m+1} \alpha'_{m+1} \delta_{t+1} \cdots \delta_{2t}),$$

где A_i — любая перестановка номеров из A_i, $(\varphi_1, \ldots, \varphi_{m+1})$ — любая перестановка номеров φ-чисел, $\varphi \in \{\alpha', \alpha'', \gamma\}$, $(\delta_1, \ldots, \delta_{2t})$ — любая перестановка номеров δ-чисел. Ясно, что перестановка π^5 задает нестрогое суммирование векторов из X в семействе областей $\mathcal{P}_5(2, b)$ при $b = (B'/2, B'/2)$, что дает оценку $\theta_5(b) = B'/2 \le D'\lambda/2 = \lambda\theta^*_5 D$. Докажем обратную импликацию.

Пусть некоторая перестановка π задает нестрогое суммирование векторов из X в семействе областей $\{P(e_1, \beta_1), P(e_2, \beta_2)\}$ при некоторых $\beta_i \le D'\lambda/2$, $i = 1, 2$. Тогда, в силу целочисленности координат векторов из X и с учетом (14.9), π задает нестрогое суммирование X в $\mathcal{P}_5(2, b)$ при $b = (\lceil D'\lambda/2 \rceil, \lceil D'\lambda/2 \rceil) = (B'/2, B'/2)$. Последнее эквивалентно тому, что π задает нестрогое суммирование чисел Z в отрезке $[-Bt, Bt]$.

Как и в задаче НСЧ1, последовательно соединим частичные суммы $z_\pi^0, \ldots, z_\pi^n$ звеньями ломаной и будем говорить о траектории суммирования. Поскольку каждое γ-звено превосходит длину отрезка $[-Bt, Bt]$ на $\alpha'+\alpha''$ и согласно (14.10) α'-, α''-звенья суть самые длинные звенья, идущие в положительном направлении, для обеспечения нестрогого суммирования в отрезке длины B' необходимо, чтобы каждое γ-звено было инцидентно хотя бы одному α''-звену. (Если $\alpha' = \alpha''$, то нам не важно, какие из них считать α''-звеньями.) А поскольку число γ-звеньев совпадает с числом α''-звеньев, каждое γ-звено инцидентно точно одному α''-звену и точно одному α'-звену. (Снова если $\alpha' = \delta$, то α'- и δ-числа взаимозаменяемы.) Таким образом, α'-, α''- и γ-звенья объединяются в $(m+1)$ троек, промежутки между которыми в траектории суммирования заполняются A- и δ-звеньями. А поскольку каждая тройка $\alpha_i''\gamma_i\alpha_i'$ начинается в точке $Bt = a + B'$ и заканчивается в точке $a \doteq -Bt$, суммарная длина звеньев каждого промежутка в точности равна B'. Покажем, что все δ-звенья содержатся в одном промежутке, которому принадлежат начальный и конечный отрезки замкнутой траектории суммирования.

Действительно, начальный отрезок $[0, B'/2]$ траектории заполнен, как известно, A- и δ-звеньями. Из (14.8) и (14.10) имеем, что суммарная длина любых $(t+1)$ таких звеньев превосходит Bt, а длина любых t из них не превосходит Bt, причем равна Bt, если и только если все звенья являются δ-звеньями. Аналогично, конечный отрезок $[-B'/2, 0]$ траектории тоже заполнен t δ-звеньями, что и доказывает наше утверждение.

Таким образом, все остальные промежутки между тройками заполнены только A-звеньями, что дает искомое разбиение в задаче k-РАЗБИЕНИЕ.

Для завершения доказательства NP-трудности задачи НСВ5 $(2, \lambda)$ достаточно заметить, что при любом фиксированном $\lambda \in (0, 1)$ размер входа задачи НСВ5 $(2, \lambda)$ совпадает по порядку с размером входа задачи k-РАЗБИЕНИЕ. Наконец, NP-полнота задачи следует из ее очевидной принадлежности классу NP.

Доказательство NP-полноты задачи НСВ6 (λ) в значительной степени аналогично приведенному выше. При этом различия в доказательствах обусловлены, главным образом, различием целевых функций задач НСВ5 и НСВ6. Приведем краткую схему доказательства.

Семейство чисел $Z' = \{z_1', \ldots, z_{n'}'\}$ для задачи НСВ6 получается из Z удалением всех δ-чисел и по одному α'-, α''-, и γ-числу.

Далее определим семейство векторов $X' = \{x_i' = (z_i', 0) \,|\, i \in \mathbb{N}_{n'}\}$, которое, как нетрудно видеть, удовлетворяет условию задачи НСВ6(λ) при $D = D'$. Покажем, что положительный ответ на вопрос в задаче НСВ6(λ) имеет место, если и только если в исходной задаче k-РАЗБИЕНИЕ существует такое разбиение $\{A_1, \ldots, A_m\}$ множества A, что

$$\sum_{i \in A_j} s(i) = B, \;\; j \in \mathbb{N}_m.$$

Если в задаче k-РАЗБИЕНИЕ при заданном входе ответ положителен, т.е. искомое k-разбиение $\{A_1, \ldots, A_m\}$ существует, то в качестве перестановки π^6 номеров $\{1, 2, \ldots, n'\}$ возьмем

$$\pi^6 = (\alpha_1''\gamma_1\alpha_1'A_1 \cdots \alpha_m''\gamma_m\alpha_m'A_m).$$

Убеждаемся, что π^6 задает нестрогое суммирование чисел из Z' в отрезке $[-B', 0]$, и тем самым, — нестрогое суммирование векторов из X' в семействе областей $\mathcal{P}_6(b)$ при $b = (0, 0, B', 0)$, что дает оценку $\theta_6(b) = B' \le \lambda\theta_6^* D'$. Докажем обратную импликацию.

Пусть некоторая перестановка π задает нестрогое суммирование векторов из X' в семействе областей $\mathcal{P}_6(2, b)$ с вектором $b = (\beta_1, \ldots, \beta_4)$, дающим оценку $\theta_6(b) \le \lambda\theta_6^* D'$. Так как из $x_\pi^k \notin P(e_2 - e_1, \beta_3)$ следует $x_\pi^k(1) < -\beta_3$, то в силу целочисленности координат векторов из X' все числа β_i можно считать целыми. (Их можно заменить на $\lfloor\beta_i\rfloor$, $i = 1, \ldots, 4$.) Таким образом, перестановка π задает нестрогое суммирование чисел из Z' в интервале $[-\beta_3, \beta_1]$, длина которого не превосходит $\beta_1 + \beta_3 \le \theta_6(b) \le \lfloor\lambda\theta_6^* D'\rfloor = B'$. Можем считать, что нестрогое суммирование чисел $\{z_i'\}$ происходит в некотором большем отрезке $[a, a + B']$. Проведя аналогичный анализ свойств траектории суммирования, приходим к выводу, что траектория содержит m троек из $\alpha'-, \alpha''-$ и γ-звеньев, а m промежутков между тройками (каждый длины B') заполнены только A-звеньями, что и дает искомое разбиение $\{A_1, \ldots, A_m\}$ для задачи k-РАЗБИЕНИЕ.

Теорема 14.14 доказана.

15 Суммирование векторов в специальных областях пространства

В этом разделе будут рассмотрены задача о суммировании l_∞-семейства векторов в прямоугольной области m-мерного пространства, ограниченной единицей по одной координате и константами — по другим координатам[57], и задача о суммировании 0-семейства векторов внутри минимального угла на плоскости и внутри прямоугольного октанта — в 3-мерном пространстве[58].

1. Суммирование векторов в минимальном параллелепипеде

Через $[i, j]$ будем обозначать интервал целых чисел $\{i, i + 1, \ldots, j\}$.

Из следствия 11.8 для нормы l_∞ вытекает существование констант $C_1, \ldots, C_m$, удовлетворяющих следующему свойству.

Sb-свойство. *Для любого l_∞-семейства векторов $\{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ существует перестановка π, такая что*

$$\left|\sum_{j \in \mathbb{N}_k} x_{\pi_j}(\nu)\right| \le C_\nu, \ \forall\ k \in \mathbb{N}_n, \ \nu \in \mathbb{N}_m. \tag{15.1}$$

Вектор $\bar{C} = (C_1, \ldots, C_m)$ будем называть *границей Штейница*, если его компоненты $\{C_\nu\}$ удовлетворяют Sb-свойству. Через $Sb(m)$ обозначим множество всех границ Штейница в $\mathbb{R}^m$. Естественно, что мы заинтересованы в нахождении **минимальных** границ

[57] результат из совместной статьи с В. Банащиком [122]

[58] результат из совместной статьи с С.В. Августиновичем [69]

Штейница, т.е. таких векторов $x \in Sb(m)$, для которых не существует $y \in Sb(m),\ y < x$ ($y < x$ означает $y(i) \le x(i),\ \forall\ i \in \mathbb{N}_m$, где хотя бы одно из неравенств строгое).

Согласно следствию 11.8, имеем $\left(m - 1 + \frac{1}{m}, \ldots, m - 1 + \frac{1}{m}\right) \in Sb(m)$. Барань и Гринберг [75] поставили вопрос: существует ли граница Штейница $(C_1, \ldots, C_m)$ с $C_1 = 1$? Они получили утвердительный ответ для случая $m = 2$, доказав, что для любого $m \ge 2$ выполняется $(m/2, 8m+2, \ldots, 8m+2) \in Sb(m)$, откуда следует $(1, 18) \in Sb(2)$. Последний результат был улучшен автором диссертации в [46], где было показано, что $(1, 3) \in Sb(2)$. Окончательный неулучшаемый результат для случая $m = 2$ был получен Банащиком [73], который показал, что $(a, b) \in Sb(2)$, если и только если $a + b \ge 3$ и $a, b \ge 1$. (Таким образом, $\{(a, b) \in \mathbb{R}^2 \,|\, a + b = 3,\ a \ge 1,\ b \ge 1\}$ есть множество **всех** минимальных границ Штейница в $\mathbb{R}^2$.) В частности, $(1, 2) \in Sb(2)$.

В этом разделе будет доказан утвердительный ответ на вопрос Бараня–Гринберга в случае произвольного m.

Теорема 15.1 *Пусть $\mathcal{A}$ — алгоритм приближенного решения задачи КСВ с временной сложностью $T_{\mathcal{A}}(n, m)$ и оценкой $f_{l_\infty, X}(\pi) \le C_{\mathcal{A}}(m)$, гарантированной для любого l_∞-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$. Тогда для любого $m \ge 2$ справедливо:*

$$\bar{C}^1 \doteq ((C_{\mathcal{A}}(m) + 2)/3, 3C_{\mathcal{A}}(m), \ldots, 3C_{\mathcal{A}}(m)) \in Sb(m);$$

$$\bar{C}^2 \doteq (1, \psi(m), \ldots, \psi(m)) \in Sb(m),$$

где $\psi(m) = 9C_{\mathcal{A}}(m) \cdot \uparrow\!\frac{1}{2}(C_{\mathcal{A}}(m) - 1)\!\uparrow_3$; $\uparrow x \uparrow_k$ есть наименьшее число вида k^l, не меньшее x, где l — целое (не обязательно положительное).

Для любого l_∞-семейства векторов $\{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ перестановки π_1 и π_2, удовлетворяющие (15.1) с границами $\bar{C}^1$ и $\bar{C}^2 \in Sb(m)$, находятся алгоритмами $\mathcal{A}_1$ и $\mathcal{A}_2$ временной сложности $O(T_{\mathcal{A}}(n, m))$ и $O(T_{\mathcal{A}}(n, m) + n \log C_{\mathcal{A}}(m))$ соответственно.

Применяя в теореме 15.1 алгоритм $\mathcal{A}$ из следствия 11.8 для нормы l_∞ и используя соотношения

$$\uparrow\!\frac{1}{2}(m - 2 + \tfrac{1}{m})\!\uparrow_3 = \uparrow\!\frac{1}{2}(m-1)\!\uparrow_3 \le \tfrac{3}{2}m - 3$$

для $m \ge 3$, получаем следующее

Следствие 15.2
(1) $(1, 21, 21) \in Sb(3)$, $\quad \bar{C}^3 \doteq (1, \varsigma(m), \ldots, \varsigma(m)) \in Sb(m)$, *где*

$$\varsigma(m) = 9\left(m - 1 + \tfrac{1}{m}\right) \uparrow^{\frac{m-1}{2}}\!\uparrow_3 \le 13.5\left(m^2 - 3m + 3 - \tfrac{2}{m}\right).$$

(2) *Для любого l_∞-семейства векторов $\{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ перестановка π, удовлетворяющая (15.1) с границей $\bar{C}^3 \in Sb(m)$, находится алгоритмом трудоемкости $O(n^2 m^2)$.*

Далее перейдем к описанию алгоритмов $\mathcal{A}_1$ и $\mathcal{A}_2$ и доказательству теоремы 15.1.

Алгоритмы

Поскольку перестановка π_1 представляет собой промежуточный результат работы алгоритма $\mathcal{A}_2$, достаточно описать лишь алгоритм $\mathcal{A}_2$. Последний состоит из двух этапов, на первом из которых находится перестановка π, удовлетворяющая оценке $f_{l_\infty,X}(\pi) \le C_{\mathcal{A}}(m)$, в то время как второй этап состоит из не более чем $O(\log C_{\mathcal{A}}(m))$ итераций. Перестановка π_1 получается на выходе первой такой итерации.

Каждая итерация преобразует перестановку, полученную на предыдущей итерации (или на первом этапе — для первой итерации). Будут использоваться преобразования двух типов: инверсии и скачки. Инверсии могут быть использованы на любой итерации, тогда как скачки — только на последней итерации (в случае необходимости).

Инверсия есть замена какого-то отрезка перестановки на обратный:

$$\pi = (\ldots, \pi_{i-1}, \pi_i, \ldots, \pi_j, \pi_{j+1}, \ldots) \to (\ldots, \pi_{i-1}, \pi_j, \pi_{j-1}, \ldots, \pi_i, \pi_{j+1}, \ldots) \doteq \sigma_{ij}(\pi).$$

Скачок есть перемещение какого-то элемента перестановки вперед (в случае *скачка вперед*) или назад (в случае *скачка назад*):

$$\pi = (\ldots, \pi_{i-1}, \pi_i, \pi_{i+1}, \ldots, \pi_j, \pi_{j+1}, \ldots) \to (\ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_j, \pi_i, \pi_{j+1}, \ldots) \doteq \gamma_{ij}^{+}(\pi)$$

(скачок вперед с i-го на j-е место в перестановке),

$$\pi = (\ldots, \pi_{i-1}, \pi_i, \ldots, \pi_{j-1}, \pi_j, \pi_{j+1}, \ldots) \to (\ldots, \pi_{i-1}, \pi_j, \pi_i, \ldots, \pi_{j-1}, \pi_{j+1}, \ldots) \doteq \gamma_{ij}^{-}(\pi)$$

(скачок назад с j-го на i-е место в перестановке).

Пусть перестановка π удовлетворяет (15.1) с $C_\nu = C'_\nu,\ \nu \in \mathbb{N}_m$. Будем выполнять "обычные" итерации, пока выполняется $C'_1 > \frac{5}{3}$. В противном случае выполняем "последнюю" итерацию. Вначале опишем "обычную" итерацию.

Положим $D \doteq (C'_1 + 2)/3$ и последовательно просмотрим список частичных сумм $\{y_k \doteq \sum_{i\in\mathbb{N}_k} x_{\pi_i} \,|\, k = 1, 2, \ldots, n\}$. Если для каких-то i, j $(i < j)$ обнаруживается

$$|y_k(1)| > D \ \forall\ k \in [i, j-1]; \quad |y_{i-1}(1)| \le D; \quad |y_j(1)| \le D$$

(такой интервал $[i, j]$ будем называть "плохим"), то инвертируем перестановку π на интервале $[i, j]$, т.е. $\pi := \sigma_{ij}(\pi)$. Итерация заканчивается, когда перестановка π полностью просмотрена и все ее плохие интервалы инвертированы.

Пусть $\bar\pi = \sigma_{ij}(\pi);\ y_k = \sum_{i\in\mathbb{N}_k} x_{\pi_i};\ \bar y_k = \sum_{i\in\mathbb{N}_k} x_{\bar\pi_i}$. Ясно, что $\bar y_k = y_k$ для всех $k < i$ и $k \ge j$. Для $k \in [i, j-1]$ векторы $\{\bar y_k\}$ являются отражениями векторов $\{y_k \,|\, k \in [i, j-1]\}$ относительно точки $z \doteq (y_{i-1} + y_j)/2$, т.е.

$$\bar y_k = 2z - y_{j+i-1-k},\ k \in [i, j-1]. \tag{15.2}$$

Если $|y_k(1)| > D$ для $k \in [i, j-1]$, то из неравенств $D - 1 < z(1) \le D,\ y_k(1) \le C'_1$ и (15.2) следует

$$D > \bar y_k(1) = 2z(1) - y_{j+i-1-k}(1) > 2D - 2 - C'_1 = -D.$$

В случае $|y_k(1)| < -D,\ \forall\ k \in [i, j-1]$ включение $\bar{y}_k(1) \in (-D, D)$ доказывается аналогично. Для остальных координат $\nu = 2, \ldots, m$ имеем:

$$|\bar{y}_k(\nu)| = |2z(\nu) - y_{j+i-1-k}(\nu)| \le |y_{i-1}(\nu)| + |y_j(\nu)| + |y_{j+i-1-k}(\nu)| \le 3C'_\nu,\ \forall\ k \in [i, j-1].$$

Таким образом, если перестановка удовлетворяет (15.1) с $C_\nu = C'_\nu$ перед началом выполнения "обычной" итерации, то она удовлетворяет (15.1) с

$$C_1 \le (C'_1 + 2)/3, \tag{15.3}$$

$$C_\nu \le 3C'_\nu,\ \nu = 2, \ldots, m \tag{15.4}$$

по окончании этой итерации.

"Последняя" итерация отличается от "обычных" и приводит к требуемому неравенству $C_1 \le 1$. Алгоритм $\mathcal{A}_2$ начинает последнюю итерацию как только перестановка π удовлетворяет (15.1) с $C_1 \le \frac{5}{3}$. В этом случае полагаем $D := 1$. Далее просматриваем перестановку π и находим все ее "плохие" интервалы $[i, j]$. Как и раньше, определяем точку $z \doteq (y_{i-1} + y_j)/2$ для "плохого" интервала $[i, j]$. Если $|z(1)| \ge \frac{1}{3}$, то инвертируем перестановку π на интервале $[i, j]$: $\pi := \bar{\pi} \doteq \sigma_{ij}(\pi)$, получая при этом $|\bar{y}_k(1)| < 1,\ \forall\ k \in [i, j-1]$. Если же $|z(1)| < \frac{1}{3}$, то выполняется по крайней мере одно из следующих двух неравенств: $|y_{i-1}(1)| < \frac{1}{3}$ либо $|y_j(1)| < \frac{1}{3}$. В первом случае к перестановке π применяется скачок вперед $\gamma^+_{ij}(\pi)$, а во втором случае — скачок назад $\gamma^-_{ij}(\pi)$. Доказательство того, что новая перестановка удовлетворяет соотношениям $|y_k(1)| < 1,\ \forall\ k \in [i, j]$, оставляем читателю. Нетрудно также заметить, что каждый скачок может увеличить интервал значений других координат ($\{y_k(\nu) \,|\, k \in \mathbb{N}_n\}$) не более чем на 1 для каждой координаты $\nu = 2, \ldots, m$. Таким образом, для последней итерации также верно соотношение (15.4).

Это завершает описание алгоритма $\mathcal{A}_2$ и анализ его поведения на итерациях.

Завершение доказательства теоремы 15.1

Для завершения доказательства обеих частей теоремы нам остается оценить число итераций алгоритма. Из (15.3) следует, что достаточно выполнить k "обычных" итераций, где k — такое наименьшее целое, что

$$\frac{5}{3} \ge \frac{1}{3^k} C_{\mathcal{A}}(m) + \frac{2}{3}\left(1 + \frac{1}{3} + \cdots + \frac{1}{3^{k-1}}\right) = \frac{1}{3^k} C_{\mathcal{A}}(m) + 1 - \frac{1}{3^k}.$$

Отсюда

$$k = \left\lceil \log_3 \frac{1}{2}(C_{\mathcal{A}}(m) - 1) \right\rceil + 1, \tag{15.5}$$

и число итераций оценивается величиной $O(\log C_{\mathcal{A}}(m))$. Наконец учитывая, что каждая итерация имеет трудоемкость $O(n)$, получаем оценки трудоемкостей алгоритмов $\mathcal{A}_1$ и $\mathcal{A}_2$, декларируемые в теореме.

С учетом (15.4) и (15.5) получаем окончательную оценку величин $\{C_\nu \,|\, \nu > 1\}$:

$$C_\nu \le C_{\mathcal{A}}(m) 3^{k+1} = 9 C_{\mathcal{A}}(m) \left\uparrow^{\frac{C_{\mathcal{A}}(m)-1}{2}}\right\uparrow_3,$$

что завершает доказательство теоремы 15.1. □

2. Суммирование векторов внутри минимального угла

В разделе 14 мы уже сталкивались с задачами, в которых требовалось для заданного 0-семейства векторов найти порядок суммирования этих векторов, при котором вся траектория суммирования лежит внутри некоторого оптимального "угла", т.е. удаляющейся в бесконечность области, являющейся пересечением конечного числа полупространств в $\mathbb{R}^m$. К числу таких задач относятся задачи НСВ2 и НСВ5. При этом направление всех ребер угла является заданным, а его оптимальность оценивается по степени близости его вершины к началу координат (согласно выбранной целевой функции). Близкой к этим задачам нам представляется задача о суммировании векторов внутри минимального угла, рассматриваемая в настоящем разделе.

Нас интересует следующий вопрос: внутри какого минимального угла (с вершиной в начале координат) можно просуммировать всякое 0-семейство векторов в $\mathbb{R}^m$, если направление угла не ограничено? Для плоскости мы показываем, что минимальным возможным углом является $\pi/3$. Для пространств большей размерности возможны различные варианты постановки задачи — в зависимости от выбора способа измерения величины угла. (Например, величи́ны трехгранных углов в трехмерном пространстве можно оценивать по значению максимума либо суммы величин его двугранных углов.) Мы показываем, что в трехмерном пространстве всякое 0-семейство векторов может быть просуммировано внутри прямоугольного октанта.

Суммирование векторов на плоскости

Пусть задано 0-семейство векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$. Угол $\alpha = \alpha(X)$ называется *углом суммирования* семейства X, если существует перестановка p векторов из X, для которой траектория суммирования векторов целиком помещается внутри угла размера α с вершиной в начале координат, и ни для какого меньшего угла такой перестановки не существует.

Теорема 15.3 *Для любого 0-семейства векторов $X \subset \mathbb{R}^2$ выполнено*

$$\alpha(X) \le \pi/3, \tag{15.6}$$

и эта оценка является наилучшей возможной.

Доказательству теоремы предшествуют несколько простых утверждений.

Утверждение 15.4 *Пусть в $\mathbb{R}^m$ задано 0-семейство векторов и некоторая траектория их суммирования. Тогда замена порядка суммирования векторов на любом сегменте траектории на обратный порядок эквивалентна отражению этого сегмента относительно средней точки отрезка, соединяющего концы сегмента. В частности, замена всей перестановки на обратную приводит к траектории, центрально симметричной с исходной.*

Лемма 15.5 *Для любого 0-семейства векторов $X \subset \mathbb{R}^2$ существует траектория их суммирования, выпуклая оболочка которой является треугольником.*

Доказательство. Пусть задано произвольное 0-семейство векторов $X \subset \mathbb{R}^2$. Занумеруем векторы семейства по часовой стрелке, начиная с произвольного вектора. Полученная траектория суммирования образует выпуклый многоугольник M. Впишем в M треугольник T максимальной площади. Это можно сделать таким образом, что вершины A, B, C треугольника T будут совпадать с некоторыми вершинами многоугольника M. Через каждую вершину треугольника T проведем прямую линию, параллельную противоположной стороне треугольника. Пересечение этих прямых образует треугольник $A'B'C'$, подобный треугольнику ABC и содержащий многоугольник M (поскольку площадь вписанного треугольника T максимальна, см. рис. 15.1).

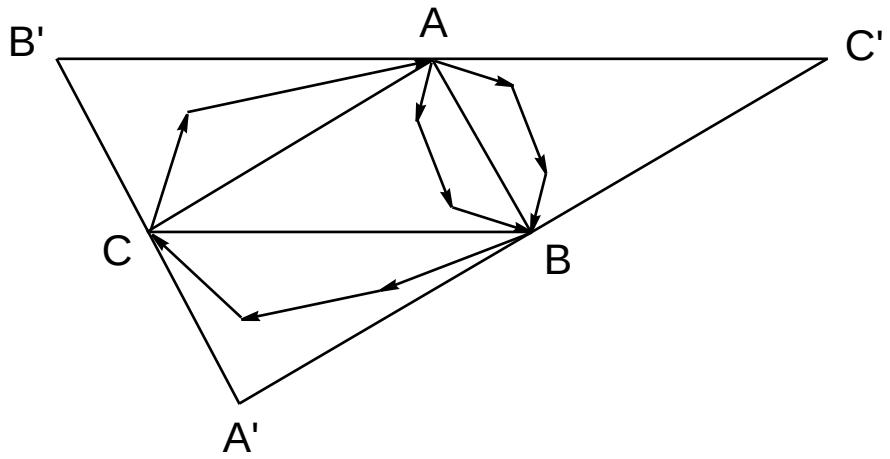


Рис. 15.1: Треугольник $T = ABC$, вписанный в многоугольник M.

Рассмотрим сегмент траектории, соединяющий точки A и B. Он целиком содержится в треугольнике ABC', подобном треугольнику ABC. Заменим порядок суммирования в этой части траектории на обратный. Согласно утверждению 15.4 новое положение этой части будет целиком внутри треугольника ABC (после отражения относительно середины отрезка AB). Выполняя такое же преобразование для остальных двух частей траектории, получим траекторию, чьей выпуклой оболочкой является треугольник ABC. □

Из леммы 15.5 и утверждения 15.4 вытекает следующее

Следствие 15.6 *Для любого 0-семейства векторов $X \subset \mathbb{R}^2$ существуют три прямых, проходящих через начало координат, таких что векторы из X могут быть просуммированы внутри любого из 6 углов, образованных этими прямыми.*

Доказательство теоремы 15.3. Оценка (15.6) непосредственно вытекает из следствия 15.6. Она достигается на семействе из трех векторов равной длины, в котором угол между любыми двумя векторами составляет $2\pi/3$. □

Суммирование векторов внутри 3-мерного октанта

Квадрантом называется произвольный прямой угол с вершиной в начале координат. Трехгранный угол, образованный тремя квадрантами, называется *октантом*.

Лемма 15.7 *Пусть в плоскости $\mathbb{R}^2$ через начало координат проходят три красных и три синих линии. Тогда существует квадрант, содержащий два красных и два синих луча.*

Доказательство. Среди 6 углов, которые образуют три красных линии, найдется два нетупых угла, имеющих общую сторону. Пусть эта сторона является лучом красной линии X, и пусть Y — линия, перпендикулярная к X и проходящая через начало координат. Эти две линии, X и Y, определяют 4 квадранта, каждый из которых содержит два красных луча. Очевидно, один из этих квадрантов содержит также два синих луча. □

Теорема 15.8 *Для любого 0-семейства векторов $X \subset \mathbb{R}^3$ существует траектория их суммирования, целиком помещающаяся в некотором октанте.*

Доказательство. Найдем подмножество индексов $I = \{i_1, \ldots, i_k\} \subset \mathbb{N}_n$, для которого частичная сумма $x^I = \sum_{j \in I} x_j$ имеет максимальную длину, и обозначим $X_I = \{x_j \,|\, j \in I\}$. Ясно, что для $J = \mathbb{N}_n \setminus I$ имеем $x^J = -x^I$. Кроме того, ортогональная проекция каждого вектора x_j на вектор x^I положительна для $j \in I$ и отрицательна для $j \in J$. Рассмотрим проекции $\{x_j^* \,|\, j \in \mathbb{N}_n\}$ векторов из X на плоскость P, перпендикулярную вектору x^I. Ясно, что семейства векторов $X_I^* = \{x_j^* \,|\, j \in I\},\ X_J^* = \{x_j^* \,|\, j \in J\}$ являются 0-семействами векторов на плоскости P. Следовательно, существуют три прямых для семейства X_I^* (будем называть их "синими") и три красных линии для семейства X_J^*, гарантированные следствием 15.6. Согласно лемме 15.7, в плоскости P найдется квадрант γ, содержащий некоторый "синий" угол α (расположенный между двумя "синими" лучами) и некоторый "красный" угол β. Согласно следствию 15.6, найдем перестановку p_I векторов из X_I, которая обеспечивает суммирование их проекций из X_I^* внутри угла α, и следовательно, — внутри квадранта γ. Аналогично находим перестановку p_J с таким же свойством для векторов из X_J. Искомый октант ограничен квадрантом γ и двумя плоскостями, перпендикулярными к P и содержащими стороны квадранта. Очевидно, что траектория суммирования согласно перестановке $p_I p_J$ содержится целиком в этом октанте. □

16 Заключение (открытые вопросы и гипотезы)

1. В связи с результатом леммы 15.5 (стр. 126) возникает следующий вопрос: существует ли такая константа C, что для любого 0-семейства векторов в $\mathbb{R}^3$ существует траектория их суммирования, выпуклая оболочка которой имеет не более C вершин?

2. Доказать, что $\inf_s \varphi_s(m)$ (где $\varphi_s(m)$ — функция Штейница в пространстве $\mathbb{R}^m$ с нормой s) неограниченно растет с ростом m.

Более сильной является следующая открытая проблема: доказать, что $\inf_s \varphi_s(m)$ достигается на евклидовой норме.

3. Верно ли, что всякое l_2-семейство векторов в $\mathbb{R}^m$ может быть просуммировано в шаре радиуса $O(\sqrt{m})$?

Часть II

Экстремальные комбинаторные задачи

В этой главе будут рассмотрены комбинаторные задачи различной природы. Будет показано, что несмотря на их внешнюю несхожесть, они имеют тесные связи друг с другом, которые позволяют, в частности, строить быстрые алгоритмы приближенного решения одной задачи, используя полиномиальные алгоритмы приближенного решения другой, либо оценивать экстремальные значения целевой функции одной задачи с помощью аналогичных характеристик другой задачи из рассматриваемого семейства. Будут рассмотрены: задача о равномерном распределении годового плана предприятия по кварталам (месяцам, и т.п.); о нахождении в сети максимального потока, равномерного по стокам; о равномерном распределении "камней" по "кучам" с ограничениями на выбор кучи; о нахождении вершины параллелепипеда, ближайшей к заданной внутренней точке; задача "эквидистанции"; задача об "округлении", и другие.

17 Ранги целых чисел и их свойства[59]

Пусть задано целое число m; $k = \lceil \log_2 m \rceil$. Для любого числа $x \in \overline{\mathbb{N}}_m$ определим его *ранг* $r_m(x)$: $r_m(m) = r_m(0) = k$; $r_m(x) = \max\{\nu \in \overline{\mathbb{N}}_{k-1} \,|\, x$ делится на $2^\nu\}$ — для $x \in \mathbb{N}_{m-1}$.

Для любых $x \in \mathbb{N}_{m-1}$, $\nu \in \{r_m(x)+1, \ldots, k\}$ через $j_m^+(\nu, x)$ и $j_m^-(\nu, x)$ обозначим ближайшие к x сверху и снизу числа x', x'', имеющие ранги $r_m(x'), r_m(x'') \geq \nu$ (будем называть их *предками* числа x). Ближайших предков числа x будем называть его *родителями* и обозначать $j_m^+(x) = j_m^+(r_m(x)+1, x), j_m^-(x) = j_m^-(r_m(x)+1, x)$. Сформулируем несколько простых свойств чисел $j_m^+(\nu, x)$, $j_m^-(\nu, x)$.

Утверждение 17.1 $j_m^+(x) = \min\{m, x + 2^{r_m(x)}\}$, $j_m^-(x) = x - 2^{r_m(x)}$, $\forall\, x \in \mathbb{N}_{m-1}$. □

Утверждение 17.2 *Если* $r_m(x) \leq k-2$, *то* $\forall\, \nu \in \{r_m(x)+1, \ldots, k-1\}$ *имеет место* $r_m(j_m^+(\nu, x)) \neq r_m(j_m^-(\nu, x))$ *и* $\min\{r_m(j_m^+(\nu, x)), r_m(j_m^-(\nu, x))\} = \nu$. □

Утверждение 17.3 *Если* $r_m(j_m^+(\nu, x)) > r_m(j_m^-(\nu, x))$, *то* $j_m^+(\nu, x) = j_m^+(j_m^-(\nu, x))$; *в противном случае* $j_m^-(\nu, x) = j_m^-(j_m^+(\nu, x))$. □

Утверждение 17.4 *Если* x_1, x_2, — *два соседних числа ранга* r $(r < k)$ *и* $x_1 < x_2$, *то* $j_m^+(x_1) = x_1 + 2^r = j_m^-(x_2) = x_2 - 2^r$. □

Определим функцию $\delta_m(x)$ для $x \in \overline{\mathbb{N}}_m$ рекурсией по убыванию $r_m(x)$:

$$\delta_m(x) = \begin{cases} 0, & \text{для } x \in \{0, m\}; \\ 1 + \lambda_m(x)\delta_m(j_m^+(x)) + (1 - \lambda_m(x))\delta_m(j_m^-(x)), & \text{для } x \in \mathbb{N}_{m-1}, \end{cases} \tag{17.1}$$

где

$$\lambda_m(x) = \frac{x - j_m^-(x)}{j_m^+(x) - j_m^-(x)} \in (0, 1). \tag{17.2}$$

[59] Результаты этого раздела используются при построении эффективного алгоритма решения задачи open shop в разделе 26.

Лемма 17.5 *Пусть* $m' = 2^k = 2^{\lceil \log_2 m \rceil}$. *Тогда*

$$\delta_m(x) \le \delta_{m'}(x),\ \forall\ x \in \overline{\mathbb{N}}_m. \tag{17.3}$$

Доказательство проводится индукцией по убыванию $r_m(x)$.

Для $x \in \{0, m\}$ (17.3) справедливо. Для $y \in \mathbb{N}_{m-1}$ из определения m' и утверждения 17.1 имеем $r_{m'}(y) = r_m(y) \in \overline{\mathbb{N}}_{k-1}$, $j^+_{m'}(y) = y + 2^{r_m(y)}$, $\lambda_{m'}(y) = 1/2$. Пусть (17.3) выполняется для $\{x \,|\, r_m(x) > r_m(y)\}$. Тогда в случае $j^+_m(y) = m$ из (17.1), (17.2) имеем:

$$\begin{aligned}
\delta_m(y) &= 1 + \frac{y - j^-_m(y)}{j^+_m(y) - j^-_m(y)}\delta_m(j^+_m(y)) + \frac{j^+_m(y) - y}{j^+_m(y) - j^-_m(y)}\delta_m(j^-_m(y)) \\
&= 1 + \frac{m - y}{m - j^-_m(y)}\delta_m(j^-_m(y)) \le 1 + \frac{1}{2}\delta_{m'}(j^-_m(y)) \\
&\le 1 + \frac{1}{2}\delta_{m'}(j^+_{m'}(y)) + \frac{1}{2}\delta_{m'}(j^-_{m'}(y)) = \delta_{m'}(y).
\end{aligned}$$

В случае $j^+_m(y) = y + 2^{r_m(y)}$ имеем:

$$\begin{aligned}
\delta_m(y) &= 1 + \frac{1}{2}\delta_m(j^+_m(y)) + \frac{1}{2}\delta_m(j^-_m(y)) \\
&\le 1 + \frac{1}{2}\delta_{m'}(j^+_{m'}(y)) + \frac{1}{2}\delta_{m'}(j^-_{m'}(y)) = \delta_{m'}(y).
\end{aligned}$$

Лемма 17.5 доказана.

Определим функции $\psi^+_m(x) = 3\delta_m(x) + \delta_m(x+1),\ x \in \overline{\mathbb{N}}_{m-1}$, $\psi^-_m(x) = 3\delta_m(x) + \delta_m(x-1),\ x \in \mathbb{N}_m$, и величины: $\psi^+_m = \max_x \psi^+_m(x)$, $\psi^-_m = \max_x \psi^-_m(x)$, $\psi_m = \max\{\psi^+_m, \psi^-_m\}$.

Из леммы 17.5 следует, что $\psi^+_m \le \psi^+_{m'}$, $\psi^-_m \le \psi^-_{m'}$, $\psi_m \le \psi_{m'}$. Поскольку в дальнейшем наша задача будет состоять в нахождении верхней оценки величины ψ_m, то это достаточно сделать для величины $\psi_{m'}$, поэтому далее будем считать (если не оговорено противное), что $m = m' = 2^k$. При этом индекс m в функциях $r_m(x), j^+_m(x), \delta_m(x)$ и других будем опускать. Из утверждения 17.1 и формулы (17.2) будем иметь $\lambda(x) \equiv \frac{1}{2},\ \forall\ x \in \mathbb{N}_{m-1}$.

Лемма 17.6

$$\delta(x) \in (\max\{\delta(j^+(x)), \delta(j^-(x))\},\ \min\{\delta(j^+(x)), \delta(j^-(x))\} + 2),\ \forall\ x \in \mathbb{N}_{m-1}. \tag{17.4}$$

Доказательство проводится индукцией по убыванию $r(x)$.

Если $r(y) = k - 1$, то $\delta(y) = 1$, в то время как $\delta(j^+(y)) = \delta(m) = \delta(j^-(y)) = \delta(0) = 0$. Пусть $r(y) \le k - 2$, а для $\{x \,|\, r(x) > r(y)\}$ соотношение (17.4) выполнено. В силу утверждений 17.2 и 17.3 одно из чисел $j^+(y), j^-(y)$ является "родителем" другого, и для них, по индукционному предположению, (17.4) выполнено. Пусть, например, $j^+(y) = j^+(j^-(y))$. Тогда $\delta(j^-(y)) - \delta(j^+(y)) \in (0, 2)$ и

$$\delta(y) = 1 + \frac{1}{2}\delta(j^+_m(y)) + \frac{1}{2}\delta(j^-_m(y)) \in (\delta(j^-_m(y)), \delta(j^+_m(y)) + 2).$$

Аналогичные соотношения получаются в случае $j^-(y) = j^-(j^+(y))$. Лемма 17.6 доказана.

Если x — четное число ($0 < x < m$), то числа $x-1$ и $x+1$ нечетны (а значит, имеют ранг 0), число x является их "родителем", и по лемме 17.6, $\delta(x) < \delta(x-1)$ и $\delta(x) < \delta(x+1)$. Отсюда $\psi^+(x) = 3\delta(x) + \delta(x+1) < \delta(x) + 3\delta(x+1) = \psi^-(x+1)$, и $\psi^-(x) = 3\delta(x) + \delta(x-1) < \delta(x) + 3\delta(x-1) = \psi^+(x-1)$. Кроме того, для $x = 0$ имеем $\psi^+(0) = \delta(1) < 3\delta(1) = \psi^-(1)$, а для $x = m$, аналогично, $\psi^-(m) = \delta(m-1) < 3\delta(m-1) = \psi^+(m-1)$. Таким образом, при вычислении величины ψ_m достаточно ограничиться значениями функций $\psi^+(x), \psi^-(x)$ от нечетных значений x, то есть от чисел ранга ноль. Каждое из таких чисел x в двоичной записи представимо в виде

$$x = \underbrace{0\ldots0}_{p_s}\underbrace{1\ldots1}_{p_{s-1}}0\ldots1\underbrace{0\ldots0}_{p_2}\underbrace{1\ldots1}_{p_1}, \tag{17.5}$$

где p_i — количество идущих подряд нулей или единиц, $\sum_{i=1}^{s} p_i = k,\ p_1 \geq 1$, то есть нулевой бит равен 1. Обозначим $p^i = \sum_{j=1}^{i} p_j$. Тогда $p^s = k,\ p^0 = 0$. Для любого $r \in \mathbb{N}_k$ определим $i(r) = \max\{i \mid p^i \leq r\},\ \nu(r) = r - p^{i(r)}$.

Далее заметим, что при $m = 2^k$ функция $\delta(x)$ симметрична относительно $m/2$, т.е. $\delta(x) = \delta(m-x),\ \forall\, x$, откуда $\psi^-(x) = \psi^+(m-x),\ \forall\, x$, и значит $\psi_m = \psi^- = \psi^+$. Поэтому далее ограничимся вычислением функции ψ^+, и индекс "+" будем опускать.

Выведем формулу для $\psi(x)$ в терминах величин $\{p_i\}$.

Лемма 17.7 *Для всякого нечетного $x \in \mathbb{N}_{m-1}$ и $t \in \mathbb{N}_k$ выполняется*

$$\psi(x) = \Delta(t) + \begin{cases} \frac{\alpha_i}{2^\nu}\delta(j^+(t,x)) + \left(4 - \frac{\alpha_i}{2^\nu}\right)\delta(j^-(t,x)), & \textit{для нечетного } i, \\ \left(4 - \frac{\alpha_i}{2^\nu}\right)\delta(j^+(t,x)) + \frac{\alpha_i}{2^\nu}\delta(j^-(t,x)), & \textit{для четного } i, \end{cases} \tag{17.6}$$

где

$$\alpha_i = 4 - \alpha_{i-1}2^{-p_i},\ i \in \mathbb{N}_s;\ \alpha_0 = 3, \tag{17.7}$$

$$\Delta(t) = -2 + \sum_{j=1}^{i}(4\alpha_j - 8) + 2\left(4 - \frac{\alpha_i}{2^\nu}\right), \tag{17.8}$$

$i = i(t),\ \nu = \nu(t)$, а величины $\{p_i\}$ определены выражением (17.5).

Доказательство проведем индукцией по возрастанию t.

При $t = 1$ имеем $i(t) = 0,\ \nu(t) = 1$. Так как $x + 1 = j^+(1, x)$, то расписывая $\delta(x)$ по формуле (17.1), получим

$$\psi(x) = 3\delta(x) + \delta(x+1) = 3 + \frac{5}{2}\delta(j^+(1,x)) + \frac{3}{2}\delta(j^-(1,x)).$$

Так как $\frac{\alpha_i}{2^\nu} = \frac{3}{2}$, и по формуле (17.8), $\Delta(1) = -2 + 2(4 - 3/2) = 3$, то (17.6) справедливо при $t = 1$.

Пусть $2 \leq t' \leq k$, и (17.6) справедливо при $t = t' - 1$.

Если $i = i(t'-1)$ нечетно, то $r(j^+(t,x)) = t,\ r(j^-(t,x)) = p^{i+1} > t$, откуда, по утверждению 17.3, $j^-(j^+(t,x)) = j^-(t,x) = j^-(t+1,x)$. Расписывая $\delta(j^+(t,x))$ по формуле (17.1), получим

$$\psi(x) = \Delta' + \frac{\alpha_i}{2^{\nu+1}}\delta(j^+(t+1,x)) + \left(4 - \frac{\alpha_i}{2^{\nu+1}}\right)\delta(j^-(t+1,x)),$$

где

$$\Delta' = \Delta(t) + \frac{\alpha_i}{2^\nu} = -2 + \sum_{j=1}^{i}(4\alpha_j - 8) + 2\left(4 - \frac{\alpha_i}{2^{\nu+1}}\right).$$

Если $\nu + 1 < p_{i+1}$, то $i(t') = i(t'-1),\ \nu(t') = \nu + 1$, и формула (17.6) справедлива при $t = t'$.

Если же $\nu + 1 = p_{i+1}$, то $i(t') = i + 1,\ \nu(t') = 0$. Отсюда с учетом (17.7) получаем, что коэффициент при $\delta(j^-(t+1,x))$ равен $4 - \frac{\alpha_i}{2^{\nu+1}} = 4 - \frac{\alpha_i}{2^{p_{i+1}}} = \alpha_{i+1} = \frac{\alpha_i(t')}{2^{\nu(t')}}$, — как в (17.6) для четного i. Поскольку при этом

$$\begin{aligned}\Delta' &= -2 + \sum_{j=1}^{i}(4\alpha_j - 8) + 2\alpha_{i+1} = -2 + \sum_{j=1}^{i+1}(4\alpha_j - 8) + 2\left(4 - \frac{\alpha_{i+1}}{2^0}\right)\\ &= -2 + \sum_{j=1}^{i(t')}(4\alpha_j - 8) + 2\left(4 - \frac{\alpha_{i(t')}}{2^{\nu(t')}}\right) = \Delta(t'),\end{aligned}$$

то (17.6) и в этом случае справедливо при $t = t'$.

Наконец, если $i = i(t'-1)$ четно, то справедливы соотношения, симметричные тем, которые выполнялись при нечетном i: $r(j^-(t,x)) = t,\ r(j^+(t,x)) = p^{i+1} > t$, откуда $j^+(j^-(t,x)) = j^+(t,x) = j^+(t+1,x)$. Расписывая $\delta(j^-(t,x))$ по формуле (17.1), аналогичным образом доказываем (17.6) при четном i. Лемма 17.7 доказана.

При $t = k$ в формуле (17.6) имеем $\delta(j^+(t,x)) = \delta(j^-(t,x)) = 0,\ i(t) = s,\ \nu(t) = 0$, поэтому справедливо следующее

Следствие 17.8

$$\psi(x) = -2 + \sum_{j=1}^{s-1}(4\alpha_j - 8) + 2\alpha_s, \tag{17.9}$$

где величины α_i определяются согласно (17.7). □

Лемма 17.9 $\max_x \psi(x) = \psi(x_{\max})$, *где* $x_{\max} = \ldots 10101$, *т.е.* $p_i \equiv 1,\ \forall\ i$.

Доказательство. Пусть задано число x, и для некоторого $t \in \mathbb{N}_s$ имеем $p_i = 1,\ \forall\ t+1,\ldots,s;\ p_t \geq 2$. В последних $k - p^t + 1$ битах числа x произведем замену $0 \leftrightarrow 1$, и полученное число обозначим x'. Докажем, что $\psi(x') > \psi(x)$, откуда с использованием индукции вытекает утверждение леммы. Через $\bar{\psi}$ обозначим "хвост" формулы

(17.9), начинающийся с t-го слагаемого, т.е. сумму слагаемых в формуле (17.9), которые различны для x и x'. Имеем:

$$\bar{\psi}(x) = \sum_{i=t}^{s-1}(4\alpha_i - 8) + 2\alpha_s,\ \bar{\psi}(x') = \sum_{i=t}^{s}(4\alpha'_i - 8) + 2\alpha'_{s+1},$$

где $\alpha'_t = 4 - \alpha_{t-1}2^{-p_t+1} = 2\alpha_t - 4;\ \alpha'_i = 4 - \frac{\alpha'_{i-1}}{2},\ i = t+1, \ldots, s+1$. Если по формуле (17.7) выразить величины $\{\alpha_i \,|\, i = \nu+1, \ldots, s\}$ через α_ν (для некоторого $\nu \geq t$), то получим

$$\bar{\psi}(x) = \sum_{i=t}^{\nu-1}(4\alpha_i - 8) + c_\nu\alpha_\nu + b_\nu = c_t\alpha_t + b_t, \tag{17.10}$$

где, очевидно,

$$c_s = 2,\ b_s = 0;\ c_\nu = 4 - \frac{c_{\nu+1}}{2},\ b_\nu = 4c_{\nu+1} + b_{\nu+1} - 8,\ (\nu < s). \tag{17.11}$$

Нетрудно видеть, что если в формуле для $\bar{\psi}(x')$ выразить величины $\{\alpha'_i \,|\, i = \nu+2, \ldots, s+1\}$ через $\alpha'_{\nu+1}$, то получим выражение

$$\bar{\psi}(x') = \sum_{i=t}^{\nu}(4\alpha'_i - 8) + c_\nu\alpha'_{\nu+1} + b_\nu$$

с коэффициентами c_ν, b_ν, определенными согласно (17.11). В частности, при $\nu = t$ имеем

$$\bar{\psi}(x') = (4\alpha'_t - 8) + c_t\alpha'_{t+1} + b_t = 4\alpha'_t - 8 + c_t\left(4 - \frac{\alpha'_t}{2}\right) + b_t = \left(4 - \frac{c_t}{2}\right)\alpha'_t + 4c_t + b_t - 8$$

$$= \left(4 - \frac{c_t}{2}\right)(2\alpha_t - 4) + 4c_t + b_t - 8 = (8 - 2c_t)\alpha_t + c_t\alpha_t + 6c_t + b_t - 24.$$

Заметим, что $c_\nu \in (0, 4)\ \forall\ \nu \geq t$, а также из (17.7) $\alpha_t > 3$, поэтому

$$\bar{\psi}(x') > (8 - 2c_t)3 + 6c_t - 24 + c_t\alpha_t + b_t = c_t\alpha_t + b_t = \bar{\psi}(x).$$

Лемма 17.9 доказана.

Лемма 17.10 $\psi_{m'} = \frac{8}{3}\log_2 m' + \frac{2}{9} + (-1)^{k-1}\frac{2}{9m'}$, *где* $k = \log_2 m'$.

Доказательство. Для вычисления $\psi_{m'}$ воспользуемся формулой $\psi_{m'} = \psi(x_{\max}) = -2 + c_1\alpha_1 + b_1$, где величины c_1, b_1 определены из соотношений (17.11), $s = k,\ \alpha_1 = \frac{5}{2}$. Из (17.11) имеем:

$$c_i = \frac{8}{3} + \frac{4}{3}\left(-\frac{1}{2}\right)^{s-i+1},\ i \in \mathbb{N}_s;$$

$$b_1 = \sum_{i=2}^{s}(4c_i - 8) = \frac{8}{3}(s-1) + \frac{16}{3}\sum_{i=2}^{s}\left(-\frac{1}{2}\right)^{s-i+1} = \frac{8}{3}(s-1) - \frac{16}{3} + \frac{16}{3}\sum_{i=0}^{s-1}\left(-\frac{1}{2}\right)^{i}$$

$$= \frac{8}{3}(s-1) - \frac{16}{3} + \frac{32}{9}\left(1 - \left(-\frac{1}{2}\right)^s\right) = \frac{8}{3}s - \frac{40}{9} - \frac{32}{9}\left(-\frac{1}{2}\right)^s;$$

$$\psi_{m'} = -2 + \frac{5}{2}\cdot\left(\frac{8}{3} + \frac{4}{3}\left(-\frac{1}{2}\right)^s\right) + \frac{8}{3}s - \frac{40}{9} - \frac{32}{9}\left(-\frac{1}{2}\right)^s$$

$$= \frac{8}{3}s + \frac{2}{9} - \frac{2}{9}\left(-\frac{1}{2}\right)^s = \frac{8}{3}k + \frac{2}{9} + (-1)^{k-1}\cdot\frac{2}{9m'}.$$

Лемма 17.10 доказана. □

18 Задачи о равномерных разбиениях

В этом разделе будут исследоваться алгоритмы приближенного решения следующей задачи.

Задача о равномерном разбиении предметов (РРП). *Имеется n предметов и m параметров для их характеризации. Требуется разбить множество предметов на r подмножеств "равномерно" по всем параметрам, чтобы суммарные m-мерные характеристики во всех подмножествах мало отличались друг от друга.*

Экономической интерпретацией этой задачи может служить следующая

Задача объемно-календарного планирования (ОКП). *Годовой план предприятия, представленный n наименованиями, требуется разбить по кварталам (или по месяцам) как можно равномернее по каждому из m параметров.*

Наконец, приведем формальную постановку сформулированных выше задач.

Задача РРП$_{m,s}(X, r)$. *В пространстве $\mathbb{R}^m$ с нормой s задано семейство векторов $X = \{x_1, \ldots, x_n\} \subset B_{m,s}$, где $B_{m,s}$ — единичный шар нормы s. Требуется найти разбиение $H = \{N_1, \ldots, N_r\}$ множества индексов $\mathbb{N}_n$ на r подмножеств, минимизирующее функционал*

$$\zeta_X^s(H) = \max_{k\in\mathbb{N}_r} \left\| \sum_{i\in N_k} x_i - \frac{1}{r}\sum_{i\in\mathbb{N}_n} x_i \right\|_s .$$

Как утверждается в следующей теореме, приближенное решение задачи РРП может быть найдено с использованием алгоритма решения задачи компактного суммирования векторов.

Теорема 18.1 *Пусть существует алгоритм $\mathcal{A}$ трудоемкости $O(\mathcal{T}_{\mathcal{A}}(n,m))$ решения задачи КСВ$_{m,s}(X)$ с априорной оценкой*

$$f_{s,X}(\pi) \le \mu_s(m), \tag{18.1}$$

справедливой для любого исходного семейства векторов $X = \{x_1, \ldots, x_n\} \subset B_{m,s}$. *Тогда существует алгоритм той же трудоемкости для отыскания решения* H *задачи РРП*$_{m,s}(X, r)$ *с гарантированной оценкой*

$$\zeta_X^s(H) \le 2\mu_s(m). \tag{18.2}$$

Доказательство. Пусть $X = \{x_1, \ldots, x_n\} \subset B_{m,s}$ — исходное семейство векторов; $n_1 = \min\{j \,|\, j \ge n;\ j \text{ делится на } r\}$; $k = n_1/r$; $n_2 = n + n_1$. К семейству X добавим n_1 "новых" векторов $\{x_j \,|\, j = n+1, \ldots, n_2\}$, равных вектору $\bar b \doteq -\frac{1}{n_1}\sum_{j=1}^{n} x_j$. Ясно, что $n_2 = O(n)$; $\bar b \in B_{m,s}$ и $\sum_{j=1}^{n_2} x_j = 0$, т.е. $X' = \{x_1, \ldots, x_{n_2}\}$ является s-семейством векторов в пространстве $\mathbb{R}^m$. Применяя к нему алгоритм $\mathcal{A}$ из формулировки теоремы, найдем перестановку π индексов $i \in \mathbb{N}_{n_2}$, удовлетворяющую оценке (18.1). Для каждого $i \in \mathbb{N}_r$ находим индекс $j(i)$, равный такому максимальному $j \le n_2$, что множество индексов $\{\pi_1, \ldots, \pi_j\}$ содержит ровно ki индексов новых векторов; положим $j(0) = 0$. Разбиение $H = \{N_1, \ldots, N_r\}$ определим из соотношений

$$N_i = \{\pi_{j(i-1)+1}, \ldots, \pi_{j(i)}\} \cap \mathbb{N}_n,\ i \in \mathbb{N}_r.$$

Тогда искомая оценка (18.2) вытекает из соотношений

$$\left\| \sum_{j\in N_i} x_j - \frac{1}{r}\sum_{j\in\mathbb{N}_n} x_j \right\|_s = \left\| \sum_{\nu=1}^{j(i)} x_{\pi_\nu} - \sum_{\nu=1}^{j(i-1)} x_{\pi_\nu} - k\bar b + k\bar b \right\|_s$$

$$\le \left\| \sum_{\nu=1}^{j(i)} x_{\pi_\nu} \right\|_s + \left\| \sum_{\nu=1}^{j(i-1)} x_{\pi_\nu} \right\|_s \overset{\text{из (18.1)}}{\le} 2\mu_s(m),\ i \in \mathbb{N}_r. \quad \square \tag{18.3}$$

Из теоремы 18.1 и следствия 11.8 теоремы 11.5 со стр. 75 вытекает следующее

Следствие 18.2 *С использованием алгоритма компактного суммирования векторов, для задачи РРП*$_{m,s}(X, r)$ *с* n *предметами и произвольной нормой* s *с трудоемкостью* $O(n^2m^2)$ *может быть найдено решение* H *с гарантированной оценкой*

$$\zeta_X^s(H) \le 2\left(m - 1 + \frac{1}{m}\right). \quad \square \tag{18.4}$$

Приведенное выше решение задачи РРП с использованием алгоритма компактного суммирования векторов основывается на свойстве "малой удаленности" отыскиваемой траектории суммирования векторов $\{x_1, \ldots, x_n\}$ от прямой $\{t\bar b \,|\, t \in \mathbb{R}\}$ на всем протяжении этой траектории. Однако ясно, что для приближенного решения задачи РРП такое свойство является излишним, поскольку нам достаточно, чтобы траектория сумм векторов была близка лишь к выделенным точкам этой прямой, а именно, к точкам $\frac{i}{r}\sum_{i\in\mathbb{N}_n} x_i$. Понятно, что такому более слабому требованию мы можем удовлетворить с бóльшим успехом, что и подтверждается последующими результатами. В доказываемой ниже теореме приближенное решение задачи РРП$_{m,s}(X, r)$ строится на основе решения задачи БВ$_{m,s}(X, x)$ из раздела 10.

Теорема 18.3 *Пусть для m-мерного векторного пространства $\mathbb{R}^m$ с нормой s существует алгоритм $\mathcal{A}$ трудоемкости $O(\mathcal{T}_{\mathcal{A}}(n,m))$, который для любого исходного семейства векторов $X=\{x_1,\dots,x_n\}\subset B_{m,s}$ и точки $x\in P(X)$ находит решение задачи БВ$_{m,s}(X,x)$ с априорной оценкой*

$$\xi^s_{X,x}(N')\le\phi_s(m). \tag{18.5}$$

Тогда существует алгоритм трудоемкости $O(\mathcal{T}_{\mathcal{A}}(n,m)r)$ для отыскания решения H задачи РРП$_{m,s}(X,r)$ с гарантированной оценкой

$$\zeta^s_X(H)\le\nu(r)\,\phi_s(m), \tag{18.6}$$

где функция $\nu(i)$ определяется из рекуррентных соотношений:

$$\nu(1)=0;\ \nu(i)=\min_{i'\in\mathbb{N}_{i-1}}\max\{\nu(i')+1/i',\nu(i-i')+1/(i-i')\},\ i>1. \tag{18.7}$$

Доказательство. Алгоритм решения задачи РРП состоит из двух шагов.

Шаг 1. Последовательно для $i=1,\dots,r$ находим значения $\nu(i)$ и $i'=i'(i)\in\mathbb{N}_{i-1}$, где $\nu(i)=\nu(i')+1/i'$. Нахождение этих значений по формулам (18.7) требует не более $O(r^2)$ вычислительных операций.

Шаг 2. Находим разбиение $H=\{N_1,\dots,N_r\}$ множества индексов $\mathbb{N}_n$ на r подмножеств с помощью рекурсивной процедуры $Parti(\mathbb{N}_n,r,H)$. ■

Процедура $Parti(N,r,H)$ состоит из четырех пунктов.

1. Если $r=1$, то $\{H:=\{N\};$ **RETURN**$\}$[60].
2. Решая задачу ПВЗС$_{m,s}(X,x)$ для семейства векторов $X=\{x_i\,|\,i\in N\}$ и точки $x=\frac{i(r)}{r}\sum_{i\in N}x_i$ согласно алгоритму $\mathcal{A}$, находим подмножество индексов $N'\subseteq N$, для которого выполнена оценка (18.5).
3. Выполняем процедуры $Parti(N',i(r),H_1)$ и $Parti(N\setminus N',r-i(r),H_2)$.
4. $H:=H_1\cup H_2$. ■

Убедимся, что разбиение H, отыскиваемое с помощью описанной процедуры, удовлетворяет оценке (18.6).

При $r=1$ это очевидно ввиду $\nu(1)=0$.

Пусть $r>1$, и для любого $r'<r$ оценка (18.6) справедлива. Пусть $N'\subseteq\mathbb{N}_n$ — подмножество индексов, найденное в п.2, а H_1 и H_2 — разбиения, найденные в п.3 процедуры $Parti(\mathbb{N}_n,r,H)$. Тогда $\forall\ N_k\in H_1$ имеем

$$\left\|\sum_{j\in N_k}x_j-\frac{1}{r}\sum_{j\in\mathbb{N}_n}x_j\right\|_s\le\left\|\sum_{j\in N_k}x_j-\frac{1}{i(r)}\sum_{j\in N'}x_j\right\|_s+\frac{1}{i(r)}\left\|\sum_{j\in N'}x_j-\frac{i(r)}{r}\sum_{j\in\mathbb{N}_n}x_j\right\|_s$$

[60] Оператор **RETURN**, в отличие от **STOP**, прекращает выполнение данного вызова процедуры, но не всего алгоритма. Это отличие для нас существенно ввиду рекурсивного определения процедуры.

$$\overset{\text{из (18.6) и (18.5)}}{\leq} \nu(i(r))\,\phi_s(m) + \frac{1}{i(r)}\,\phi_s(m) = \left(\nu(i(r)) + \frac{1}{i(r)}\right)\,\phi_s(m).$$

Аналогично, $\forall\ N_k \in H_2$ получаем оценку

$$\left\| \sum_{j\in N_k} x_j - \frac{1}{r}\sum_{j\in\mathbb{N}_n} x_j \right\|_s \leq \left(\nu(r-i(r)) + \frac{1}{r-i(r)}\right)\,\phi_s(m),$$

откуда, согласно определению (18.7) функции $\nu(r)$ и определению функции $\zeta_X^s(H)$, получаем требуемую оценку (18.6).

Поскольку в приведенном выше алгоритме решение задачи $\text{РРП}_{m,s}(X,r)$ фактически сводится к решению $r-1$ задач $\text{ПВЗС}_{m,s}(X,x)$ с не более чем n векторами в каждом семействе X, то трудоемкость алгоритма не превосходит $O(\mathcal{T}_{\mathcal{A}}(n,m)r)$. Теорема 18.3 доказана. □

Нетрудно убедиться, что функция $\nu(r)$, определенная в теореме 18.3, имеет верхнюю границу, не зависящую от r. Для целого L обозначим $\nu_L = \max\{\nu(r)\,|\,r\in\mathbb{N}_L\}$, $\nu'_L = \max\{\nu(r)\,|\,r = L,\ldots,2L\}$. Определим величину $\nu''_L = \sup\{\nu(r)\,|\,r = L, L+1,\ldots\} = \sup\{\nu'_{2^kL}\,|\,k=0,1,\ldots\}$ и покажем, что она конечна. Действительно, $\forall\ r = 2L,\ldots,4L$ из (18.7) для $k = \lfloor\frac{r}{2}\rfloor \in \{L,\ldots,2L\}$ имеем оценку

$$\nu(r) \leq \max\left\{\nu(k) + \frac{1}{k}, \nu(r-k) + \frac{1}{r-k}\right\} \leq \nu'_L + \frac{1}{L},$$

откуда $\nu'_{2L} \leq \nu'_L + \frac{1}{L}$, $\nu'_{4L} \leq \nu'_L + \frac{1}{L} + \frac{1}{2L}, \ldots,\ \nu''_L \leq \nu'_L + \frac{2}{L}$. Таким образом, величина $\bar\nu = \max\left\{\nu_L, \nu'_L + \frac{2}{L}\right\}$ задает верхнюю границу значений $\nu(r)$ по всем $r \geq 1$.

Нетрудно видеть, что если найдется L, для которого выполняется

$$\nu'_L + \frac{2}{L} < \nu_L, \tag{18.8}$$

то $\bar\nu = \nu_L$. Последнее означает, что **максимум** функции $\nu(r)$ достигается на некотором значении $r \in \mathbb{N}_L$. С помощью несложной ПАСКАЛЬ-программы значения $\nu(r)$ были посчитаны по формуле (18.7) для $r \in \mathbb{N}_{10000}$. Было установлено, что $\nu(r) < 2,\ \forall\ r \in \mathbb{N}_{250}$. Кроме того, при $L = 4000$ установлена справедливость неравенства (18.8) и найдено максимальное из чисел $\{\nu(r)\}$. Им оказалось значение $\nu(909) < 2.00045038$. С учетом этого результата и леммы 10.8 получаем следующее

Следствие 18.4 *С использованием алгоритма из леммы 10.8 для задачи $\text{РРП}_{m,s}(X,r)$ с n предметами и произвольной нормой s с трудоемкостью $O(nrm^2)$ может быть найдено решение H с гарантированной оценкой*

$$\zeta_X^s(H) \leq 1.00023m. \tag{18.9}$$

При $r \leq 250$ оценка улучшается до $\zeta_X^s(H) \leq m$. □

Сформулированный выше результат приведен в работе [45]. В более ранней работе диссертанта [34] представлен алгоритм решения задачи РРП, позволяющий с учетом результата леммы 10.8 получать решения задачи РРП$_{m,s}(X,r)$ для произвольной нормы s с чуть более слабой оценкой

$$\zeta^s_X(H) \le 1.06m. \tag{18.10}$$

Тот алгоритм также основывался на решении задачи ПВЗС, однако при этом использовалась совсем другая схема разбиения семейства векторов. Конкретно, номера $\{i\}$ узлов $\frac{i}{r}\,x$, к которым мы стремимся приблизить суммы векторов $\sum_{j\in N^i} x_j$, $N^i \doteq \cup_{\nu=1}^{i} N_\nu$, выбирались в порядке убывания ранга числа i (см. раздел 17), — аналогично тому, как мы поступаем сейчас в разделе 26 при построении расписания в задаче open shop. Поскольку выбранная в [34] схема разбиения семейства векторов не являлась единственно возможной, оставался открытым вопрос: можно ли предложить другую схему разбиения, которая позволяла бы заменить коэффициент 1.06 в оценке (18.10) на 1? Представленный выше результат дает отрицательный ответ на этот вопрос, поскольку используемая в нашем новом алгоритме схема является в каком-то смысле наилучшей возможной (среди алгоритмов, основанных на решении задачи ПВЗС).

19 Задачи о максимальном потоке, равномерном по стокам, и о распределении камней с запретами

Рассмотрим задачу о распределении камней с запретами (РКЗ), которая отличается от классической задачи о распределении камней по кучам тем, что для каждого "камня" задано множество допустимых куч.

Задача РКЗ. *Имеется N "камней" с заданными весами $b_1,\dots,b_N$ и множество "куч" $\mathcal{M}=\{1,\dots,m\}$, $m>1$. Требуется распределить камни по кучам, минимизируя вес максимальной кучи ($l_{\max}$), если для каждого камня $j\in\mathbb{N}_N$ задано множество допустимых куч $\mathcal{M}_j\subseteq\mathcal{M}$.*

Замечание 19.1 *Без ограничения общности можем считать, что $\bigcup_{j=1}^{N}\mathcal{M}_j=\mathcal{M}$. В противном случае переопределим множество $\mathcal{M}$, положив $\mathcal{M}:=\bigcup_{j=1}^{N}\mathcal{M}_j$.*

Обозначим $B=\{b_i\,|\,i\in\mathbb{N}_N\}$, $b_{\max}=\max_i b_i$, E_{opt} — оптимальное распределение камней в задаче РКЗ.

Задаче РКЗ поставим в соответствие задачу о нахождении максимального потока, равномерного по стокам в сети $G=(V,U)$. Множество вершин V содержит источник s, множество внутренних вершин $V_1\doteq\{v_1,\dots,v_N\}$ и множество стоков $V_2\doteq\{w_1,\dots,w_m\}$. Множество дуг состоит из двух подмножеств: $U=U_1\cup U_2$, где $U_1=\{(s,v_j)\,|\,j\in\mathbb{N}_N\}$; $U_2=\bigcup_{j=1}^{N}U_2^j$, $U_2^j=\{(v_j,w_i)\,|\,i\in\mathcal{M}_j\}$. На дугах определена функция пропускной способности

$$c(u)=\begin{cases} b_j & ,u=(s,v_j)\in U_1,\\ \infty & ,u\in U_2.\end{cases}$$

Пусть F — множество допустимых потоков в сети G. Для $f \in F$ определим вектор $d_f = (d_f(1), \ldots, d_f(m)) \in \mathbb{R}^m$, где

$$d_f(i) = \sum_{(v_j, w_i) \in U} f(v_j, w_i).$$

Обозначим: $P(f) = \sum_{i=1}^{m} d_f(i)$ — мощность потока f, $P_o = \max_{f \in F} P(f)$, $F_o = \{f \in F \mid P(f) = P_o\}$.

Поток $f \in F_o$ назовем *правильным*, если $f(u) \in \{0, b_j\},\ \forall u \in U_2^j,\ j \in \mathbb{N}_N$. Нетрудно видеть, что между правильными потоками и допустимыми распределениями камней по кучам существует взаимнооднозначное соответствие.

Определим функцию $\gamma : \mathbb{R}^m \to \mathbb{R}^m$, которая упорядочивает координаты каждого вектора $x \in \mathbb{R}^m$ по невозрастанию, и сформулируем задачу МПРС о нахождении в сети G максимального потока, наиболее равномерного по стокам.

Задача МПРС. *Найти поток $f_o \in F_o$, на котором достигается*

$$\operatorname{lex}\min_{f \in F_o} \gamma(d_f) \doteq (\gamma_1^o, \ldots, \gamma_m^o).$$

Очевидно, для любого распределения E камней по кучам в задаче РКЗ справедлива оценка

$$l_{\max}(E) \geq \gamma_1^o. \tag{19.1}$$

Теорема 19.2 *Существует алгоритм трудоемкости $O(N^2m^2)$, находящий такое распределение E' камней по кучам в задаче* РКЗ*, для которого выполнена оценка*

$$l_{\max}(E') \leq \gamma_1^o + b_{\max}. \tag{19.2}$$

Заметим, что из оценок (19.1),(19.2) вытекает оценка абсолютной погрешности алгоритма:

$$l_{\max}(E') - l_{\max}(E_{\text{opt}}) \leq b_{\max}.$$

Справедливость теоремы 19.2 вытекает из следующих двух лемм.

Лемма 19.3 *Оптимальный поток f_o в задаче* МПРС *строится за $O(N^2m^2)$ операций.*

Лемма 19.4 *Любой поток $f \in F_o$ с трудоемкостью $O(Nm \cdot \min\{N, m\})$ перестраивается в правильный поток $f' \in F_o$ такой, что*

$$d_{f'}(i) - d_f(i) \leq b_{\max},\ i \in \mathbb{N}_m \tag{19.3}$$

Доказательство леммы 19.3. Из сети G получим сеть G' добавлением фиктивного стока t и дуг $(w_i, t),\ i \in \mathbb{N}_m$, с пропускными способностями $c(w_i, t) = P_o/m$. Найдем максимальный поток f' в сети G' и определим сеть $G_{f'}$ с "остаточными" пропускными способностями на дугах, следуя работе [1]. (Напомним, что сеть $G_{f'}$ имеет такое же множество вершин V, а множество дуг U' образуется следующим образом. Включаем в U' все дуги из U и обратные к ним, определив на них пропускные способности c':

$$c'(u) := c(u) - f'(u);\ c'(u') := f'(u),$$

где u' — дуга, обратная к u. После этого исключаем из U' все дуги с $c'(u) = 0$.) Найдем в сети G' минимальный разрез $R = (X, Y)$, определив X как множество вершин, доступных из вершины s в сети $G_{f'}$. (Понятно, что $s \in X$.) Если $P_{f'} = P_o$, то поток f' — оптимальный. Пусть

$$P_{f'} < P_o. \tag{19.4}$$

Докажем несколько простых утверждений

Утверждение 19.5 *Каждое из множеств $X \cap V_1$, $Y \cap V_1$, $X \cap V_2$, $Y \cap V_2$ непусто.*

Доказательство. Равенство $X \cap V_1 = \emptyset$ означало бы, что X состоит из единственной вершины s. Но тогда все дуги из U_1 насыщены потоком f', т.е. $P(f') = P_o$, что противоречит (19.4). Таким образом, $X \cap V_1 \neq \emptyset$. Аналогично доказывается, что $Y \cap V_2 \neq \emptyset$.

Поскольку $X \cap V_1 \neq \emptyset$ и дуги $\{u \in U_2^j \mid j \in X \cap V_1\}$ не могут быть насыщены (они имеют бесконечную пропускную способность), то их конечные вершины (из V_2) также принадлежат X, откуда $X \cap V_2 \neq \emptyset$.

Если $Y \cap V_1 = \emptyset$, то это возможно лишь в том случае, если во множество $Y \cap V_2$ не заходит ни одной дуги, что противоречит замечанию 19.1 □

Утверждение 19.6 *Для $f = f'$ выполнено свойство*

$$f(u) = 0,\ \forall u \in (Y \cap V_1, X \cap V_2). \tag{19.5}$$

Доказательство. Если для дуги $u = (v_j, w_i),\ v_j \in Y \cap V_1,\ w_i \in X \cap V_2$, выполнено $f(u) \neq 0$, то в сети $G_{f'}$ существует обратная дуга $u' = (w_i, v_j)$ с ненулевой пропускной способностью, и тогда $v_j \in X$, что противоречит $v_j \in Y$ □

Утверждение 19.7 *Для $f = f_o$ выполнено свойство (19.5).*

Доказательство. Допустим противное, т.е.

$$f_o(Y \cap V_1, X \cap V_2) \doteq \delta > 0. \tag{19.6}$$

Из определения множества X следует, что множество дуг из $X \cap V_1$ в $Y \cap V_2$ пусто, откуда

$$f(X \cap V_1, Y \cap V_2) = 0,\ \forall f. \tag{19.7}$$

Кроме того, все дуги $\{(s, v_j) \,|\, v_j \in Y \cap V_1\}$ насыщены потоком f'. Так как поток f_o, будучи максимальным потоком в сети G, также насыщает все дуги из $(s, Y \cap V_1)$ (как и вообще все дуги из (s, V_1)), то

$$f_o(s, Y \cap V_1) = f'(s, Y \cap V_1), \tag{19.8}$$

и разница в мощностях потоков f_o и f' слагается из разницы этих потоков на дугах множества $(s, X \cap V_1)$, т.е.

$$P_o - P(f') = f_o(s, X \cap V_1) - f'(s, X \cap V_1). \tag{19.9}$$

Так как из (19.5),(19.7) имеем

$$\sum_{w_i \in X \cap V_2} d_{f'}(i) = f'(s, X \cap V_1),$$

а из (19.6) и (19.7):

$$\sum_{w_i \in X \cap V_2} d_{f_o}(i) = f_o(s, X \cap V_1) + \delta,$$

то с учетом (19.9) получаем

$$\sum_{w_i \in X \cap V_2} (d_{f_o}(i) - d_{f'}(i)) = P_o - P(f') + \delta. \tag{19.10}$$

Из (19.6), (19.7) и (19.8) также имеем

$$\sum_{w_i \in Y \cap V_2} (d_{f_o}(i) - d_{f'}(i)) = -\delta. \tag{19.11}$$

Разложим положительный допустимый поток $f'' = f' \ominus f_o$ в сети G_{f_o} на элементарные потоки вдоль путей и циклов согласно теореме из [1, стр. 11]. Из соотношений (19.10),(19.11) следует, что один из элементарных потоков ($\bar{f}$) будет направлен вдоль пути из вершины $w_{i'} \in X \cap V_2$, где выполняется $d_{f_o}(i') > d_{f'}(i') = P_o/m$, в вершину $w_{i''} \in Y \cap V_2$, где $d_{f_o}(i'') < d_{f'}(i'') \le P_o/m$. Следовательно, при малых $\varepsilon > 0$ поток $f_o \oplus \varepsilon \bar{f}$ будет, во-первых, допустимым потоком в сети G, а во-вторых, — более оптимальным решением задачи МПРС по сравнению с потоком f_o. Противоречие доказывает свойство (19.5) для потока f_o □

Из утверждений 19.6, 19.7 следует, что задача МПРС разбивается на две независимые подзадачи: для сети G_1 с источником s, множеством внутренних вершин $X \cap V_1$ и множеством стоков $X \cap V_2$ и — для сети G_2 с источником s, множеством внутренних вершин $Y \cap V_1$ и множеством стоков $Y \cap V_2$. Отсюда с учетом утверждения 19.5 следует, что алгоритм построения потока f_o состоит из не более чем $\min\{N, m\} - 1$ этапов разбиения сети G на элементарные подсети, у которых либо число внутренних вершин, либо число стоков равно единице, и из не более чем $\min\{N, m\}$ этапов построения оптимальных потоков в элементарных подсетях. На каждом из этапов разбиения находим

максимальный поток в сети G_i, являющейся подсетью сети G, и при этом получаем либо оптимальный поток в G_i, либо ее разбиение на две подсети меньшего размера. Применяя для нахождения максимального потока на каждом из этапов разбиения алгоритм тупиковых потоков [1, параграф 1.5], трудоемкость которого равна $O((N+m)^2K)$, где K — число k-этапов, и оценивая K величиной $O(\min\{N,m\})$, получаем суммарную трудоемкость этапов разбиения, равную $O(\max^2\{N,m\}\cdot\min^2\{N,m\}) = O(N^2m^2)$. Поскольку суммарная трудоемкость построения оптимальных потоков в элементарных подсетях не превосходит $O(N+m)$, то лемма 19.3 доказана.

Доказательство леммы 19.4. Пусть поток $f \in F_o$ на дугах из множества U_2 задается матрицей $(f_{ji})_{j=1,\dots,N}^{i=1,\dots,m}$. Построим матрицу $D = (d_{ji})_{j=1,\dots,N}^{i=1,\dots,m}$, в которой

$$d_{ji} = \begin{cases} 1, & \text{если } 0 < f_{ji} < b_j, \\ 0, & \text{в противном случае.} \end{cases}$$

Так как $P(f) = \sum_{j=1}^{N} b_j$, то в каждой ненулевой строке матрицы D должно быть не меньше двух единиц.

Далее вычислим матрицы $D_1 = (d^1_{ji})$ и $D_2 = (d^2_{ji})$ ($j \in \mathbb{N}_N$; $i \in \overline{\mathbb{N}}_m$) горизонтальной адресации, элементы которых для каждого $d_{ji} = 1$ указывают адрес предыдущего и последующего единичного элемента матрицы D в строке j (с учетом "замыкания" последнего элемента на первый), а также позволяют найти первый единичный элемент в строке j, если такой имеется. Аналогично определим матрицы D_3 и D_4 вертикальной адресации, а также матрицы $\Gamma_1 = (g^1_j)$, $\Gamma_2 = (g^2_j)$, $j = 0,\dots,N$, позволяющие найти номер произвольной ненулевой строки и указывающие номера предыдущей и последующей ненулевых строк соответственно для каждой ненулевой строки матрицы D.

Отметим главные особенности работы алгоритма с матрицами адресации.

- Для каждого единичного элемента матрицы D имеется возможность за $O(1)$ операций найти следующий единичный элемент в той же строке и том же столбце либо выяснить, что такого не существует.
- За $O(1)$ операций можно найти ненулевую строку матрицы D.
- Вычисление исходных массивов адресации требует не более $O(Nm)$ операций (для чего достаточно сделать лишь один просмотр матрицы D).
- При обнулении какого-либо единичного элемента матрицы D необходимые изменения в массивах адресации (переадресация) осуществляются за $O(1)$ операций.

Рассмотрим граф G^* со множеством вершин $V^* = V_1 \cup V_2$ и множеством дуг $U^* = \{(v_j, w_i),\ (w_i, v_j) \mid d_{ji} = 1\}$. На каждой дуге из U^* определим новую пропускную способность c^*:

$$c^*(v_j, w_i) = b_j - f_{ji}, \tag{19.12}$$

$$c^*(w_i, v_j) = f_{ji}. \tag{19.13}$$

Определим два случая, в которых выполняется элементарная перестройка потока f.

a Если найден перестраивающий цикл в графе G^* (т.е. цикл, не содержащий одновременно прямую и обратную дугу), то следует изменить поток вдоль дуг цикла на величину пропускной способности цикла ("a-перестройка").

b Если найдена вершина w_i, инцидентная только одной вершине v_j в графе G^*, то для всех $i' \in \mathbb{N}_m$, $i' \neq i$ уменьшается поток на дугах $(v_j, w_{i'})$ на величину $f_{ji'}$, при этом поток на дуге (v_j, w_i) увеличивается на величину $b_j - f_{ji}$, что соответствует "склеиванию" камня j и помещению его в кучу i ("b-перестройка").

После каждой перестройки потока пересчитываются пропускные способности c^* на дугах из U^* по формулам (19.12),(19.13), и если на какой-то дуге получается $c^* = 0$, то удаляем эту дугу вместе с обратной из множества U^*, занулив при этом соответствующий элемент d_{ji} матрицы D и сделав переадресацию во всех массивах адресации.

Оценка (19.3) вытекает из следующих трех фактов.

1. При a-перестройке" потока f значения $d_f(k)$, $k \in \mathbb{N}_m$, не меняются.

2. При b-перестройке" потока f значение $d_f(k)$ увеличивается только для $k = i$, причем не более чем на величину $b_{\max}$.

3. Так как после увеличения значения $d_f(i)$ вершина w_i в графе G^* становится изолированной, то для всех $k \in \mathbb{N}_m$ увеличение значения $d_f(k)$ в течение всего алгоритма перестройки происходит не более одного раза.

С целью получения оценки трудоемкости алгоритма перестройки потока f опишем его более подробно. Алгоритм состоит из этапов, каждый из которых сопровождается элементарной перестройкой потока и занулением хотя бы одного элемента матрицы D. Таким образом, число этапов не превосходит Nm.

Этап алгоритма начинается с отыскания ненулевой строки j_1 матрицы D и в ней — единичного элемента $d_{j_1 i_1}$. Вершины j_1 и i_1 помечаются. Затем находится следующий единичный элемент $d_{j_2 i_1}$ в столбце i_1, помечается вершина j_2, находится следующий единичный элемент $d_{j_2 i_2}$ в строке j_2, помечается вершина i_2, и т.д., пока не придем к одному из двух случаев, в которых осуществляется элементарная перестройка потока. (Это произойдет, когда либо в очередном столбце i_k не окажется других единичных элементов, либо попадем на уже помеченную вершину, образовав цикл.) Так как граф G^* двудольный, то длина построенной последовательности вершин не превосходит $2\min\{N, m\} + 1$, а следовательно, ее построение, а также элементарная a-перестройка потока и последующее удаление пометок требует $O(\min\{N, m\})$ действий. Для b-перестройки потока требуется $O(m)$ действий. Но поскольку после каждой b-перестройки в матрице D зануляются хотя бы один столбец и одна строка, то количество этапов, заканчивающихся b-перестройкой, не превосходит $O(\min\{N, m\})$. Отсюда трудоемкость всего алгоритма перестройки потока f в искомый поток f' равна $O(Nm \cdot \min\{N, m\})$.

Лемма 19.4 доказана.

Доказательство теоремы 19.2. Из лемм 19.3 и 19.4 вытекает следующий алгоритм решения задачи РКЗ с оценкой точности (19.2).

Алгоритм $\mathcal{A}_{19.1}$

Алгоритм состоит из двух этапов.

Этап 1. По входу задачи РКЗ построим сеть $G = (V, U)$ с одним источником и m стоками, как это описано выше. Применяя алгоритм леммы 19.3, решим задачу МПРС о нахождении в сети G максимального потока f_o, наиболее равномерного по стокам.

Этап 2. Перестроим поток f_o в правильный поток f', мощность которого в каждом из стоков $i \in \mathbb{N}_m$ удовлетворяет оценке (19.3). Ненулевые значения потока f' задают искомое распределение камней по кучам в задаче РКЗ.

Алгоритм $\mathcal{A}_{19.1}$ описан.

Оценка трудоемкости алгоритма $\mathcal{A}_{19.1}$ складывается из оценок трудоемкостей алгоритмов, работающих на его этапах. Из лемм 19.3 и 19.4 следует, что эта сумма не превосходит $O(N^2m^2)$. Оценка точности (19.2) алгоритма $\mathcal{A}_{19.1}$ вытекает из неравенств (19.3).

Теорема 19.2 доказана.

20 Связи между комбинаторными задачами

В этом разделе мы сформулируем задачи:

- Дворецкого;
- балансировки;
- “integer making”;
- эквидистанции в булевом кубе.

Будут показаны их связи с сформулированными ранее задачами РРП (стр.135), БВ (стр.64) и КСВ (стр.70), которые позволяют, во-первых, имея алгоритмы приближенного решения одних задач, строить аналогичные алгоритмы для других задач, и во-вторых, отыскивать нетривиальные оценки экстремальных функций, определяемых для каждой из рассматриваемых оптимизационных задач.

Введем несколько обозначений, используемых в этом разделе:
$\mathcal{H}_{n,r} = \{H \,|\, H : \ \mathbb{N}_n \to \mathbb{N}_r\}$ обозначает множество разбиений n-элементного множества на r подмножеств;
$\mathcal{F}_{n,m} = \{F \,|\, F : \ \mathbb{N}_m \to 2^{\mathbb{N}_n}\}$ обозначает множество, каждый элемент которого есть семейство из m подмножеств n-элементного множества;
$B^n = \{0,1\}^n$ обозначает n-мерный булевый куб.

Задачи и функции

Задача Дворецкого. *В пространстве $\mathbb{R}^m$ с нормой s задано конечное семейство векторов не более чем единичной длины каждый: $X = \{x_1, \ldots, x_n\} \subset B_{m,s}$. Требуется найти семейство чисел $E = \{\varepsilon_i = \pm 1 \,|\, i \in \mathbb{N}_n\}$, минимизирующее функцию $d^s_X(E) = \| \sum_{i=1}^n \varepsilon_i x_i \|_s$.*

Функция

$$D_s(m) \doteq \sup_{X \subset B_{m,s}} \min_E d^s_X(E)$$

определяет значение оптимума целевой функции в задаче Дворецкого в наихудшем случае. Корректность определения этой функции была показана Дворецким. Проблема ее нахождения сформулирована Дворецким в 1963 году и включена в список нерешенных проблем 7-го Симпозиума AMS по чистой математике [83, p.496].

Задача балансировки (Олсон и Спенсер [101]). *Заданы множество $\mathbb{N}_n$ и семейство $F = \{F_1, \ldots, F_m\}$ его подмножеств. Требуется найти разбиение $H = \{N_1, N_2\}$ множества $\mathbb{N}_n$ на два подмножества: $\mathbb{N}_n = N_1 \cup N_2, \quad N_1 \cap N_2 = \emptyset$, минимизирующее функцию*

$$q_F(H) \doteq \max_{i \in \mathbb{N}_m} \left| |F_i \cap N_1| - |F_i \cap N_2| \right|.$$

Другими словами, мы хотим найти разрез $(N_1, \mathbb{N}_n \setminus N_1)$ множества $\mathbb{N}_n$, который делит каждое подмножество F_i из заданного семейства на две как можно более равные части. Олсон и Спенсер поставили вопрос о нахождении функции

$$f(m) \doteq \sup_n f(n, m), \tag{20.1}$$

где $f(n,m) \doteq \max_{F \in \mathcal{F}_{n,m}} \min_{H \in \mathcal{H}_{n,2}} q_F(H)$, и доказали, что этот вопрос корректен, поскольку $\sup_n$ в правой части (20.1) ограничен сверху для каждого m.

Бек и Фиала [76] обобщили задачу балансировки на случай разбиения множества $\mathbb{N}_n$ на r подмножеств.

Задача r-балансировки. *Заданы множество $\mathbb{N}_n$ и семейство $F = \{F_1, \ldots, F_m\}$ его подмножеств. Требуется найти разбиение $H = \{N_1, \ldots, N_r\}$ множества $\mathbb{N}_n$ на r подмножеств, минимизирующее функционал*

$$q_F(H) \doteq \max_{i,j,k} \left| |F_i \cap N_j| - |F_i \cap N_k| \right|.$$

Определим функцию

$$\psi(r, m) \doteq \sup_n \max_{F \in \mathcal{F}_{n,m}} \min_{H \in \mathcal{H}_{n,r}} q_F(H).$$

Бек и Фиала показали, что функция ψ хорошо определена, поскольку она ограничена сверху функцией от m. Очевидно,

$$\psi(2, m) = f(m).$$

Задача "integer making" (Бек и Фиала [76]). *Заданы вектор* $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{R}^n$ *и семейство* $F = \{F_1, \ldots, F_m\}$ *подмножеств множества* $\mathbb{N}_n$. *Требуется найти целочисленный вектор* $\bar{\alpha} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_n)$, $|\bar{\alpha}_i - \alpha_i| < 1$, $(i \in \mathbb{N}_n)$, *минимизирующий функционал*

$$\chi_{F,\alpha}(\bar{\alpha}) \doteq \max_{F_k \in F} \left| \sum_{i \in F_k} \alpha_i - \sum_{i \in F_k} \bar{\alpha}_i \right|.$$

Без ограничения общности можно считать, что в данной задаче требуется для заданного вещественнозначного вектора $\alpha \in [0,1]^n$ найти "округляющий" вектор $\bar{\alpha} \in \{0,1\}^n$, "ближайший" к вектору α. В качестве меры близости вектора $\alpha - \bar{\alpha}$ к нулю берутся суммы его компонент по всем индексам из заданных "тестирующих" множеств $\{F_i\}$. Определяем экстремальные функции

$$g(n,m) \doteq \sup_{F \in \mathcal{F}_{n,m}} \sup_{\alpha \in [0,1]^n} \min_{\bar{\alpha} \in \{0,1\}^n} \chi_{F,\alpha}(\bar{\alpha});$$

$$g(m) \doteq \sup_n g(n,m).$$

Задача эквидистанции. *Задано семейство* $F = \{f_1, \ldots, f_m\} \subseteq B^n$ *вершин* n*-мерного куба. Требуется найти вершину* $f \in B^n$, *минимизирующую функцию*

$$\omega_F(f) \doteq \max_{i,j} |\rho(f, f_i) - \rho(f, f_j)|,$$

где $\rho(x,y)$ *— расстояние Хэмминга между булевыми векторами* $x, y \in B^n$.

Определим множество функций $\mathcal{F}'_{n,m} \doteq \{F \,|\, F : \mathbb{N}_m \to B^n\}$ (нетрудно заметить взаимно однозначное соответствие между множествами $\mathcal{F}'_{n,m}$ и $\mathcal{F}_{n,m}$) и определим экстремальные функции

$$\theta(n,m) \doteq \max_{F \in \mathcal{F}'_{n,m}} \min_{f \in B^n} \omega_F(f);$$

$$\theta(m) \doteq \sup_n \theta(n,m).$$

Ясно, что $\theta(n,m) \leq n$, и что $\theta(n,m)$ является функцией, возрастающей по n и m.

Определим также экстремальную функцию

$$\gamma_s(r,m) \doteq \sup_{n=|X|} \sup_{X \subset B_{m,s}} \min_{H \in \mathcal{H}_{n,r}} \zeta^s_X(H) \tag{20.2}$$

для задачи РРП из раздела 18 и функцию

$$\eta_s(n,m) \doteq \sup_{X \subset B_{m,s}} \sup_{x \in P(X)} \min_{N' \subseteq \mathbb{N}_n} \xi^s_{X,x}(N') \tag{20.3}$$

для задачи БВ из раздела 10. Выделим три важных частных случая задачи БВ с нормой $s = l_\infty$, когда векторы $\{x_i\}$ принимают значения из специальных множеств:

A) $x_i \in \{-1, 0, 1\}^m$;

B) $x_i \in [0, 1]^m$;

C) $x_i \in \{0, 1\}^m$.

Соответствующие подзадачи будем обозначать через БВ_Z, а их экстремальные функции — через η^Z ($Z \in \{A, B, C\}$). Если же при вычислении функции η или η^Z значения функции $\xi^{l_\infty}_{X,x}(N')$ определяются для единственной точки $x = \frac{1}{2}\sum_{x_i \in X} x_i$ (то бишь для центра параллелепипеда $P(X)$), то соответствующие задачи и их экстремальные функции будем отмечать чертой сверху: $\overline{\text{БВ}}, \overline{\text{БВ}}_Z, \bar\eta, \bar\eta^Z$.

Аналогично, задачу РРП с нормой $s = l_\infty$ и векторами $x_i \in B^m$ будем обозначать РРП_C, а соответствующую экстремальную функцию — через γ^C.

Наконец, определим функции

$$\eta_s(m) \doteq \sup_n \ \eta_s(n, m), \ \bar\eta_s(m) \doteq \sup_n \ \bar\eta_s(n, m),$$

$$\eta^Z(m) \doteq \sup_n \ \eta^Z(n, m), \ \bar\eta^Z(m) \doteq \sup_n \ \bar\eta^Z(n, m), \ Z \in \{A, B, C\}.$$

(Из дальнейшего будет видно, что все эти функции также хорошо определены.)

Связи между задачами и соотношения между функциями

Для краткости, задачи Дворецкого, балансировки, r-балансировки, Integer making и эквидистанции будем обозначать через *Dv, Бал, Бал(r), Im* и *Эк* соответственно. Задачу $\text{РРП}_{m,s}(X, r)$ о равномерном разбиении предметов на r куч будем кратко обозначать РРП(r).

1. ($\overline{\text{БВ}} \Leftrightarrow$ РРП(2) $\Leftrightarrow$ Dv) Из равенств

$$2\left\|\sum_{i\in N'} x_i - \frac{1}{2}\sum_{i\in\mathbb{N}_n} x_i\right\|_s = \left\|\sum_{i\in N'} x_i - \sum_{i\in\mathbb{N}_n\setminus N'} x_i\right\|_s = \left\|\sum_{i\in\mathbb{N}_n} \varepsilon_i x_i\right\|_s$$

(с $\varepsilon_i = \pm 1$), справедливых для всякого семейства векторов $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subset \mathbb{R}^m$ и подмножества $N' \subseteq \mathbb{N}_n$, вытекают соотношения между целевыми функциями:

$$\xi^s_{X,x_0}(N') = \zeta^s_X(H) = \frac{1}{2}\ d^s_X(E)$$

(где x_0 — центр параллелепипеда $P(X)$, а $H = \{N', \mathbb{N}_n \setminus N'\}$ — разбиение множества $\mathbb{N}_n$ на два подмножества) и между экстремальными функциями:

$$\bar\eta_s(m) = \gamma_s(2, m) = \frac{1}{2}\ D_s(m). \tag{20.4}$$

2. (КСВ $\Leftrightarrow$ Dv) Из теоремы 18.1 получаем соотношение между экстремальными функциями задач РРП и КСВ:

$$\gamma_s(r,m) \le 2\varphi_s(m),\ \forall\ r.$$

Нетрудно видеть, что при $r=2$ из (18.3) вытекает соотношение

$$\gamma_s(2,m) \le \varphi_s(m), \tag{20.5}$$

поскольку $\left\| \sum_{\nu=1}^{j(i)} x_{\pi_\nu} \right\|_s = 0$ при $i=r$ и $\left\| \sum_{\nu=1}^{j(i-1)} x_{\pi_\nu} \right\|_s = 0$ при $i=1$. Из (20.5) с учетом (20.4) получаем соотношение:

$$\frac{1}{2}\ D_s(m) \le \varphi_s(m). \tag{20.6}$$

А теперь заметим, что выбирая в (12.4) вектор $a_t = 0$, получаем оценку функции $\varphi_s(m)$:

$$\varphi_s(m) \ge \frac{1}{2} \sup_{\{a_1,\ldots,a_{t-1}\}\subset B_{m,s}} \min_{\lambda_i=\pm 1} \left\| \sum_{i=1}^{t-1} \lambda_i a_i \right\|_s = \frac{1}{2}\ D_s(m). \tag{20.7}$$

Любопытно, что в то время как вывод соотношения (20.6) основывался на существовании (гипотетического, т.е не известного нам) **наихудшего** примера в задаче РРП(2), вывод оценки (20.7), дающей в точности такое же соотношение между функциями $\varphi_s(m)$ и $D_s(m)$, основывается на построении **конкретного** примера задачи КСВ. Это косвенно подтверждает, что применяемый нами при выводе оценки (20.7) Метод 1 построения "плохих" семейств векторов в задаче КСВ не так уж плох.

3. (Бал(r) $\Leftrightarrow$ РРП$_C(r)$) Для семейства $F=\{F_1,\ldots,F_m\}$ подмножеств множества $\mathbb{N}_n$, заданного в задаче Бал(r), определим семейство векторов $X=\{x_i \,|\, i\in\mathbb{N}_n\}$ по формуле

$$x_i = (x_1^i,\ldots,x_m^i);\ x_j^i = \begin{cases} 1, & \text{если } i\in F_j, \\ 0, & \text{в противном случае.} \end{cases} \tag{20.8}$$

Тогда для разбиения $H=\{N_1,\ldots,N_r\}$ будем иметь

$$\begin{aligned} q_F(H) &= \max_{i,j,k} \big||F_i\cap N_j| - |F_i\cap N_k|\big| = \max_{j,k}\max_i \left| \sum_{\nu\in N_j} x_i^\nu - \sum_{\nu\in N_k} x_i^\nu \right| \\ &= \max_{j,k} \| \sum_{\nu\in N_j} x_\nu - \sum_{\nu\in N_k} x_\nu \|_{l_\infty} \le 2\zeta_X^{l_\infty}(H), \end{aligned}$$

откуда следует соотношение

$$\psi(r,m) \le 2\gamma^C(r,m). \tag{20.9}$$

При $r=2$ в (20.9) имеем равенство:

$$\psi(2,m) = 2\gamma^C(2,m). \tag{20.10}$$

4. (Бал(2) $\Leftrightarrow$ $\overline{\text{БВ}}_C$ $\Leftrightarrow$ $\overline{\text{БВ}}_A$ $\Leftrightarrow$ $\overline{\text{БВ}}_B$ $\Leftrightarrow$ $\overline{\text{БВ}}$) Для разбиения $H = \{N_1, N_2\}$ в задаче Бал(2) и семейства векторов $X = \{x_i \,|\, i \in \mathbb{N}_n\} \subseteq B^m$, определенных в задаче $\overline{\text{БВ}}_C$ согласно (20.8), выполнены равенства

$$q_F(H) = \max_{j\in\mathbb{N}_m} \left| \sum_{i\in\mathbb{N}_n} \varepsilon_i x_j^i \right| = 2 \left\| y - \frac{1}{2}\sum x_i \right\|_{l_\infty} = 2\xi^{l_\infty}_{X,x_0}(N_1),$$

где $y = \sum_{i\in N_1} x_i$, x_0 — центр параллелепипеда $P(X)$, $\varepsilon_i = 1$ для $i \in N_1$ и $\varepsilon_i = -1$ для $i \in N_2$. Из выписанных равенств следуют соотношения $f(n,m) = 2\bar{\eta}^C(n,m)$ и

$$\psi(2,m) = 2\bar{\eta}^C(m).$$

Для получения приближенного решения в задаче балансировки Олсон и Спенсер применяют следующий трюк. Объединяя векторы x_i в попарные разности: $x'_1 = x_1 - x_2$, $x'_2 = x_3 - x_4, \ldots$, они приходят к семейству векторов x'_i, состоящему из $n' = \lfloor \frac{n+1}{2} \rfloor$ векторов. И хотя компоненты этих векторов берутся уже не из множества $\{0,1\}$, а из более широкого множества $\{-1,0,1\}$, это не влияет на применяемый ими алгоритм решения задачи БВ, поскольку в последнем используется лишь свойство $|x_j^i| \le 1$.

Применяя этот трюк для получения соотношений между экстремальными функциями различных вариантов задачи БВ, получаем:

$$\bar{\eta}^C(n,m) \le \bar{\eta}^A\left(\left\lfloor \frac{n+1}{2} \right\rfloor, m\right), \tag{20.11}$$

$$\bar{\eta}^B(n,m) \le \bar{\eta}\left(\left\lfloor \frac{n+1}{2} \right\rfloor, m\right). \tag{20.12}$$

В дополнение к указанным выше имеем очевидные соотношения:

$$\eta^A(n,m) \le \eta(n,m),\ \eta^C(n,m) \le \eta^B(n,m), \tag{20.13}$$

$$\bar{\eta}^A(n,m) \le \bar{\eta}(n,m),\ \bar{\eta}^C(n,m) \le \bar{\eta}^B(n,m). \tag{20.14}$$

Аналогичные соотношения имеют место для функций $\eta^Z(m)$ и $\bar{\eta}^Z(m)$. Комбинируя (20.11) и (20.12) с (20.14), получаем

$$\bar{\eta}^C(n,m) \le \left\{ \bar{\eta}^B(n,m),\ \bar{\eta}^A\left(\left\lfloor \frac{n+1}{2} \right\rfloor, m\right)\right\} \le \bar{\eta}\left(\left\lfloor \frac{n+1}{2} \right\rfloor, m\right). \tag{20.15}$$

И если справедливо наше предположение

$$\bar{\eta}^A(n,m) = \bar{\eta}(n,m),$$

то для четырех величин в соотношении (20.15) получаем линейный порядок.

5. (Бал(2) $\Leftrightarrow$ Эк) Покажем, что функционалы этих задач удовлетворяют двусторонним соотношениям. Пусть $F = \{F_1, \ldots, F_m\} \in \mathcal{F}_{n,m},\ X \subseteq \mathbb{N}_n,\ \bar{X} = \mathbb{N}_n \setminus X,\ \bar{F}_i = \mathbb{N}_n \setminus F_i$, и пусть $x, f_i \in B^n$ — характеристические векторы множеств X и F_i $(i \in \mathbb{N}_m)$. Тогда

$$\big||F_i \cap X| - |F_i \cap \bar{X}|\big| = \big|(|F_i \cap X| + |\bar{F}_i \cap X|) - (|F_i \cap \bar{X}| + |\bar{F}_i \cap X|)\big| = |\rho(x,0) - \rho(x,f_i)|.$$

Таким образом, для решения задачи Бал(2) мы можем использовать ее сведение к задаче Эк с входным семейством векторов $F = \{0, f_1, \ldots, f_m\} \subseteq B^n$. Решив задачу Эк с априорной оценкой $\omega_F(x) \le \theta(n, m+1)$, получаем решение $(X, \bar{X})$ задачи Бал(2) с гарантированной оценкой $q_F(X) \le \omega_F(x) \le \theta(n, m+1)$, откуда следует соотношение

$$f(n,m) \le \theta(n, m+1).$$

С другой стороны, если задано семейство $F = \{f_1, \ldots, f_m\}$ вершин единичного куба B^n, мы можем поместить одну из вершин (скажем, вершину f_1) в начало координат, а затем найти вершину $x \in B^n$ (соответствующую некоторому подмножеству $X \subseteq \mathbb{N}_n$), такую что для любых $i \ne 1, j \ne 1$ выполняется

$$|\rho(x,f_i) - \rho(x,f_j)| \le |\rho(x,f_i) - \rho(x,0)| + |\rho(x,f_j) - \rho(x,0)|$$

$$= \big||F_i \cap X| - |F_i \cap \bar{X}|\big| + \big||F_j \cap X| - |F_j \cap \bar{X}|\big| \le 2f(n, m-1),$$

откуда $\theta(n,m) \le 2f(n,m-1)$. Таким образом, получаем

$$f(n,m-1) \le \theta(n,m) \le 2f(n,m-1). \tag{20.16}$$

6. (Im $\Leftrightarrow$ БВ$_C$) В задаче Im мы можем предполагать, что $\alpha_i \in [0,1],\ \bar{\alpha}_i \in \{0,1\},\ (i \in \mathbb{N}_n)$. Для заданного семейства подмножеств $F = \{F_1, \ldots, F_m\} \in \mathcal{F}_{n,m}$ определим семейство векторов $X = \{x_1, \ldots, x_n\} \subset B^m$ согласно (20.8). Тогда

$$\chi_{F,\alpha}(\bar{\alpha}) = \max_{F_k \in F} \left| \sum_{i \in F_k} \alpha_i - \sum_{i \in F_k} \bar{\alpha}_i \right| = \max_{k \in \mathbb{N}_m} \left| \sum_{i \in \mathbb{N}_n} \alpha_i x_k^i - \sum_{i \in \mathbb{N}_n} \bar{\alpha}_i x_k^i \right|$$

$$= \left\| \sum_{i \in \mathbb{N}_n} \alpha_i x_i - \sum_{i \in \mathbb{N}_n} \bar{\alpha}_i x_i \right\|_{l_\infty} = \|x - y\|_{l_\infty} = \xi^{l_\infty}_{X,x}(N'),$$

где $x = \sum_{i \in \mathbb{N}_n} \alpha_i x_i$ — заданная точка параллелепипеда $P(X)$, построенного на векторах $x_i \in X$, $y \doteq \sum_{i \in \mathbb{N}_n} \bar{\alpha}_i x_i$ — вершина параллелепипеда, $N' = \{i \in \mathbb{N}_n \,|\, \bar{\alpha}_i = 1\}$. Таким образом, задача Integer making эквивалентна задаче БВ$_C$, откуда вытекает соотношение между экстремальными функциями:

$$g(n,m) = \eta^C(n,m). \tag{20.17}$$

7. (РРП $\Leftrightarrow$ БВ) В п.1 уже было установлено, что задача $\overline{\text{БВ}}$ (когда отыскивается вершина параллелепипеда, ближайшая к его центру) эквивалентна частному случаю

задачи РРП, когда множество предметов делится на две кучи. Теперь заметим, что в разделе 18 для приближенного решения общего случая задачи РРП мы использовали ее сведение к задаче БВ. При этом в теореме 18.3 гарантировалось нахождение разбиения H в задаче РРП(r) с оценкой

$$\zeta_X^s(H) \le \nu(r)\,\xi_{X,x}^s(N'), \tag{20.18}$$

где функция $\nu(r)$ удовлетворяет оценке

$$\nu(r) < 2.00046,\ \forall\ r. \tag{20.19}$$

Из (20.18) вытекает соотношение между экстремальными функциями обеих задач:

$$\gamma_s(r,m) \le \nu(r)\,\eta_s(m),\ \forall\ r,m. \tag{20.20}$$

8. (Бал $\Leftrightarrow$ Im) Бек и Фиала [76] установили соотношение между экстремальными функциями задач Балансировки и Integer Making:

$$\psi(r,m) < \left(12 - \frac{4}{r-1}\right) g(m). \tag{20.21}$$

Покажем, что более точные соотношения между этими функциями могут быть получены с использованием обнаруженных нами "родственных связей" между задачами.

Лемма 20.1 *Для любых r, m справедливы соотношения:*

$$\psi(r,m) < 4.001\,g(m); \tag{20.22}$$

$$\psi(2,m) \le 2g(m).$$

Доказательство.

$$\psi(r,m) \overset{\text{из (20.9)}}{\le} 2\gamma^C(r,m) \overset{\text{из (20.20)}}{\le} 2\nu(r)\,\eta^C(m) \overset{\text{из (20.19) и (20.17)}}{<} 4.001\,g(m).$$

При $r = 2$ с учетом $\nu(2) = 1$ получаем требуемое соотношение:

$$\psi(2,m) \le 2g(m). \qquad \square$$

Оценки экстремальных функций

Из предыдущего раздела нетрудно заметить, что задача БВ является как бы связующим звеном в семействе рассматриваемых здесь комбинаторных задач. Экстремальные функции этой задачи (для различных вариантов исходных данных) могут быть использованы для вывода верхних и нижних оценок экстремальных функций других задач. Ввиду этого обстоятельства получение оценок функций $\eta^Z(n,m), \eta(m), \bar\eta^Z(n,m), \bar\eta(m)$ приобретает дополнительный смысл. Ниже будут сформулированы и доказаны верхние и нижние оценки этих функций.

1. Для произвольной нормы s из леммы 10.2 получаем оценку

$$\eta_s(n,m) \le \frac{1}{2}\min\{n,m\},$$

которая точна для нормы $s = l_1$. Кроме того, из леммы 10.2 вытекают соотношения

$$\eta_s(m) = \sup_n \eta_s(n,m) = \eta_s(m,m).$$

В случае нормы $s = l_\infty$ из теоремы 10.13 получаем оценку

$$\eta(n,m) \le \begin{cases} k(n,m)\sqrt{m\ln 2m}, & \text{для всех } m; \\ 0.735\cdot k(n,m)\sqrt{m\ln 2m}, & \text{для всех } m \ge 1500; \end{cases}$$

где $k(n,m)$ определяется по формуле (10.27) и удовлетворяет оценке $k(n,m) \le 2,\ \forall\, n,m$. Отсюда

$$\eta(m) \le \begin{cases} 2\sqrt{m\ln 2m}, & \text{для всех } m; \\ 1.47\sqrt{m\ln 2m}, & \text{для всех } m \ge 1500; \end{cases}$$

Для задачи $\overline{\text{ББ}}_{m,l_\infty}(X,x)$ (когда точка x есть центр параллелепипеда $P(X)$) имеется возможность вдвое улучшить оценку (10.19), получаемую в лемме 10.12, поскольку величины w_i в этом случае удовлетворяют соотношениям $w_i = \pm\frac{1}{2}$ (против более общего соотношения $w_i \in [-1,1]$, которое мы использовали в лемме 10.12). Это позволяет выписать оценки

$$\bar\eta(n,m) \le \begin{cases} 0.5\cdot k(n,m)\sqrt{m\ln 2m}, & \text{для всех } m; \\ 0.37\cdot k(n,m)\sqrt{m\ln 2m}, & \text{для всех } m \ge 1500; \end{cases}$$

$$\bar\eta(m) \le \begin{cases} \sqrt{m\ln 2m}, & \text{для всех } m; \\ 0.735\cdot\sqrt{m\ln 2m}, & \text{для всех } m \ge 1500. \end{cases}$$

2. Олсон и Спенсер [101] доказали оценку

$$f(m) > \left(\frac{1}{2} - o(1)\right)\ \sqrt{m},$$

откуда ввиду $2\bar\eta^C(m) = f(m)$ следует оценка

$$\bar\eta^C(m) > \left(\frac{1}{4} - o(1)\right)\ \sqrt{m}.$$

Далее будут получены оценки функций $\bar\eta^A(k,m)$ и $\bar\eta^A(m)$.

Лемма 20.2 *Пусть для некоторого $n \le m$ существует матрица Адамара A_n. Тогда*

$$\bar\eta^A(k,m) \ge \frac{1}{2}\sqrt{k},\ \forall\ k \le n. \tag{20.23}$$

Доказательство. В качестве "плохого" семейства векторов $X = \{x_1, \ldots, x_k\} \in \{-1, 0, 1\}^m$ берем k произвольных строк матрицы A_n, где недостающие $(m-n)$ координат каждого вектора достроены нулями. Тогда $(x_i, x_i) = n,\ \forall\ i$ и $(x_i, x_j) = 0,\ \forall\ i \neq j$. Имеем оценку

$$2\bar{\eta}^A(k,m) \geq \min_{\varepsilon_i = \pm 1} \left\| \sum_{i=1}^{k} \varepsilon_i x_i \right\|_{l_\infty}.$$

При любом выборе коэффициентов $\varepsilon_i = \pm 1$ для вектора $\sigma = \sum_{i=1}^{k} \varepsilon_i x_i$ выполняется $(\sigma, \sigma) = \sigma_1^2 + \ldots + \sigma_n^2 = kn$. Отсюда $\|\sigma\|_{l_\infty} = \max_i |\sigma_i| \geq \sqrt{k}$, и оценка (20.23) выполнена. □

Из доказанной леммы и факта плотности матриц Адамара на целочисленной решетке вытекает

Следствие 20.3 $\bar{\eta}^A(m) \geq \left(\frac{1}{2} - o(1)\right)\sqrt{m}$. □

21 Заключение (открытые вопросы и гипотезы)

Из последнего раздела данной части мы пришли к выводу, что задача о ближайшей вершине играет ключевую роль в семействе рассматриваемых здесь комбинаторных задач. В связи с этим исследование задачи БВ приобретает особый интерес. Однако на сегодня остается еще много открытых вопросов, часть из которых будут сформулированы ниже.

1. Верно ли, что для любых m и s выполняется

$$\eta_s(m,m) = \bar{\eta}_s(m,m),$$

т.е. центр "наихудшего" параллелепипеда является его "наихудшей" точкой, наиболее удаленной от ближайшей вершины? (Это верно для норм l_1 и l_2.)

2. Верно ли, что для любых m и s выполняется

$$\bar{\eta}_s(m,m) \geq \bar{\eta}_{l_2}(m,m),$$

т.е. евклидова норма является "наилучшей" нормой для задачи $\overline{\text{БВ}}_{m,s}$?

3. Верно ли, что для любых n и m выполняется

$$\bar{\eta}(n,m) = \bar{\eta}^A(n,m),$$

т.е. "наихудшее" l_∞-семейство векторов для задачи $\overline{\text{БВ}}_{m,l_\infty}$ имеет координаты из множества
$\{-1, 0, 1\}$?

Из положительного ответа на первые два вопроса будет следовать нетривиальная нижняя оценка функции Дворецкого:

$$D_s(m) \geq \sqrt{m},$$

откуда с учетом оценки (20.6) удастся доказать неограниченный рост $\inf_s \varphi_s(m)$ (см. проблему N 2 из заключения к первой части).

Часть III

Многостадийные системы теории расписаний

22 Общие определения и предварительные замечания

1. Рассматриваемые далее модели обслуживающих систем будут представлять собой частные случаи следующей общей модели.

Модель G. Задана многостадийная обслуживающая система, состоящая из конечного множества *работ* $\mathcal{J} = \{J_1, \ldots, J_n\}$, для выполнения которых имеется конечное множество *машин* $\mathcal{M} = \{M_1, \ldots, M_m\}$. Каждая работа $J_j \in \mathcal{J}$ состоит из множества *операций*: $O^j = \{o_1^j, \ldots, o_{r_j}^j\}$. (Иначе говоря, процесс выполнения каждой работы состоит из нескольких стадий, откуда и проистекает название "многостадийные".) На каждом множестве операций $O^j,\ J_j \in \mathcal{J}$, заданы отношения предшествования и несовместности в виде редуцированного смешанного графа $G_j = (O^j; U_j, E_j)$, где O^j — множество вершин, U_j — множество дуг, E_j — множество ребер. Наличие дуги $(o', o'') \in U_j$ в графе G_j означает, что операция o'' не может начаться раньше, чем закончится операция o'. (Это отношение будем также записывать в виде $o' \to o''$.) Наличие ребра $(o', o'') \in E_j$ означает, что операции o', o'' не могут выполняться одновременно. Для каждой операции o_q^j задано множество допустимых исполнителей $\mathcal{M}_q^j \subseteq \mathcal{M}$; операция o_q^j может выполняться на любой машине из множества $\mathcal{M}_q^j$; при этом время ее выполнения p_q^j (длительность операции) не зависит от выбора исполнителя $M_i \in \mathcal{M}_q^j$. Также предполагаются выполненными следующие условия:

- все работы готовы к выполнению в начальный (нулевой) момент времени;
- все работы независимы (они не связаны отношениями предшествования и нет условий предпочтения в виде директивных сроков);
- все операции неразрывны;
- в каждый момент времени на каждой машине выполняется не более одной операции (*условие* $(*)$).

2. Опишем некоторые частные случаи модели G, представляющие самостоятельный интерес и играющие важную роль как в наших дальнейших исследованиях, так и в теории расписаний.

- Обслуживающая система, в которой операции любой работы $J_j \in \mathcal{J}$ могут выполняться в любом порядке, но никакие две операции $o', o'' \in O^j$ не могут выполняться одновременно (что выражается условиями $U_j = \emptyset,\ E_j = O^j \times O^j \setminus \{(o, o) \,|\, o \in O^j\},\ J_j \in \mathcal{J}$), называется *системой открытого типа*. Если дополнительно предполагается $r_j = m,\ |\mathcal{M}_q^j| \equiv 1$ и $\mathcal{M}_q^j \cap \mathcal{M}_p^j = \emptyset,\ \forall p, q, j$ (т.е. машина-исполнитель для каждой операции задана однозначно, и каждая работа имеет ровно по одной операции на каждой машине), то получаем систему *open shop* (OS). Системы открытого типа рассматриваются в главе 1.

- Если для каждой работы $J_j \in \mathcal{J}$ на множестве O^j задан линейный порядок (т.е. граф редукции G_j состоит из единственной цепи $o_1^j \to o_2^j \to \cdots \to o_{r_j}^j$), а исполнитель каждой операции o_q^j задан однозначно ($|\mathcal{M}_q^j| = 1$), то имеем систему *job shop* (JS). Легко видеть, что в системе job shop каждая работа $J_j \in \mathcal{J}$ имеет заранее заданный маршрут прохождения машин, и этот машрут может быть различным для разных работ. Системам job shop, а также системам с произвольными множествами U_j посвящена глава 4.

- Если $r_j \equiv r,\ \mathcal{M}_q^j \equiv \mathcal{M}_q,\ \forall J_j \in \mathcal{J};\ \mathcal{M}_q \cap \mathcal{M}_p = \emptyset$, при $q \neq p$, и на каждом множестве O^j задан линейный порядок:

$$o_1^j \to o_2^j \to \cdots \to o_r^j, \tag{22.1}$$

  то обслуживающая система называется *системой поточного типа*. В частном случае, когда $|\mathcal{M}_q| \equiv 1,\ \forall q$, и $r = m$, такая система известна под названием *flow shop* (коротко, FS). Таким образом, в системе flow shop для каждой работы задан маршрут ее прохождения по машинам, и этот маршрут $(M_1, M_2, \ldots, M_m)$ одинаков для всех работ. (Отсюда следует, что операция o_i^j выполняется на машине M_i.) Системы поточного типа будут изучаться в главе 2.

- Наконец, в главе 3 будет исследоваться система "линия сборки", которая отличается от системы flow shop тем, что вместо (22.1) для каждой работы $J_j \in \mathcal{J}$ устанавливается следующий порядок предшествования операций из множества O^j:

$$o_i^j \to o_m^j,\ i \in \mathbb{N}_{m-1}. \tag{22.2}$$

  Здесь операции $\{o_i^j \,|\, i \in \mathbb{N}_{m-1}\}$ интерпретируются как операции обработки отдельных деталей сложного прибора. Они выполняются независимо одна от другой и параллельно на разных машинах. Последняя операция работы J_j (сборка деталей прибора на машине M_m) может выполняться лишь после того, как готовы все детали этого прибора.

3. Процесс функционирования обслуживающей системы может быть описан путем задания расписания S, в котором указывается, когда и на каких машинах выполняется каждая операция o_q^j. Поскольку в рассматриваемых нами моделях все операции неразрывны, то каждая из них выполняется только на одной машине (обозначим ее $M(o_q^j)$) и только в одном непрерывном интервале времени $[s_q^j, s_q^j + p_q^j)$, однозначно задаваемом своим левым концом s_q^j (временем "старта" операции o_q^j в расписании S). Таким образом, расписание задается парой: множеством $\{s_q^j \,|\, q \in \mathbb{N}_{r_j};\ J_j \in \mathcal{J}\}$ неотрицательных чисел s_q^j и функцией $M : \mathcal{O} \to \mathcal{M}$, где $\mathcal{O} = \cup_j O^j$ — множество всех операций работ $J_j \in \mathcal{J}$; $M(o_q^j) \in \mathcal{M}_q^j,\ \forall o_q^j \in \mathcal{O}$. В случае, когда $|\mathcal{M}_q^j| \equiv 1,\ \forall j, q$, функция M однозначно определена условиями задачи, поэтому остается найти лишь множество $\{s_q^j\}$, которое в этом случае и будем называть расписанием.

Расписание называется *допустимым*, если оно удовлетворяет всем требованиям данной обслуживающей системы.

Для оценки качества расписаний во всех рассматриваемых системах будет использоваться целевая функция

$$C_{\max}(S) = \max_{j,q}(s_q^j + p_q^j),$$

называемая *длиной* расписания S. *Оптимальным расписанием* (S_{opt}) называется расписание, на котором достигается минимум функции $C_{\max}(S)$ на множестве всех допустимых расписаний. Расписание S_{opt} будем называть также расписанием *наименьшей длины*. Задачу нахождения оптимального расписания в системе flow shop, job shop, open shop, и т.д. будем называть, соответственно, задачей flow shop, job shop, и т.д. (Задачу flow shop называют также задачей Беллмана-Джонсона.)

4. Как известно [61], задачи flow shop и open shop являются *NP*-трудными при $m \geq 3$, а job shop — уже при $m = 2$ и $p_q^j \in \{1, 2\}$, поэтому для точного решения этих задач можно предложить лишь переборные алгоритмы. Поскольку нас интересуют только алгоритмы полиномиальной трудоемкости, это предопределяет ориентацию наших исследований в двух направлениях:

- поиск максимально широких подклассов примеров исследуемых задач, допускающих решение за полиномиальное время;
- построение полиномиальных алгоритмов приближенного решения этих задач с гарантированными оценками качества получаемых решений.

Чтобы кратко охарактеризовать результаты данной главы, введем несколько обозначений.

Если функция M в расписании S уже определена (либо однозначно задана условием задачи и не зависит от расписания), то для каждой машины $M_i \in \mathcal{M}$ мы можем определить множество операций, которые на ней выполняются: $F_i \doteq M^{-1}(M_i)$. Обозначим $L_i = \sum_{o_q^j \in F_i} p_q^j$; $L_{\max} = \max_{M_i \in \mathcal{M}} L_i$; $p_{\max} = \max_{j,q} p_q^j$; $r = \max_{J_j \in \mathcal{J}} r_j$.

5. Ясно, что с учетом условия $(*)$, машина $M_i \in \mathcal{M}$ работает не меньше L_i единиц времени, поэтому длина любого расписания S (в том числе, S_{opt}) ограничена снизу величиной $L_{\max}$. Как известно, при выводе оценок абсолютной и относительной погрешности алгоритмов в задачах на минимум используются нижние оценки оптимума. Полученная нами простая нижняя оценка

$$C_{\max}(S_{\text{opt}}) \geq L_{\max} \tag{22.3}$$

неожиданно оказывается довольно точной для рассматриваемых задач. Ее погрешность не превышает некоторой величины, не зависящей от числа работ n. Другими словами, для любого примера задачи P его оптимум содержится в известном заранее интервале $I_P = [L_{\max}, L_{\max} + \mu_P(m, r)p_{\max}]$, где функция μ_P зависит от числа машин и максимального числа операций одной работы и не зависит от n. Значение функции μ_P конкретизируется (и по возможности — минимизируется) для каждой задачи P. Так например, для общей задачи job shop найденная нами функция μ_{JS} значительно превосходит подобную функцию μ_{FS} для задачи flow shop, оставаясь, тем не менее, полиномом от m

и r. Для каждого из создаваемых нами приближенных алгоритмов будет доказываться гарантированное построение расписания S со значением $C_{\max}(S)$ в интервале I_P :

$$C_{\max}(S) \leq L_{\max} + \mu_P(m, r)p_{\max}. \tag{22.4}$$

Тем самым гарантируется оценка абсолютной погрешности получаемого решения:

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leq \mu_P(m, r)p_{\max}, \tag{22.5}$$

не зависящая от числа работ n. Из (22.3) и (22.5) можно получить оценку относительной погрешности расписания S:

$$\frac{C_{\max}(S) - C_{\max}(S_{\text{opt}})}{C_{\max}(S_{\text{opt}})} \leq \frac{\mu_P(m, r)p_{\max}}{L_{\max}}. \tag{22.6}$$

Для отдельных задач P будут найдены верхние и нижние оценки минимальной функции $\tilde{\mu}_P$, определяющей точную верхнюю границу интервала локализации оптимумов в классе примеров задачи P.

6. Нетривиальные полиномиально разрешимые подклассы примеров отыскиваются в главе 1 для задачи open shop. Для этого вводятся понятия нормального расписания и нормального примера. А именно, расписание называется *нормальным*, если его длина равна $L_{\max}$. Ввиду оценки (22.3), всякое нормальное расписание оптимально. Пример, обладающий нормальным расписанием, также называется *нормальным*.

Отыскиваются достаточные условия, при которых любой удовлетворяющий им пример задачи open shop является нормальным. Каждое такое условие описывает *класс нормальных примеров*, или *нормальный* класс. Поскольку, однако, мы заинтересованы также в отыскании *полиномиально разрешимых* подклассов, то вводится понятие *эффективно-нормального класса* примеров. А именно, класс примеров эффективно-нормален, если он нормален и существует полиномиальный алгоритм построения нормального (а следовательно, оптимального) расписания для любого примера из заданного класса.

Описываются три способа построения эффективно-нормальных классов примеров задачи open shop. В первом из этих способов, рассматриваемом в разделах 25, 26, каждый класс определяется условием вида:

$$L_{\max} \geq \eta_i(m)p_{\max} \tag{22.7}$$

для некоторой, достаточно большой функции η_i от числа машин. Исследуется два способа получения таких функций η_i. В первом, использующем метод компактного суммирования векторов, определяется функция $\eta_1(m)$ порядка $\sim m^2$ (см. раздел 25), в то время как функция $\eta_2(m)$, определяемая вторым способом (в разделе 26), имеет лучший порядок ($\eta_2(m) \sim m \log m$), но проигрывает при относительно малых значениях m из-за сравнительно большой мультипликативной константы. Второй способ определяется более сложной процедурой построения нормального расписания, использующей алгоритм решения задачи о нахождении подмножества операций (из раздела 10) и результаты

о свойствах рангов целых чисел (из раздела 17). Поскольку в обоих случаях в явном виде представлены полиномиальные алгоритмы построения нормальных расписаний, то обе функции $\eta_1(m), \eta_2(m)$ являются верхними оценками наименьшей функции $\eta_{en}(m)$, для которой условие (22.7) определяет эффективно-нормальный класс примеров задачи open shop. Определяются также минимальные функции $\eta_n(m)$ и $\eta_e(m)$, для которых условие (22.7) определяет, соответственно, нормальный и полиномиально-разрешимый классы примеров задачи open shop. Устанавливаются нижние оценки этих функций. (Нетрудно понять, что $\eta_{en}(m) = \max\{\eta_e(m), \eta_n(m)\},\ \forall m$.)

Другой способ построения эффективно-нормальных классов определяется на основе задания условий на попарные разности нагрузок машин. С этой целью вводится понятие *вектора разностей* нагрузок машин (VOD). Решается задача отыскания минимальных векторов разностей, определяющих эффективно нормальные классы.

Наконец, третий способ является комбинацией первых двух.

Из вида ограничений (22.7) легко заметить, что при возрастании числа работ n доля примеров, удовлетворяющих этому условию, стремится к единице, иначе говоря, **почти все** примеры являются нормальными. Таким образом, свойство нормальности является "нормой" в классе всех примеров задачи open shop.

7. Как было отмечено в предыдущем пункте, для довольно широкого спектра примеров задачи open shop их оптимум в точности равен $L_{\max}$. Оказывается, что и во всех остальных случаях он лишь незначительно отклоняется от величины $L_{\max}$. Как было показано Анной Рачмань (см. [74]), простейший жадный алгоритм линейной от n трудоемкости гарантирует абсолютную оценку

$$C_{\max}(S) \leq L_{\max} + (m-1)p_{\max}. \tag{22.8}$$

В разделе 30 приводится алгоритм такой же трудоемкости, который уточняет оценку (22.8) в зависимости от значения величины $L_{\max}$. Так например, при $L_{\max} \geq 7mp_{\max}$ алгоритм гарантирует оценку $C_{\max}(S) \leq L_{\max} + (m-1)p_{\max}/3$.

8. При доказательстве теорем нам часто будет удобно предполагать выполненным свойство

$$L_i \equiv L_{\max},\ \forall M_i \in \mathcal{M}. \tag{22.9}$$

В случае, когда на всех машинах M_i выполняется одинаковое количество операций (например, в задачах flow shop, open shop и Акерса-Фридман, где $|F_i| \equiv n,\ \forall M_i \in \mathcal{M}$), нетрудно добиться выполнения (22.9) при тех же значениях $m, n, L_{\max}$ и $p_{\max}$ путем увеличения длительностей некоторых операций на "недогруженных" машинах. (Трудоемкость процедуры выравнивания нагрузки по всем машинам не превосходит $O(nm)$.) Построив расписание для полученного входа задачи (удовлетворяющего теперь (22.9)) и получив для него оценку вида (22.4), мы вернемся к исходным длительностям. При этом построенное расписание $S = \{s_i^j\}$ остается допустимым, а полученная оценка вида (22.4) остается справедливой.

Для удобства выкладок мы будем часто опускать множитель $p_{\max}$, предполагая выполненным равенство

$$p_{\max} = 1. \tag{22.10}$$

Это соответствует выбору $p_{\max}$ в качестве новой единицы времени. При этом длительности всех операций принимают значения из интервала $[0,1]$. (Очевидно, что случай $p_{ji} \equiv 0,\ \forall j,i,$ — когда выбор $p_{\max}$ в качестве единицы невозможен, — не представляет интереса.)

9. Для построения расписаний в рассматриваемых ниже многостадийных системах типа flow shop, job shop и open shop мы будем часто применять так называемые "жадные" алгоритмы, определяемые следующим образом.

Пусть исполнитель каждой операции $o_i^j \in \mathcal{O}$ определен функцией $M : \mathcal{O} \to \mathcal{M}$, т.е. для каждой машины $M_i \in \mathcal{M}$ известно множество ее операций $F_i = M^{-1}(M_i)$, и на нем в каждый момент времени задан некоторый порядок ($\succ_p$), который будем называть "приоритетным". (Отношение $o' \succ_p o''$ означает больший приоритет операции o' по отношению к o''.) Алгоритм построения расписания будем называть "жадным"[1], если он работает на следующих принципах.

1. Машина не простаивает, если есть работа, готовая на ней выполняться.

2. Выбор очередной операции для освободившейся машины (M_i) определяется текущим приоритетным порядком $\succ_p$ на множестве F_i.

В случае систем открытого типа, когда порядок выполнения операций каждой работы может быть произволен, для корректности определения жадного алгоритма необходимо к указанным двум принципам добавить еще один.

3. Выбор очередной операции освободившейся работы J_j определяется приоритетным порядком на множестве O^j.

Расписание, получаемое жадным алгоритмом, также будем называть "жадным". (Иногда его называют "плотным", "ранним", и т.п.)

Приоритетной укладкой (S^p) назовем такое (вообще говоря недопустимое) расписание, в котором операции каждой машины M_i выполняются подряд, без простоев, начиная с нулевого момента времени, в соответствии с заданным на множестве F_i приоритетным порядком $\succ_p$. Таким образом, время старта $\hat{s}(o')$ операции $o' \in F_i$ в укладке S^p определяется по формуле:

$$\hat{s}(o') = \sum_{\{o_q^j \in F_i \mid\ o_q^j \succ_p o'\}} p_q^j.$$

Ясно, что в тех случаях, когда укладка задает допустимое расписание, оно является жадным для заданных приоритетов на множествах F_i.

[1] Мы называем наш алгоритм "жадным" вслед за Лолэ и др. [96], скорее отдавая дань традиции, чем правильности использования этого термина. Более правильно относить термин "жадный" к тем алгоритмам, которые на каждом шаге стремятся "откусить кусок побольше", т.е. — к градиентным алгоритмам. В нашем же случае алгоритм "кусает" первый попавшийся кусок, поэтому его лучше называть "голодным". Так например, в этой "гастрономической" терминологии алгоритм Best Fit будет отнесен к "жадным", а First Fit — к "голодным".

10. Во многих рассматриваемых ниже системах (таких как open shop, flow shop, сборочная линия, система Акерса-Фридман и ее частные случаи) для любой работы $J_j \in \mathcal{J}$ и машины $M_i \in \mathcal{M}$ однозначно определена операция работы J_j на машине M_i. В этих случаях o_{ji} будет обозначать операцию работы J_j на машине M_i, p_{ji} — ее длительность, s_{ji}, c_{ji} — моменты начала и окончания операции o_{ji} в расписании S. Поскольку в указанных случаях на каждой машине выполняется ровно n операций, то мы можем предполагать выполненными соотношения (22.9). Далее для каждой работы $J_j \in \mathcal{J}$ определим вектор

$$d_j = (d_j(1), \dots, d_j(m-1)) \doteq (p_{j1} - p_{jm}, \dots, p_{j,m-1} - p_{jm}) \in \mathbb{R}^{m-1}. \tag{22.11}$$

Тогда из (22.9) имеем $\sum_{j=1}^{n} d_j = 0$. Кроме того, из определения векторов d_j имеем свойства

$$|d_j(i)| \le p_{\max},\ |d_j(i_1) - d_j(i_2)| \le p_{\max},\ \forall\ i, i_1, i_2 \in \mathbb{N}_{m-1}.$$

Таким образом, принимая $p_{\max}$ за новую единицу времени, получаем $p_{\max} = 1$, и семейство векторов $D = \{d_1, \dots, d_n\}$ становится $\hat{s}$-семейством, согласно определению нормы $\hat{s}$, представленному в разделе 14 (стр. 107).

Для заданной перестановки $\pi = (\pi_1, \dots, \pi_n)$ введем следующие обозначения для частичных сумм скаляров p_{ji} и векторов d_j, просуммированных по j в порядке перестановки π:

$$P_\pi^k(i) \doteq \sum_{j=1}^{k} p_{\pi_j i},\ d_\pi^k \doteq \sum_{j=1}^{k} d_{\pi_j}. \tag{22.12}$$

Пусть $\{e_1, \dots, e_{m-1}\}$ обозначает множество единичных базисных векторов в пространстве $\mathbb{R}^{m-1}$. Тогда для любых $i_1, i_2 \in \mathbb{N}_m$ и $k \in \mathbb{N}_n$ величина

$$P_\pi^k(i_1, i_2) \doteq P_\pi^k(i_1) - P_\pi^{k-1}(i_2) \tag{22.13}$$

может быть оценена сверху следующим образом:

$$P_\pi^k(i_1, i_2) = P_\pi^{k-1}(i_1) - P_\pi^{k-1}(i_2) + p_{\pi_k i_1} = P_\pi^k(i_1) - P_\pi^k(i_2) + p_{\pi_k i_2}$$
$$\le \min\{P_\pi^{k-1}(i_1) - P_\pi^{k-1}(i_2), P_\pi^k(i_1) - P_\pi^k(i_2)\} + p_{\max} =$$

(используя определение векторов d_π^k и полагая $d_\pi^k(m) = 0,\ \forall k$)

$$= p_{\max} + \min\{d_\pi^{k-1}(i_1) - d_\pi^{k-1}(i_2), d_\pi^k(i_1) - d_\pi^k(i_2)\}$$
$$= p_{\max} + \min\{(e_{i_1} - e_{i_2}, d_\pi^{k-1}), (e_{i_1} - e_{i_2}, d_\pi^k)\}. \tag{22.14}$$

(В этой формуле полагаем $e_m = 0$.) Используя эту оценку величины $P_\pi^k(i_1, i_2)$, мы можем сформулировать следующее

Замечание 22.1 *Пусть $D = \{d_1, \dots, d_n\}$ — семейство векторов в $\mathbb{R}^{m-1}$, определенных согласно (22.11), и пусть перестановка $\pi = (\pi_1, \dots, \pi_n)$ задает нестрогое суммирование векторов из D в полупространстве $P(e_{i_1} - e_{i_2}, \beta)$ для некоторых $i_1, i_2 \in \mathbb{N}_m,\ i_1 \ne i_2,\ \beta \in \mathbb{R}$, где $e_m = 0$. Тогда справедливы оценки*

$$P_\pi^k(i_1, i_2) \le p_{\max} + \beta,\ k \in \mathbb{N}_n.$$

□

$$\begin{array}{ccccccc}
o_1^{\pi_1} & \to & o_1^{\pi_2} & \to & \cdots & \to & o_1^{\pi_n} \\
\downarrow & & \downarrow & & & & \downarrow \\
o_2^{\pi_1} & \to & o_2^{\pi_2} & \to & \cdots & \to & o_2^{\pi_n} \\
\downarrow & & \downarrow & & & & \downarrow \\
\vdots & & \vdots & & & & \vdots \\
\downarrow & & \downarrow & & & & \downarrow \\
o_m^{\pi_1} & \to & o_m^{\pi_2} & \to & \cdots & \to & o_m^{\pi_n}
\end{array}$$

Рис. 22.1: Сеть $G_{FS}(\pi)$.

11. В разделе 2, посвященном задаче flow shop, а также при рассмотрении задачи о сборочной линии (*СЛ*) в разделе 3 мы будем иметь дело с так называемыми *перестановочными расписаниями*, в которых порядок выполнения работ одинаков для всех машин и задается единственной перестановкой номеров работ: $\pi = (\pi_1, \ldots, \pi_n)$. Для заданной перестановки π через S_π будем обозначать наиболее раннее перестановочное расписание с порядком работ π. "Перестановочность" расписания S_π задает ограничения предшествования на множестве F_i операций каждой машины M_i :

$$o_i^{\pi_1} \to o_i^{\pi_2} \to \ldots \to o_i^{\pi_n}, \ M_i \in \mathcal{M}. \tag{22.15}$$

Для задачи flow shop соотношения (22.15) и (22.1) определяют сеть $G_{FS}(\pi)$ ограничений предшествования на множестве $\mathcal{O}$ всех операций (см. рис. 22.1).

Из сетевого планирования известно, что длина раннего расписания, задаваемого сетью G, равна длине критического пути в сети G, т.е. сумме длительностей лежащих на нем вершин-операций. С учетом вида сети $G_{FS}(\pi)$ это приводит нас к известной формуле длины перестановочного расписания (S_π) задачи flow shop:

$$C_{\max}(S_\pi) = \max_{1 \le k_1 \le k_2 \le \cdots \le k_{m-1} \le n} \left(\sum_{j=1}^{k_1} p_{\pi_j 1} + \sum_{j=k_1}^{k_2} p_{\pi_j 2} + \cdots + \sum_{j=k_{m-1}}^{n} p_{\pi_j m} \right). \tag{22.16}$$

Для задачи о сборочной линии сеть $G = G_{\text{СЛ}}(\pi)$ формируется из отношений (22.15) и (22.2), поэтому формула длины ее перестановочного расписания выглядит несколько иначе:

$$C_{\max}(S_\pi) = \max_{i \in \mathbb{N}_{m-1}} \ \max_{k \in \mathbb{N}_n} \left(\sum_{j=1}^{k} p_{\pi_j i} + \sum_{j=k}^{n} p_{\pi_j m} \right). \tag{22.17}$$

Глава 1

Системы с нефиксированными маршрутами

В этой главе будет представлен комплесный анализ систем открытого типа (иначе говоря, систем с нефиксированными маршрутами), определенных на стр.157. Нашим основным объектом исследований будет классическая задача $O||C_{\max}$ (или коротко, задача OS). Более общие системы открытого типа, допускающие неоднозначный выбор исполнителей для каждой операции и произвольное, не зависящее от числа машин количество операций, рассматриваются в разделах 31, 35 и 37. Поскольку уже простейшая задача $O3||C_{\max}$ является NP-трудной, акцент в исследованиях смещается как на отыскание максимально широких подклассов полиномиально разрешимых примеров данной задачи, так и на построение приближенных алгоритмов с гарантированными оценками точности. Основным понятием, используемым при получении результатов первого типа, являются так называемые *нормальные* расписания, т.е. допустимые расписания длины $L_{\max}$. Ясно, что в тех случаях, когда удается эффективно построить такое нормальное расписание, мы получаем эффективный алгоритм точного решения нашей задачи. В разделах 24–27, 30, 34 исследуется, при каких достаточных условиях на входные данные задачи заданный пример обладает нормальным расписанием, а в каких случаях — ответ на вопрос о существовании такого расписания представляется NP-трудной проблемой.

Найдены три типа условий, определяющих нормальность того или иного примера. Условие первого типа задает нижнюю границу на $L_{\max}$ (в терминах произведения $p_{\max}$ на функцию от числа машин), начиная с которой любой пример, удовлетворяющий этому условию, обладает нормальным расписанием. В разделах 25–26 найдены два различных условия первого типа и, соответственно, два алгоритма точного решения задачи при выполнении этих условий. В разделе 27 исследуется, когда условие первого типа недостаточно для существования нормального расписания, а также — когда проверка существования такого расписания становится NP-трудной проблемой.

Условие второго типа предусматривает определенную систему ограничений на попарные разности нагрузок машин. С этой целью вводится понятие *вектора разностей* нагрузок машин (*VOD*). В разделах 24, 34 отыскиваются минимально возможные век-

торы разностей, для которых любой пример с данным *VOD* является нормальным либо эффективно нормальным. В последнем случае мы одновременно получаем максимально широкий подкласс эффективно разрешимых примеров задачи OS.

Наконец, условие третьего типа является комбинацией первых двух. Условие такого типа, гарантирующее эффективное построение нормального расписания, найдено в разделе 30.

Одним из основных инструментов, используемых как при построении нормальных (оптимальных) расписаний, так и для построения расписаний, близких к оптимальным (в разделах 30 и 35) являются так называемые *жадные* алгоритмы. Так в разделе 28 описывается общая схема ($\mathcal{A}_{\mathrm{GP}}$) жадного алгоритма с заданными приоритетными порядками операций на машинах, в разделе 29 исследуются общие свойства получаемых такими алгоритмами расписаний, а в разделе 30 мы приводим два конкретных жадных алгоритма с различными правилами построения приоритетных порядков на машинах.

Жадные алгоритмы другого типа описываются в разделах 32 и 35. Схема их реализации ($\mathcal{A}_{\mathrm{GC}}$) несколько отличается от схемы $\mathcal{A}_{\mathrm{GP}}$. Отличие обусловлено тем, что схема $\mathcal{A}_{\mathrm{GC}}$ ориентирована на достройку (жадным способом) некоторого уже имеющегося расписания. В разделе 32 алгоритмы типа $\mathcal{A}_{\mathrm{GC}}$ используются для построения нормальных расписаний, в то время как в разделе 35 алгоритм $\mathcal{A}_{\mathrm{GC}}$ играет вспомогательную роль — для расписывания средних и малых работ в аппроксимационной схеме.

Несколько слов по поводу последней. Если для какой-то оптимизационной задачи установлена ее NP-трудность (а таковыми является большинство задач, рассматриваемых в дискретной оптимизации), то следующим шагом в анализе этой задачи должно быть исследование ее на “приближаемость”. Прежде всего, это — выяснение вопроса, существует ли для нее полиномиальная аппроксимационная схема (PTAS), т.е. возможность при любом заданном ε построить полиномиальный по трудоемкости ε-оптимальный алгоритм ее решения. Если такой схемы не существует, то — существует ли константа C, для которой возможен эффективный алгоритм решения задачи с относительной погрешностью C? Какова минимальная такая константа? И т.д., т.е. исследуются предельные возможности для приближенного решения задачи. Долгое время вопрос о существовании PTAS для задачи OS оставался открытым, и наши знания о приближаемости задачи ограничивались наличием нескольких C-приближающих алгоритмов ([82], [74], [2]). В разделе 35 диссертации дается положительный ответ на вопрос о существовании PTAS в случае фиксированного числа машин. Более того, схема обобщена на случай многопроцессорной задачи OS, когда число цехов и число машин в каждом цехе ограничены константой. Особо ценным является факт “сравнительно небольшой” (по сравнению с известными схемами — для других задач) трудоемкости схемы — она линейна от числа работ (n), причем мультипликативная константа, стоящая в $O(n)$, не зависит от ε. Таким образом, от “полностью полиномиальной” схемы ее отделяет лишь аддитивная константа, зависящая от ε (правда, очень большая). Отрицательный же результат содержится в разделе 36, где доказывается, что при нефиксированном числе машин невозможно построение эффективного приближенного алгоритма решения задачи OS с относительной погрешностью меньшей 5/4, если $P \neq NP$. (А следовательно, не существует и аппроксимационной схемы.)

Поскольку во всем разделе (за исключением подразделов 35 и 37, где исследуются многопроцессорные задачи) мы будем рассматривать классическую задачу open shop, в которой для любой работы $J_j \in \mathcal{J}$ и любой машины $M_i \in \mathcal{M}$ существует единственная операция работы J_j на машине M_i, то для удобства будем предполагать, что операция o_q^j выполняется машиной M_q.

23 Понятие нормальности в системах open shop

Определение 23.1 Допустимое расписание длины $L_{\max}$ называется *нормальным*. Пример задачи open shop, обладающий нормальным расписанием, называется *нормальным*. Подмножество нормальных примеров называется *нормальным классом*. Если существует алгоритм полиномиальной трудоемкости, который для любого примера из заданного нормального класса находит его нормальное (а следовательно, оптимальное) расписание, то нормальный класс называется *эффективно-нормальным классом*.

Как будет видно из результатов данной главы, "нормальность" типична для большинства примеров задачи open shop. Например, при фиксированном числе машин (m), произвольном числе работ (n) и случайном независимом выборе длительностей операций ($p_i^j \geq 0$) полученный случайный пример почти всегда будет удовлетворять неравенству

$$\max_{i=1,\ldots,m} \sum_{j=1}^{n} p_i^j \geq \eta(m) \cdot \max_{j,i} p_i^j, \tag{23.1}$$

что при надлежащем выборе функции $\eta(m)$ (например, $\eta(m) = m^2 - 1 + \frac{1}{m-1}$ — в разделе 25, и $\eta(m) = \frac{16}{3}\, m \log_2 m + \frac{61}{9}\, m - 7.4$ — в разделе 26) обеспечивает нормальность примера. (Вероятность невыполнения (23.1) при любом фиксированном m, а следовательно — вероятность "ненормальности" примера стремится к нулю с ростом n, — что выполняется для любой функции распределения величин $\{p_i^j\}$, обладающей ненулевым конечным матожиданием.) Для сравнения: существование нормального расписания для систем типа job shop — явление достаточно редкое и не сулящее открытия широких полиномиально разрешимых подклассов для этих задач.

Далее будем предполагать масштабированность всех рассматриваемых примеров, с величиной $p_{\max}$, взятой за единицу. Кроме того, без ограничения общности можем предполагать, что машины занумерованы по невозрастанию их нагрузок:

$$L_{\max} = L_1 \geq L_2 \geq \cdots \geq L_m.$$

(Это соглашение несущественно для разделов 25, 26, где будет предполагаться равенство нагрузок всех машин.) Указанную выше нумерацию машин будем называть *нумерацией по умолчанию*. Для этой нумерации и любого заданного примера определяем *вектор разностей VOD* $= (\delta(1), \ldots, \delta(m))$, где $\delta(i) = L_{\max} - L_i,\ i \in \mathbb{N}_m$. Таким образом, всегда выполнены соотношения

$$0 = \delta(1) \leq \delta(2) \leq \cdots \leq \delta(m). \tag{23.2}$$

Определение 23.2 (нормализующий вектор, *EN*-вектор)
• Вектор $\Delta \in \mathbb{R}^m$ называется *нормализующим*, если $0 = \Delta(1) \le \Delta(2) \le \cdots \le \Delta(m)$, и любой пример с $VOD = \Delta$ является нормальным.
• Вектор Δ называется *эффективно нормализующим* (коротко, *EN*-вектором), если он нормализующий и существует полиномиальный алгоритм, который для любого примера с $VOD = \Delta$ находит его оптимальное (нормальное) расписание.

Простейшим примером нормализующего и эффективно нормализующего вектора является вектор $(0, 1) \in \mathbb{R}^2$. Действительно, известный полиномиальный алгоритм Гонзалеза и Сани [87] для любого примера задачи $O2||C_{\max}$ гарантирует построение расписания длины $\max\{L_{\max}, l_{\max}\}$, где $l_{\max} = \max_j l_j$, $l_j = p_{j1} + p_{j2}$ — длина работы $J_j \in \mathcal{J}$. Если для заданного примера выполнено $VOD = (0, 1)$, это означает, что $L_1 - L_2 = p_{\max}$, откуда для любой работы J_j имеем $l_j = p_{j1} + p_{j2} \le p_{\max} + L_2 = L_1 = L_{\max}$, и значит, $l_{\max} \le L_{\max}$, т.е. алгоритм Гонзалеза и Сани гарантирует эффективное построение **нормального** расписания для любого примера с $VOD = (0, 1)$.

Определим частичный порядок на множестве векторов в $\mathbb{R}^m$. Для $\Delta', \Delta'' \in \mathbb{R}^m$ будем писать $\Delta' \le \Delta''$, если

$$\Delta'(i) \le \Delta''(i), \ i \in \mathbb{N}_m, \tag{23.3}$$

и будем писать $\Delta' < \Delta''$, если $\Delta' \le \Delta''$ и хотя бы одно из неравенств (23.3) строгое.

Лемма 23.3 *Пусть $\Delta', \Delta'' \in \mathbb{R}^m$, Δ' — нормализующий (эффективно нормализующий) вектор и $\Delta'' \ge \Delta'$. Тогда Δ'' — также нормализующий (эффективно нормализующий).*

Доказательство. Пусть нам известен нормализующий (эффективно нормализующий) вектор Δ' и задан пример с $VOD = \Delta'' > \Delta'$. Мы можем увеличить нагрузки машин $M_2, \ldots, M_m$, увеличив длительности выполняемых на них операций, так чтобы VOD стал равным Δ', а значения $L_{\max}$ и $p_{\max}$ остались неизменными. Для полученного примера, согласно предположению, может быть построено расписание длины $L_{\max}$. Это расписание допустимо и для исходного примера (с более короткими операциями), имеющего то же значение $L_{\max}$. Следовательно, исходный пример является нормальным. Тем самым доказано, что любой пример с $VOD = \Delta''$ является нормальным, а это значит, что вектор Δ'' — нормализующий. Более того, если существует полиномиальный алгоритм, который строит оптимальное расписание для любого примера с $VOD = \Delta'$, то существует и алгоритм для класса примеров с $VOD = \Delta''$. □

Из леммы 23.3 следует, что имеет смысл поиск минимально возможных нормализующих (эффективно нормализующих) векторов, поскольку они задают максимально широкие нормальные (эффективно нормальные) классы примеров задачи OS.

Определение 23.4 (*MN*-вектор, *MEN*-вектор)
• Вектор Δ называется **минимальным нормализующим** вектором (коротко, *MN-вектором*), если он нормализующий и не существует нормализующего вектора $\Delta' < \Delta$.

• Вектор Δ называется **минимальным эффективно нормализующим** вектором (коротко, *MEN-вектором*), если он эффективно нормализующий и не существует эффективно нормализующего вектора $\Delta' < \Delta$.

Мы уже ознакомились с простейшим примером нормализующего вектора. Простейший нетривиальный пример ненормализующего вектора приводится в следующем утверждении.

Утверждение 23.5 *Для любого $\varepsilon > 0$, вектор $(0, m-1-\varepsilon, \ldots, m-1-\varepsilon) \in \mathbb{R}^m$ не является нормализующим.*

Доказательство. Для доказательства достаточно привести "ненормальный" пример с *VOD* , равным заданному вектору. В качестве такового возьмем пример, в котором одна работа имеет m операций единичной длины, и еще $(n-1)$ работ имеют ненулевые операции лишь на первой машине, образуя на ней суммарную нагрузку $L_1 = m - \varepsilon$. Тогда $VOD = (0, m-1-\varepsilon, \ldots, m-1-\varepsilon)$, и $C^*_{\max} = m > L_{\max}$, т.е. пример не является нормальным. □

Таким образом, любой вектор с последней компонентой меньшей $m-1$ не является нормализующим. С учетом этого утверждения при $m = 2$ и приведенного выше факта об эффективной нормальности вектора $(0, 1)$ получаем

Утверждение 23.6 *Вектор $(0, 1)$ является единственным MN-вектором и единственным MEN-вектором в пространстве $\mathbb{R}^2$.* □

В следующем разделе будут найдены все *MN*- и *MEN*-векторы в пространстве $\mathbb{R}^3$. К нахождению семейств нормализующих и эффективно нормализующих векторов в пространствах большей размерности мы вернемся позже, в разделе 34, после того как в разделах 25, 26, 30 получим достаточные условия нормальности, а в разделе 32 ознакомимся с методом последовательной достройки нормального расписания.

24 Минимальные нормализующие векторы в $\mathbb{R}^3$

Основной результат раздела содержит следующая

Теорема 24.1 *Вектор $(0, 0, 2)$ — эффективно нормализующий. Для любого примера задачи open shop с n работами и 3 машинами, который удовлетворяет условию $L_1 - L_3 \geq 2$ (при нумерации машин по умолчанию), его нормальное (а следовательно, оптимальное) расписание находится за время $O(n)$.*

Следующее утверждение непосредственно вытекает из теоремы 24.1 и утверждения 23.5.

Утверждение 24.2 *Вектор $(0, 0, 2)$ является единственным MN-вектором и единственным MEN-вектором в $\mathbb{R}^3$.* □

Остальная часть данного раздела посвящена исследованию свойств двухмашинных расписаний, подводящих нас к доказательству теоремы 24.1. Вначале описываются две схемы построения оптимального расписания на двух машинах (A и B), представляющие собой модификации схемы Гонзалеза и Сани. Далее вводится понятие непатологического расписания S_{AB} и отыскиваются условия его существования. После этого мы сможем приступить к доказательству основного результата.

Обозначения и соглашения

Без ограничения общности можем считать, что машины занумерованы по умолчанию и что $L_2 = L_1$. (Для достижения этого равенства мы можем использовать процедуру выравнивания, описанную в п.8 раздела 22.) Это делает машины M_1 и M_2 "равноправными". Более того, до конца раздела будут рассматриваться лишь примеры, удовлетворяющие соотношениям

$$L_1 = L_2 \geq L_3 + 2 \tag{24.1}$$

Машины M_1, M_2, и M_3 для удобства будем обозначать A, B и C (при этом машины A и B будут называться *доминантами*); операции работы J_j на машинах A, B и C будут обозначаться, соответственно, A_j, B_j и C_j, а для обозначения их длительностей будем использовать маленькие буквы: a_j, b_j, c_j.

Для задания ограничений предшествования наряду с обычными сетями (в которых вершины соответствуют операциям работ на машинах) мы будем использовать так называемые "схемы": в них каждая вершина x может быть либо операцией, либо *временно́й вершиной* с *временно́й пометкой* t_x. Если в схеме есть дуга $x \to y$, то это означает, что в расписании, удовлетворяющем данной схеме, время завершения операции x (или время t_x — в случае временно́й вершины x) должно предшествовать времени начала операции y в этом расписании (или времени t_y — в случае временно́й вершины y). Для определенности, при графическом изображении схемы, операции будут помещаться внутрь окружностей (чтобы отличать их от временны́х вершин).

В качестве временны́х пометок могут использоваться времена начала или окончания каких-то операций **в других расписаниях**.

Если расписание задано схемой, то мы должны во-первых убедиться, что схема допустима (т.е. длина любого пути, предшествующего временной вершине x, не превосходит t_x), и во-вторых, при вычислении длины расписания мы должны каждую временную вершину интерпретировать как корневую операцию длины t_x.

Времена начала и завершения операции O_j в расписании S будут обозначаться $s(O_j, S)$ и $c(O_j, S)$, соответственно; S_i будет обозначать раннее расписание, определенное схемой G_i.

Путь в схеме G называется *нормальным*, если его длина не превосходит L_1. Ясно, что если каждый путь в схеме нормален, то схема задает нормальное расписание. (Такие схемы будем также называть *нормальными*.)

В процессе построения желаемой нормальной схемы мы будем иногда использовать схемы, определяющие заведомо недопустимые расписания. Такие схемы будут назы-

ваться *начальными* и использоваться в дальнейшем для преобразования их в нормальные схемы.

Две модифицированных схемы Гонзалеза и Сани

Обозначим $l_j = \sum_{i=1}^{m} p_{ji},\ l_{\max} = \max_j l_j,\ \tilde{C} = \max\{L_{\max}, l_{\max}\}$. Ясно, что $C^*_{\max} \geq \tilde{C}$.

Как было отмечено выше, для двух машин алгоритм Гонзалеза и Сани [87] строит расписание длины $C_{\max}(S) = \tilde{C}$, которое тем самым является оптимальным. (Это расписание мы будем называть *GS-расписанием* и обозначать S_{GS}.) Схема G_{GS} для этого расписания изображена на рис.24.1 и обладает следующими характерными свойствами.

1. Имеется *диагональная* работа J_d, чья операция B_d предшествует операции A_d. Все остальные работы J_j имеют обратный порядок выполнения операций: $A_j \to B_j$.

2. Все работы в $\mathcal{J}$ поделены на два множества: $\bar{\mathcal{J}}_1$ — множество работ с $a_j \leq b_j$, и $\bar{\mathcal{J}}_2$ — множество остальных работ. Диагональная работа может быть как из $\bar{\mathcal{J}}_1$, так и из $\bar{\mathcal{J}}_2$. Положим $\mathcal{J}_1 = \bar{\mathcal{J}}_1 \setminus \{J_d\},\ \mathcal{J}_2 = \bar{\mathcal{J}}_2 \setminus \{J_d\}$, и перенумеруем работы в $\mathcal{J}$ таким образом, чтобы диагональная работа получила нулевой номер, работы из $\mathcal{J}_1$ получили первые $k \doteq |\mathcal{J}_1|$ номеров, а работы из $\mathcal{J}_2$ — номера с $k+1$ по $n-1$. Тогда согласно GS-схеме, операции на машине A должны выполняться в порядке $A_1 \to A_2 \to \cdots \to A_{n-1} \to A_0$, а на машине B — в порядке $B_0 \to B_1 \to B_2 \to \cdots \to B_{n-1}$.

3. Диагональная работа и нумерации работ в $\mathcal{J}_1$ и $\mathcal{J}_2$ должны быть выбраны таким образом, чтобы обеспечивать свойство

$$C_{\max}(S_{\mathrm{GS}}) = \tilde{C}. \tag{24.2}$$

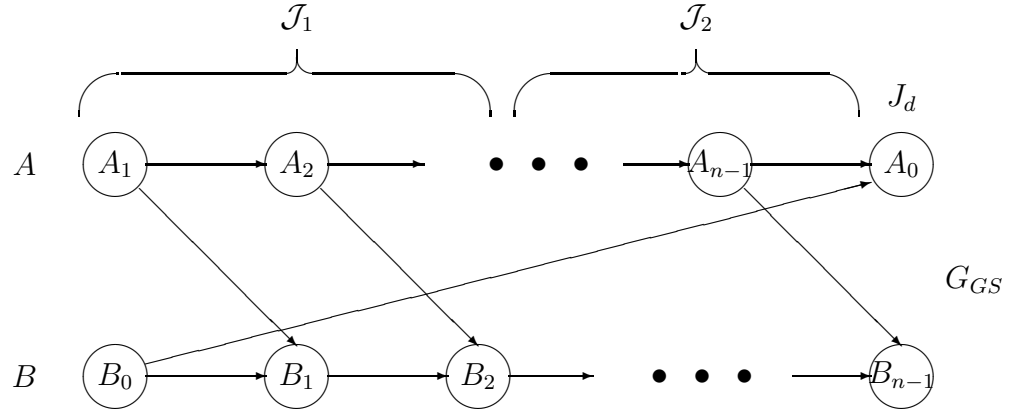


Рис. 24.1: Схема G_{GS}

Пункты 1–3 допускают большое разнообразие схем, характеризующихся различными правилами выбора диагональной работы и порядка выполнения работ в каждом

из множеств $\mathcal{J}_i$. В литературе известно по меньшей мере три различных реализаций GS-схемы.

Оригинальная реализация Гонзалеза и Сани [87] была итеративной: работы просматривались в порядке их исходной нумерации, и на каждом шаге $j = 1, \ldots, n$ постановка в расписание очередной работы J_j зависела от текущего частичного расписания. В такой реализации трудно предсказать, какая из работ станет диагональной.

Более явная процедура выбора диагональной работы описана в [61]. Она состоит из трех шагов. На первом шаге мы находим работу $J_l \in \bar{\mathcal{J}}_1$, такую что $b_l \geq a_j,\ \forall J_j \in \bar{\mathcal{J}}_1$, и ставим ее первой среди работ из $\bar{\mathcal{J}}_1$. (Например, это может быть работа J_l с $b_l = \max_{J_j \in \bar{\mathcal{J}}_1} b_j$.) На втором шаге мы аналогично находим работу $J_r \in \bar{\mathcal{J}}_2$, такую что $a_r \geq b_j,\ \forall J_j \in \bar{\mathcal{J}}_2$, и ставим ее последней среди работ из $\bar{\mathcal{J}}_2$. На третьем шаге мы стыкуем два полученных расписания (для работ из $\bar{\mathcal{J}}_1$ и работ из $\bar{\mathcal{J}}_2$) и анализируем, какая из двух работ $\{J_l, J_r\}$ должна стать диагональной. (Нам не потребуется более детальных знаний об этой процедуре.)

Наконец, совершенно определенная процедура P_B выбора диагональной работы описана в книге Брукера [78]. Она также состоит из трех шагов.

Шаг 1. Найти работу $J_l \in \bar{\mathcal{J}}_1$, такую что $a_l = \max_{J_j \in \bar{\mathcal{J}}_1} a_j$.

Шаг 2. Найти работу $J_r \in \bar{\mathcal{J}}_2$, такую что $b_r = \max_{J_j \in \bar{\mathcal{J}}_2} b_j$.

Шаг 3. Если $a_l \geq b_r$, то работа J_l берется в качестве диагональной. В противном случае диагональной становится работа J_r.

Именно благодаря такому выбору диагональной работы, построенное по этой схеме расписание обладает следующим свойством, доказанным в [78].

Лемма 24.3 (Brucker [78, pp.146–148]) *Если диагональная работа в GS-схеме определена согласно процедуре P_B, то* **для любого** *порядка выполнения работ в $\mathcal{J}_1$ и* **любого** *порядка выполнения работ в $\mathcal{J}_2$ раннее расписание S_{GS}, определяемое GS-схемой, имеет длину $\tilde{C}$.*

При построении искомого нормального расписания, необходимого для доказательства теоремы 24.1, мы будем использовать это свойство в более слабой форме, а именно, что любая работа (или две работы) из $\mathcal{J}_1$ может быть поставлена первой, и любая работа (или две работы) из $\mathcal{J}_2$ может быть поставлена последней в расписании S_{GS} без потери свойства (24.2). (Заметим, что расписания, построенные согласно алгоритмов из [87] и [61], этим свойством не обладают. Действительно, если, например, в алгоритме [61] работа J_r выбрана в качестве диагональной, то мы не можем произвольную работу из множества $\mathcal{J}_1$ поставить первой, поскольку согласно алгоритма первой должна стать работа J_l.)

GS-схема с диагональной работой, определенной согласно процедуре P_B, будет обозначаться G_{MGS}. Другая схема (G_{MGS_2}) получается для машин A, B из схемы G_{MGS} переносом операции A_1 из начала последовательности операций на машине A в ее конец, что порождает вторую диагональную работу в схеме (см. рис.24.2).

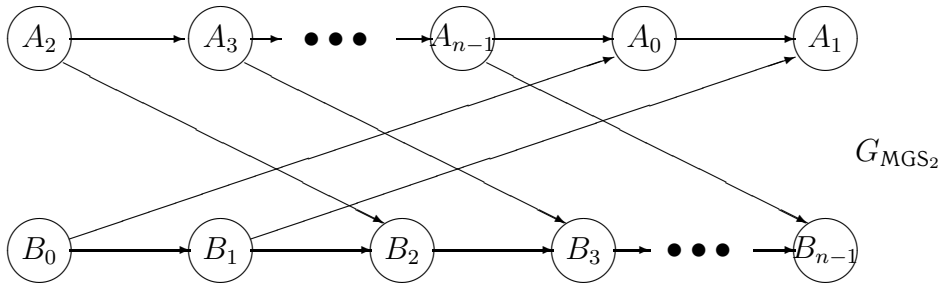


Рис. 24.2: Схема G_{MGS_2}

Наиболее ранние расписания, построенные по схемам G_{MGS} и G_{MGS_2}, будут называться *MGS-* и *MGS_2-расписаниями*.

Непатологические расписания

Теперь мы возвращаемся к случаю трех машин A, B, C. Заметим, что в силу (24.1) выполнено $L_1 \geq a_j + b_j$, $\forall\, J_j$, поэтому всякое расписание, оптимальное для машин A, B, заканчивается на этих машинах не позднее $\max\{L_{\max}, l_{\max}\} = L_{\max}$. Если при этом и на C оно заканчивается не позднее $L_{\max}$, то расписание нормально.

Определение 24.4 Нормальное расписание S_{AB} для машин A, B называется *патологическим* по отношению к работе J_k (а работа J_k называется *патологией* в расписании S_{AB}), если операция C_k не может быть выполнена во временном интервале $[0, L_1]$ без пересечения с одной из операций A_k, B_k. Расписание S_{AB} называется *непатологическим*, если оно нормально и не является патологическим по отношению к любой из работ $J_k \in \mathcal{J}$. Работа J_k называется *потенциальной патологией*, если существует нормальное расписание S_{AB}, патологическое по отношению к J_k.

Предположим, что GS-расписание S_{AB} является патологическим по отношению к работе J_k. Из (24.1) следует, что $c_d + a_d + b_d \leq L_1$, и значит, работа J_k не является диагональной в расписании S_{AB}. Поскольку известно, что для расположения операции C_k недостаточно места ни перед операцией A_k, ни после операции B_k, ни между ними, то мы приходим к соотношениям

$$a_k + b_k + 3c_k > L_1 \geq L_3 + 2, \tag{24.3}$$

из которых вытекает следующая

Лемма 24.5 *В любом примере, удовлетворяющем условиям (24.1), содержится не более двух потенциальных патологий.*

Доказательство. Предположим противное, т.е. имеются три различных работы J_k, J_l, J_x, являющиеся патологиями в каких-то нормальных расписаниях. Сложив неравенства (24.3) для этих трех работ, получим противоречие:

$$6 + 3L_3 \geq a_k + b_k + a_l + b_l + a_x + b_x + 3(c_k + c_l + c_x) > 3L_1 \geq 3L_3 + 6. \qquad \square$$

Лемма 24.6 *Для любого примера задачи $O3||C_{\max}$, удовлетворяющего условиям (24.1), с трудоемкостью $O(n)$ может быть найдено расписание S_{AB} для машин A, B, которое является либо непатологическим* MGS- *расписанием (см. рис.24.1), либо нормальным* MGS_2- *расписанием, определенным согласно MGS_2-схеме (см. рис.24.2) и удовлетворяющим следующим дополнительным условиям (для подходящей нумерации машин-доминант M_1, M_2):*

$$J_1, J_2 \in \mathcal{J}_1, \tag{24.4}$$

$$c_2 > a_1, \tag{24.5}$$

$$a_1 + a_2 + c_2 > b_0 + b_1, \tag{24.6}$$

$$b_0 + b_1 + b_2 + c_2 > L_1, \tag{24.7}$$

$$c_1 > a_2, \tag{24.8}$$

$$a_1 + a_2 + c_1 > b_0 + b_2, \tag{24.9}$$

$$b_0 + b_1 + b_2 + c_1 > L_1. \tag{24.10}$$

Доказательство. Пусть задан произвольный пример I задачи $O3||C_{\max}$. Сначала построим для него произвольное MGS- расписание S_1 для машин A, B. Если оно непатологическое, берем его за искомое расписание S_{AB}. Если же оно оказалось патологическим по отношению к некоторой работе J_k (что проверяется за время $O(n)$), то мы строим другое MGS- расписание S_2, в котором работа J_k ставится первой среди всех работ из $\mathcal{J}_1$ (в случае, когда $J_k \in \mathcal{J}_1$), или иначе, J_k ставится последней среди всех работ из $\mathcal{J}_2$ (в случае, когда $J_k \in \mathcal{J}_2$; так как диагональная работа в GS-расписании не может быть его патологией, то других случаев — кроме двух перечисленных — не может быть).

Предположим, что S_2 тоже патологическое — на этот раз, по отношению к другой работе $J_l \neq J_k$, поскольку J_k не может быть патологией в S_2. (Действительно, например в случае $J_k = J_1$ из соотношений $b_0 + b_1 + c_1 \leq 2 + L_3 \leq L_1$ следует, что операция C_1 вполне помещается в интервале $[c(B_1, S_1), L_1]$. Случай $J_k = J_{n-1}$ симметричен.) Рассмотрим два случая: когда J_k и J_l принадлежат различным множествам $\mathcal{J}_i$, и когда они находятся в одном и том же множестве $\mathcal{J}_i$.

В первом случае (без ограничения общности мы можем предположить, что $J_k \in \mathcal{J}_1, J_l \in \mathcal{J}_2$), строим MGS- расписание S_3, в котором $J_k = J_1$ и $J_l = J_{n-1}$. Поскольку ни J_k, ни J_l не могут патологиями в S_3, и по лемме 24.5 никакая другая работа не может быть патологией в S_3, то расписание S_3 может быть взято за искомое непатологическое расписание S_{AB}.

Во втором случае без ограничения общности мы можем предположить, что $J_k, J_l \in \mathcal{J}_1$. (Случай, когда обе работы находятся в $\mathcal{J}_2$, сводится к случаю $J_k, J_l \in \mathcal{J}_1$ переменой номеров машин M_1, M_2: $M_1 := B,\ M_2 := A$. — Мы можем это сделать, т.к. согласно (24.1) обе машины — доминанты.) Сначала построим два MGS- расписания S_4 и S_5, в которых работы J_k и J_l ставятся первыми среди всех работ из $\mathcal{J}_1$: в порядке $(J_k, J_l) \doteq (J_1, J_2)$ — в расписании S_4, и в порядке (J_2, J_1) — в расписании S_5. Если одно из этих расписаний непатологическое, берем его за искомое S_{AB}.

Теперь предположим, что оба расписания (S_4 и S_5) патологические. Поскольку работа J_2 является патологией в S_4, это влечет соотношения (24.5)–(24.7), которые означают, что никакое из трех возможных расположений операции C_2 (до A_2, между A_2 и B_2, после B_2, соответственно), не может привести к допустимому нормальному расписанию для трех машин. Аналогично, поскольку J_1 является патологией в S_5, это влечет соотношения (24.8)–(24.10).

Теперь построим MGS_2- расписание S_6 из расписания S_4, переместив операцию A_1 из начала в конец расписания (см. рис.24.2). Поскольку $b_0 + b_1 + a_1 < b_0 + b_1 + c_2 \leq 2 + L_3 \leq L_1$, то путь $B_0 \to B_1 \to A_1$ является нормальным в схеме G_6. Аналогично, путь $B_0 \to A_0 \to A_1$ также нормален. Нормальность остальных путей в G_6 следует из их нормальности в G_4. Следовательно, расписание S_6 нормально.

Таким образом, для нахождения искомого расписания S_{AB} достаточно построить не более четырех MGS- расписаний и не более одного MGS_2- расписания, что может быть сделано за время $O(n)$. Лемма 24.6 доказана. □

Доказательство теоремы 24.1

Предположим, что мы уже построили расписание S_{AB} для машин A и B со свойствами, декларируемыми в лемме 24.6. Покажем, что S_{AB} может быть достроено до нормального расписания для трех машин A, B, C (возможно, с небольшими изменениями расписания для A, B).

1. Сначала рассмотрим случай, когда S_{AB} является непатологическим MGS- расписанием (рис.24.1). Распишем операции $\{C_j\}$ на машине C в интервале $[0, L_3]$ в порядке $C_{n-1} \to C_{n-2} \to \cdots \to C_1 \to C_0$ (т.е. в порядке, обратном порядку выполнения работ на машине B). Полученное недопустимое расписание S' берется в качестве начального расписания, которое предстоит преобразовать в нормальное. Моменты начала и окончания какой-либо операции O' в расписании S' будем обозначать через $s(O')$ и $c(O')$.

Определение 24.7 Пусть у нас имеется недопустимое расписание S. Будем говорить, что *машины X и Y недопустимо перекрываются по работе J_j*, если интервалы выполнения операций X_j и Y_j перекрываются в расписании S (где $X, Y \in \{A, B, C\}$).

Нетрудно видеть, что в расписании S' машины C и A недопустимо перекрываются не более чем по одной работе J_k $(0 < k < n-1)$, и аналогично, машины C и B недопустимо перекрываются не более чем по одной работе (J_l). Заметим, что $l \leq k$, и что B_l и C_l перекрываются внутри интервала $[s(A_k), c(B_k)]$ (см. рис.24.3).

Теперь покажем, как преобразовать расписание S', чтобы устранить оба перекрытия. Сначала рассмотрим случай $k = l$. Поскольку $c(B_k) < L_3 + b_k$, получаем $c(B_k) + c_k < L_3 + 2 \leq L_1$, откуда следует, что положив $s(C_k) := L_1 - c_k$, мы устраним все недопустимые перекрытия в расписании. Таким образом, далее можем предполагать, что $k \neq l$.

Замечание 24.8 *Пусть мы имеем начальное расписание S, такое что, во-первых, операции на машине C выполняются в интервале $[0, L_3]$, во-вторых, нет недопустимых перекрытий на машинах A и C, в-третьих, B и C недопустимо перекрываются*

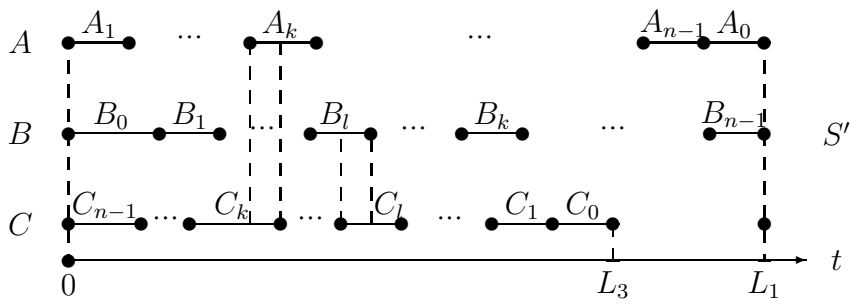


Рис. 24.3: Начальное расписание S'

по единственной работе J_l, и в-четвертых, S представляет собой MGS- *расписание на машинах A и B. Тогда это единственное перекрытие может быть легко устранено либо перемещением операции C_l в конец расписания (после C_0 и B_l) — в случае $J_l \neq J_0$, либо установлением порядка $B_0 \to C_0 \to A_0$, — в случае $J_l = J_0$.*

Далее мы рассмотрим несколько случаев и покажем, что в каждом случае существует преобразование начального недопустимого расписания S' в нормальное.

Случай 1:

$$c(C_k) \geq c(A_k). \tag{24.11}$$

Сдвинув операции $\{C_j \,|\, j \leq k\}$ вправо на 1, мы устраним все перекрытия по A и C и придем к некоторому расписанию S_1. В расписании S_1 перекрытие по B и Cо не более чем по одной работе $J_\nu,\ \nu \in \mathbb{N}_k$. Если такое перекрытие действительно возникло, берем S_1 в качестве начального расписания и пытаемся устранить перекрытие.

Случай 1.1: $J_\nu \neq J_k$.

Преобразуем расписание S_1 в расписание S_2, положив $s(C_\nu, S_2) := L_1 - c_\nu$. Если операции C_ν, B_ν перекрываются в S_2, имеем

$$c(B_\nu) > L_1 - 1. \tag{24.12}$$

В этом случае преобразуем S_1 в S_3, положив

$$s(C_\nu, S_3) := \max\{s(A_k), s(C_k)\},$$

и если необходимо, сдвинув операции $\{C_j \,|\, \nu < j \leq k\}$ вправо (установив ограничение $C_\nu \to C_k$).

Операции C_ν и B_ν не перекрываются в расписании S_3, поскольку $s(C_\nu, S_3) < s(C_\nu)$, и следовательно, $c(C_\nu, S_3) < c(C_\nu) \leq L_3 < s(B_\nu)$, согласно (24.12). Кроме того, для каждой операции $C_j,\ j \in \{\nu+1, \ldots, k\}$ имеем

$$c(C_j, S_3) \leq L_3 + 1 \overset{\text{по (24.12)}}{<} c(B_\nu) \leq s(B_j, S_3),$$

откуда следует, что операции C_j и B_j не перекрываются. Таким образом, расписание S_3 нормально.

Случай 1.2: $J_\nu = J_k$.

Очевидно, что $J_k \neq J_0$. Если C_k умещается в интервале $[c(B_k), L_1]$, то переместив C_k в конец расписания на C (после B_k), мы устраним все перекрытия. Поэтому далее можем предполагать, что

$$c(B_k) + c_k > L_1, \tag{24.13}$$

откуда следует

$$\sum_{j>k} b_j < c_k. \tag{24.14}$$

Построим расписание S_4 из S', сдвинув операции $\{C_j \,|\, j \leq k\}$ вправо, но не на 1 (как в S_1), а на меньшую положительную величину, достаточную для достижения равенства $c(C_k, S_4) = s(B_k)$. (Это возможно, поскольку $J_k \neq J_l$, и значит, C_k и B_k не перекрываются в S'.) В полученном расписании S_4 не будет перекрытий по B и C, но операции A_k и C_k все еще могут перекрываться. Последнее означает, что C_k не вмещается между A_k и B_k. Поскольку расписание S_{AB} непатологическое, то из (24.13) следует, что C_k помещается в промежуток перед A_k.

Построим расписание S_5 из S', положив $s(C_k, S_5) := 0$ и сдвинув операции $\{C_j \,|\, j > k\}$ вправо (установив предшествование $C_k \to C_{n-1}$). В расписании S_5 существует перекрытие на A и C по некоторой работе $J_x,\ x > k$. Преобразуем схему G_5 в одну из схем G_6, G_7 (в зависимости от того, какой из двух случаев: $J_l \neq J_0$ или $J_l = J_0$ имеет место), переместив операции C_x, C_l, как показано на рис.24.4.

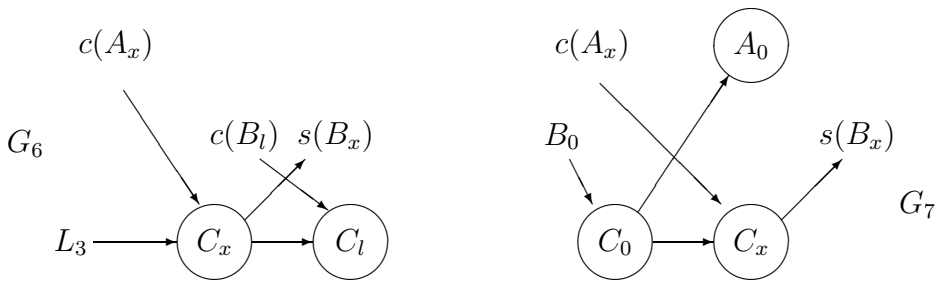


Рис. 24.4: Схемы G_6 и G_7

Case 1.2.1:

$$s(C_x, S_5) \geq s(A_x). \tag{24.15}$$

Поскольку

$$c(A_x) + c_x \overset{\text{по (24.15)}}{\leq} s(C_x, S_5) + c_x + a_x \leq L_3 + 1 \overset{\text{по (24.1)}}{\leq} L_1 - 1 \overset{\text{по (24.13)}}{\leq} c(B_k) \leq s(B_x),$$

путь $c(A_x) \to C_x \to s(B_x)$ допустим в G_6 и G_7. Поскольку $L_3 + c_x \leq L_1 - 1 \leq s(B_x)$, путь $L_3 \to C_x \to s(B_x)$ допустим в G_6. Поскольку

$$b_0 + c_0 + c_x \leq 1 + L_3 \leq L_1 - 1 \leq L_1 - c_k \overset{\text{по (24.14)}}{\leq} s(B_x),$$

путь $B_0 \to C_0 \to C_x \to s(B_x)$ допустим в G_7. Далее, из $c(A_x)+c_x+c_l \le L_1-1+c_l \le L_1$ следует, что путь $c(A_x) \to C_x \to C_l$ нормален в G_6, а нормальность пути $L_3 \to C_x \to C_l$ следует из (24.1). Поскольку операция B_l перекрывается с C_l в S', имеем $c(B_l) < L_3+1$, что в силу (24.1) влечет нормальность пути $c(B_l) \to C_l$ в G_6. Наконец, нормальность пути $B_0 \to C_0 \to A_0$ в G_7 следует из $b_0+c_0+a_0 \le L_3+2 \le L_1$. Нормальность остальных путей в G_6 и G_7 вытекает из их нормальности в расписании S_{AB}.

Case 1.2.2:

$$s(C_x, S_5) < s(A_x). \tag{24.16}$$

Нормальность схем G_6 и G_7 легко проверяется в случае $c(A_x) \le L_3$. Далее предполагаем, что

$$c(A_x) > L_3. \tag{24.17}$$

Case 1.2.2.1: $J_x \in \mathcal{J}_2$.

Преобразуем расписание S_5 в расписание S_8, сделав небольшую реконструкцию двухмашинного расписания S_{AB}: работу J_x поставим последней среди работ из $\mathcal{J}_2$. (Поскольку S_{AB} было MGS- расписанием, такое перемещение работы J_x, согласно лемме 24.3, не нарушает допустимости и нормальности расписания S_{AB}.) Кроме того, такое перемещение работы J_x устраняет перекрытие ее операций A_x, C_x, и в силу (24.16), новых перекрытий на машинах A и C не возникает. Единственное перекрытие по B и C устраняется согласно замечанию 24.8.

Case 1.2.2.2: $J_x \in \mathcal{J}_1$.

Поскольку $x > k$, отсюда следует, что $J_k \in \mathcal{J}_1$.

Case 1.2.2.2.1: $s(A_x) \ge c_k + c_x$.

Преобразуем S_5 в S_9, поместив C_x сразу после C_k и сдвинув операции $\{C_j \,|\, j > x\}$ вправо (см. рис.24.5).

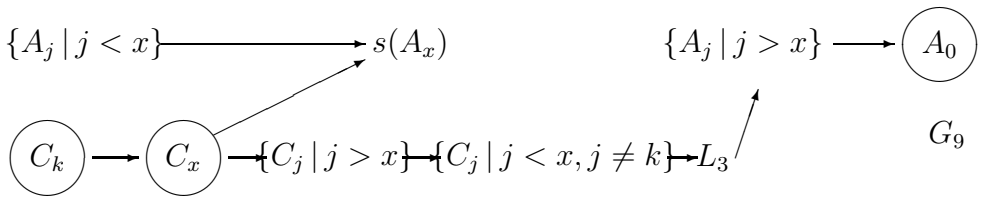


Рис. 24.5: Схема G_9

Из (24.17) следует, что в S_9 нет перекрытий по A и C. Перекрытие по B и C устраняется согласно замечанию 24.8.

Case 1.2.2.2.2:

$$s(A_x) < c_k + c_x. \tag{24.18}$$

Построим расписание S_{10} по схеме G_{10} (рис.24.6).

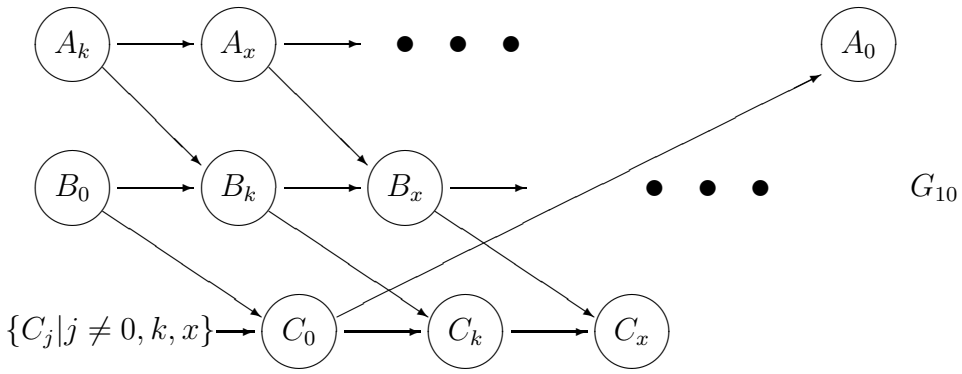


Рис. 24.6: Схема G_{10}

Из (24.17),(24.18) получаем $\sum_{j\neq k,x} c_j < a_x \leq b_0$, откуда следует, что операции $\{C_j \mid j \neq 0, k, x\}$ не перекрываются с операциями этих же работ на машинах A, B в расписании S_{10}. Нормальность путей в G_{10} следует из соотношений

$$\begin{aligned}
&\text{по (24.14)}\\
a_k + a_x + b_x + c_x \quad &\leq \quad a_k + a_x + (c_k + c_x) \leq 2 + L_3 \leq L_1,\\
b_0 + b_k + b_x + c_x &\leq b_0 + b_k + (c_k + c_x) \leq 2 + L_3 \leq L_1,\\
b_0 + c_0 + c_k + c_x &\leq 1 + L_3 < L_1,\\
b_0 + c_0 + a_0 &\leq 2 + L_3 \leq L_1.
\end{aligned}$$

(Рассмотрение путей, начинающихся с $A_k \to B_k$, может быть опущено ввиду $b_0 \geq a_k$.) Таким образом, расписание S_{10} нормально.

Case 2:

$$c(C_k) < c(A_k). \tag{24.19}$$

В случае $s(C_k) \geq s(A_k)$ мы вновь строим расписание S_1, и одно из расписаний $S_2 - S_5$ будет нормальным. (При этом нет необходимости строить расписания $S_6 - S_{10}$, поскольку, в силу (24.19), в расписании S_5 не будет перекрытий по A и C, а возможное перекрытие по B и C устраняется согласно замечанию 24.8.) Поэтому далее предполагаем, что

$$s(C_k) < s(A_k). \tag{24.20}$$

Case 2.1: $J_k \in \mathcal{J}_2$.

Преобразуем S' в расписание S_{11} путем небольшого преобразования расписания S_{AB}, а именно, поставив работу J_k последней среди работ из $\mathcal{J}_2$. В силу (24.1), мы тем самым устраняем перекрытие по работе J_k. При этом новых перекрытий по A и C не возникает ввиду (24.20). Возможное перекрытие по B и C устраняется согласно замечанию 24.8.

Case 2.2:

$$J_k \in \mathcal{J}_1. \tag{24.21}$$

Если $c_k \le s(A_k)$, то берем расписание S_5, не имеющее перекрытий по A и C, согласно (24.19). Перекрытие по B и C устраняется согласно замечанию 24.8, что дает требуемое расписание. Далее предполагаем, что $c_k > s(A_k)$. С учетом (24.21) и (24.20) получаем

$$\sum_{J_j \in \mathcal{J}_2} c_j \le \sum_{j>k} c_j = s(C_k) < s(A_k) < c_k, \tag{24.22}$$

т.е. суммарная длина операций $\{C_j \mid J_j \in \mathcal{J}_2\}$ строго меньше длины единственной операции C_k (для $J_k \in \mathcal{J}_1$).

Теперь построим новое начальное расписание S'', которое представляет собой "зеркальное отражение" расписания S': переименовываем машины A и B, тем самым меняя местами множества $\mathcal{J}_1$ и $\mathcal{J}_2$. Построив аналогичную серию расписаний ($\bar{S}_1 - \bar{S}_{11}$) на основе начального расписания S'', мы получим аналогичное "условие неудачи":

$$\sum_{J_j \in \mathcal{J}_1} c_j < c_\mu, \text{ for some } J_\mu \in \mathcal{J}_2,$$

вступающее в противоречие с (24.22). Отсюда следует, что одна из построенных схем должна обеспечивать допустимое нормальное расписание.

2. Теперь рассмотрим случай, когда расписание S_{AB} является MGS_2- расписанием, удовлетворяюшим условиям (24.4)–(24.10) (рис.24.2). Для получения искомого нормального расписания для трех машин распишем операции на машине C в интервале $[0, L_3]$ в порядке $\{C_j \mid j \ne 0,1,2\} \to C_1 \to C_2 \to C_0$ (см. схему G_{23} на рис. 24.7).

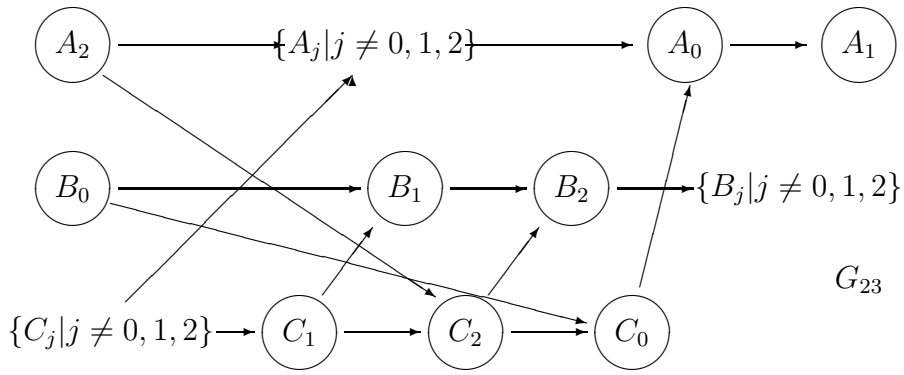


Рис. 24.7: Схема G_{23}

Для доказательства нормальности расписания S_{23} достаточно убедиться, что оно допустимо.

Из (24.1), (24.7), и (24.10) получаем

$$\min\{b_0, b_1, b_2\} > \max\left\{\sum_{j\ne 1} c_j,\ \sum_{j\ne 2} c_j\right\}. \tag{24.23}$$

Из соотношений

$$a_2 + c_2 + c_1 \overset{\text{по (24.5)}}{>} a_2 + a_1 + c_1 \overset{\text{по (24.9)}}{>} b_0 + b_2 \overset{\text{по (24.23)}}{>} 2\sum_{j\neq 1,2} c_j + c_1 + c_2$$

следует, что $a_2 > 2\sum_{j\neq 1,2} c_j$, откуда имеем $c(C_j, S_{23}) < s(A_j, S_{23}),\ \forall\ j \neq 0, 1, 2$.

Поскольку из (24.23) следует $b_0 > \sum_{j\neq 2} c_j$, получаем $c(C_1, S_{23}) < s(B_1, S_{23})$.

Из (24.8) имеем $c(A_2, S_{23}) < s(C_2, S_{23})$.

С учетом 42.25 получаем $b_0 + b_1 > L_3$, откуда следует $c(C_2, S_{23}) < s(B_2, S_{23})$.

Используя соотношения

$$b_0 \overset{\text{по (24.6)}}{<} a_1 + a_2 + c_2 - b_1 \overset{\text{по (24.23)}}{<} a_1 + a_2 \overset{\text{по (24.5),(24.8)}}{<} c_2 + c_1 \leq L_3 - c_0,$$

выводим $c(B_0, S_{23}) < s(C_0, S_{23})$.

Наконец, $c(C_0, S_{23}) \leq s(A_0, S_{23})$ следует из (24.1).

Таким образом, расписание S_{23} допустимо и нормально.

Для завершения доказательства теоремы 24.1 остается заметить, что как построение расписания S_{AB} со свойствами, декларированными в лемме 24.6, так и последующие преобразования его в расписания $S_1 - S_{11}, \bar{S}_1 - \bar{S}_{11}$ и S_{23} выполняются за время $O(n)$.

Теорема 24.1 доказана.

25 Нахождение эффективно нормальных классов с использованием нестрогого суммирования векторов

Теорема 25.1 *Пусть имеется алгоритм $\mathcal{A}$, который для некоторого $m \geq 2$ и любого $\hat{s}$-семейства векторов $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^{m-1}$ с трудоемкостью $\mathcal{T}_{\mathcal{A}}(m-1, n)$ решает задачу* HCB1$(m-1)$ *с гарантированной оценкой*

$$\theta_1(b_m) \leq \theta_1^*(m-1). \tag{25.1}$$

Тогда существует алгоритм, который для всякого примера задачи open shop с m машинами и n работами, удовлетворяющего условию

$$L_{\max} \geq (\theta_1^*(m-1) + 2m - 2)p_{\max}, \tag{25.2}$$

с трудоемкостью $O(\mathcal{T}_{\mathcal{A}}(m-1, n) + mn)$ находит оптимальное расписание длины $L_{\max}$.

Доказательство. Приведем описание алгоритма $\mathcal{A}_{25.1}$ построения оптимального расписания в задаче OS при условии выполнения (25.2) и в условиях теоремы 25.1.

Алгоритм $\mathcal{A}_{25.1}$

Шаг 1. Сохраняя неизменными величины $L_{\max}$ и $p_{\max}$, увеличим длительности отдельных операций так, чтобы выполнялись равенства (22.9). (Трудоемкость шага равна $O(mn)$.) Чтобы не менять обозначения всех величин, зависящих от величин $\{p_{ji}\}$, будем предполагать, что равенства (22.9) выполнялись для исходной матрицы (p_{ji}).

Определим векторы $d_j \in \mathbb{R}^{m-1},\ j \in \mathbb{N}_n$, согласно (22.11). Из (22.9) и определения нормы $\hat{s}$ следует, что если величину $p_{\max}$ принять за новую единицу измерения времени, то семейство векторов $D = \{d_1, \ldots, d_n\}$ становится $\hat{s}$-семейством в пространстве $\mathbb{R}^{m-1}$.

Шаг 2. Применив к семейству D алгоритм $\mathcal{A}$ из условия теоремы, найдем перестановку $\pi = (\pi_1, \ldots, \pi_n)$, задающую нестрогое суммирование векторов из D семействе полупространств

$$\{P(e_1 - e_2, \beta_1), \ldots, P(e_{m-1} - e_m, \beta_{m-1}), P(e_m - e_{m+1}, \beta_m)\},$$

где $e_m \doteq 0,\ e_{m+1} \dot{e}_1$,

$$\sum_{i=1}^{m} \beta_i \le \theta_1^*(m-1)p_{\max}. \tag{25.3}$$

Трудоемкость шага 2 совпадает с трудоемкостью алгоритма $\mathcal{A}$, т.е. равна $O(\mathcal{T}_{\mathcal{A}}(m-1, n))$.

Шаг 3. Для примера, полученного на шаге 1, построим нормальное расписание согласно следующему алгоритму.

Для каждой машины $M_i \in \mathcal{M}$ построим независимое расписание S_i, начинающееся в момент 0 и такое, что операции работ $J_j \in \mathcal{J}$ выполняются подряд без простоев в одном и том же порядке, задаваемом перестановкой π. Из (22.9) следует, что каждая машина работает непрерывно в интервале $[0, L_{\max}]$. (Очевидно, что набор расписаний $\{S_1, \ldots, S_m\}$, вообще говоря, не образует допустимого расписания для всего примера.) В последующем эти расписания претерпят ряд сдвигов вправо, при этом часть расписания, выходящая за границу интервала $[0, L_{\max}]$, будет отсекаться и перемещаться в начало этого интервала. Таким образом, расписания работы машин $M_i \in \mathcal{M}$ удобно представлять находящимися на “барабане” с длиной окружности, равной $L_{\max}$, где каждое расписание S_i представлено независимо вращающимся "кольцом"$\mathcal{K}_i$. При этом считаем, что кольцо $\mathcal{K}_1$ остается неподвижным и начало расписания S_1 задает точку отсчета времени. Положим

$$\Delta_i \doteq \max_{k \in \mathbb{N}_n} P_\pi^k(i, i+1),\ i \in \mathbb{N}_m, \tag{25.4}$$

где величины $P_\pi^k(i, i+1)$ определены согласно (22.13), а при определении $P_\pi^k(m, m+1)$ предполагаем, что $p_{\pi_j, m+1} \doteq p_{\pi_j 1}$.

Пусть k^* — (первое) значение k, на котором достигается максимум величины, стоящей в правой части (25.4) при $i = 1$. Таким образом,

$$\Delta_1 = P_\pi^{k^*}(1, 2) = P_\pi^{k^*}(1) - P_\pi^{k^*}(2). \tag{25.5}$$

Выполним $m-1$ последовательных поворотов (будем нумеровать их от 2 до m) колец $\mathcal{K}_2,\dots,\mathcal{K}_m$, оставляя кольцо $\mathcal{K}_1$ на месте.

Поворот 2. Повернем каждое из колец $\mathcal{K}_2,\dots,\mathcal{K}_m$ вправо на величину Δ_1, сохраняя неизменным их взаимное расположение. С учетом (25.5), момент $t^* \doteq P_\pi^{k^*}(1)$ совпадет с моментами окончания операции работы $J_{\pi_{k^*}}$ на машине M_1 и операции работы π_{k^*-1} на машине M_2. Момент t^* принимаем за будущую *точку разрезания* расписания на "барабане".

Поворот ν ($\nu=3,\dots,m$). Повернем каждое из колец $\mathcal{K}_\nu,\dots,\mathcal{K}_m$ вправо на величину $\Delta_{\nu-1}$, и затем — еще на минимальную величину $\delta_{\nu-1}\ge 0$ (*дополнительный поворот*), достаточную для того, чтобы момент t^* совпал с моментом окончания какой-то операции машины M_ν. Ясно, что

$$\delta_\nu < p_{\max},\ \ \nu=2,\dots,m-1. \tag{25.6}$$

Суммарный (по всем $m-1$ поворотам) поворот кольца $\mathcal{K}_m$ вправо относительно кольца $\mathcal{K}_1$ равен $\Delta' \doteq \sum_{i=1}^{m-1}\Delta_i + \sum_{i=2}^{m-1}\delta_i$, что эквивалентно его повороту влево на $L_{\max}-\Delta'$, или повороту кольца $\mathcal{K}_1$ вправо относительно $\mathcal{K}_m$ на эту же величину. Нетрудно убедиться, что если

$$L_{\max}-\Delta' \ge \Delta_m, \tag{25.7}$$

то полученное в результате всех поворотов колец расписание S' будет допустимым **на барабане**. Это означает, что интервалы выполнения любых двух операций одной и той же работы не пересекаются.

Теперь для получения из S' допустимого расписания длины $L_{\max}$ достаточно разрезать все кольца в точке t^*. Поскольку при этом ни для какой операции интервал ее выполнения не будет разрезан (что обеспечено величинами $\{\delta_\nu\}$ дополнительных поворотов), полученное расписание является допустимым. Очевидно, что оно будет допустимым и для исходной матрицы (более коротких) длительностей операций, имеющейся на входе до шага 1.

Трудоемкость шага 3 складывается из трудоемкости вычисления величин $\{P_\pi^k(i)\,|\,i\in\mathbb{N}_m;\ k\in\mathbb{N}_n\}$ согласно (22.12), величин $\{P_\pi^k(i,i+1)\,|\,i\in\mathbb{N}_m;\ k\in\mathbb{N}_n\}$ — согласно (22.13), величин Δ_i поворота каждого кольца $\mathcal{K}_i$, $i\in\mathbb{N}_m$, определенных согласно (25.4), и наконец, из вычисления величин δ_i дополнительного поворота этих колец. Ясно, что все эти вычисления могут быть выполнены за $O(mn)$ операций.

Алгоритм $\mathcal{A}_1$ описан. ■

Из оценок трудоемкости каждого шага алгоритма $\mathcal{A}_1$, приведенных в ходе его описания, следует, что общая трудоемкость алгоритма $\mathcal{A}_1$ удовлетворяет оценке, объявленной в теореме 25.1.

Из описания алгоритма $\mathcal{A}_1$ нетрудно заметить, что он всегда строит расписание длины $L_{\max}$ (возможно, недопустимое). Кроме того, при описании шага 3 было замечено, что построенное расписание будет допустимым, если исходный пример удовлетворяет условию (25.7). Предположим теперь, что исходный пример удовлетворяет условию (25.2) из формулировки теоремы 25.1. Докажем, что он удовлетворяет также (25.7).

Действительно,

$$\Delta' + \Delta_m = \sum_{i=1}^{m} \Delta_i + \sum_{i=2}^{m-1} \delta_i \leq \text{ (из (25.4),(25.6), замечания 22.1 и (25.3))}$$

$$\leq \sum_{i=1}^{m} \max_k P_\pi^k(i, i+1) + (m-2)p_{\max} \leq \sum_{i=1}^{m} (\beta_i + p_{\max}) + (m-2)p_{\max}$$

$$\leq (\theta_1^*(m-1) + 2m - 2)p_{\max} \leq \text{ (из (25.2))} \leq L_{\max}.$$

Таким образом, условие (25.7) выполнено и построенное расписание допустимо. Теорема 25.1 доказана. □

Из теоремы 25.1 и следствия 14.7 вытекает

Теорема 25.2 *Существует алгоритм трудоемкости $O(n \log n)$, который для всякого примера задачи* OS *с тремя машинами и n работами, удовлетворяющего условию*

$$L_{\max} \geq 7p_{\max}, \tag{25.8}$$

строит оптимальное расписание длины $L_{\max}$. □

Заметим, что построение оптимального расписания задачи OS(3) алгоритмом $\mathcal{A}_1$ возможно и в тех случаях, когда пример не удовлетворяет условию (25.8), но удовлетворяет (25.7). Во всех прочих случаях, как показано в [74], применение вместо алгоритма $\mathcal{A}_1$ простого жадного алгоритма позволяет с линейной от n трудоемкостью построить приближенное расписание S длины $C_{\max}(S) \leq L_{\max} + 2p_{\max}$.

В случае четырех и более машин задача open shop, как это следует из теоремы 25.1, сводится к решению задачи HCB1$(m-1)$ в пространстве размерности $m - 1 \geq 3$. Поскольку мы пока не имеем специального алгоритма решения этой задачи в пространствах такой размерности, нам остается воспользоваться каким-либо алгоритмом компактного суммирования векторов. Поскольку компактное суммирование $\hat{s}$-семейства векторов $D \subset \mathbb{R}^m$ в $\hat{s}$-шаре радиуса C обеспечивает выполнение неравенств

$$|d_\pi^k(i_1)| \leq C, \ |d_\pi^k(i_1) - d_\pi^k(i_2)| \leq C, \ i_1, i_2 \in \mathbb{N}_m,$$

тем самым обеспечивается их нестрогое суммирование в семействе $\mathcal{P}_1(m, b_{m+1})$ при $b_{m+1} = (C, \ldots, C)$, что гарантирует решение задачи HCB1(m) с оценкой

$$\theta_1(b_{m+1}) \leq (m+1)C \doteq \theta_1^*(m).$$

С учетом теоремы 25.1 получаем следующее следствие.

Теорема 25.3 *Пусть имеется алгоритм $\mathcal{A}$, который при любых натуральных n и m для любого $\hat{s}$-семейства $\{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ с трудоемкостью $\mathcal{T}_\mathcal{A}(m, n)$ решает задачу* KCB *с гарантированной оценкой*

$$f_{\hat{s},X}(\pi) \leq C_{\hat{s},m}^{\mathcal{A}}.$$

Тогда существует алгоритм, который для всякого примера задачи open shop с m машинами и n работами с трудоемкостью $O(\mathcal{T}_{\mathcal{A}}(m-1,n)+mn)$ строит расписание длины $L_{\max}$. При выполнении условия

$$L_{\max} \geq (mC^{\mathcal{A}}_{\hat{s},m-1} + 2m - 2)p_{\max},$$

построенное расписание является допустимым, а следовательно, оптимальным. □

Из теоремы 25.3 с учетом следствия 11.8 получаем

Следствие 25.4 *Условие*

$$L_{\max} \geq \left(m^2 - 1 + \frac{1}{m-1}\right) p_{\max} \tag{25.9}$$

определяет эффективно нормальный класс примеров задачи open shop с m машинами. Существует алгоритм трудоемкости $O(n^2m^2)$, который для любого примера задачи open shop с m машинами и n работами, удовлетворяющего условию (25.9), строит оптимальное расписание длины $L_{\max}$. □

26 Построение эффективно нормального класса методом Фиалы

Результат этого раздела основывается на методе Фиалы [84] построения расписаний длины $L_{\max}$. Идея этого метода состоит в построении расписания в виде "слоеного пирога" (типа "негр с блондинкой"): на каждой из m машин m' "белых" слоев чередуются с m' "черными" слоями (где $m' = 2^k,\ k = \lceil \log_2 m \rceil$). Предварительно всё множество работ разбивается на две кучи ("белых" и "черных" работ) так, чтобы нагрузка на каждой машине делилась примерно поровну, а затем множество операций каждой кучи делится на m' подмножеств так, чтобы, во-первых, в каждом подмножестве было не более одной операции каждой работы (что выполнимо, поскольку каждая работа имеет не более m операций), а во-вторых, чтобы эти подмножества давали примерно одинаковую нагрузку на каждую из m машин. Затем строится плотное расписание, в котором найденные подмножества операций формируют на каждой машине "белые" и "черные" слои. Находятся достаточные условия, при выполнении которых построенное плотное расписание является допустимым, а следовательно (ввиду $C_{\max} = L_{\max}$), оптимальным.

В качестве такого условия в работе Фиалы предлагалось неравенство

$$L_{\max} \geq (16m' \log_2 m' + 4m' + m)\, p_{\max}. \tag{26.1}$$

При этом нормальное расписание строится за время $O(n^2m^3)$.

На этой же идее основан алгоритм Бараня и Фиалы, описанный в работе [74] (на венгерском языке). В ней предлагается более эффективный (трудоемкости $O(nm^3)$) алгоритм построения нормального расписания при ослабленном условии:

$$L_{\max} \geq (16m \log_2 m + 26m)\, p_{\max}. \tag{26.2}$$

Поскольку в худшем случае величина m' оценивается сверху как $2m-2$, то по сравнению с (26.1), условие (26.2) примерно вдвое сужает интервал неблагоприятных значений $L_{\max}$ (при которых не гарантируется оптимальность строящегося расписания).

Дальнейшим развитием метода Фиалы является алгоритм, предлагаемый в теореме 26.1. В ней трудоемкость алгоритма снижается на два порядка (по сравнению с алгоритмом Фиалы), а интервал неблагоприятных значений $L_{\max}$ сужается примерно в 6 раз. Общая схема алгоритма остается примерно той же, однако вместо m' слоев каждого цвета мы предлагаем делать m слоев. При этом используется более эффективный алгоритм решения задачи о нахождении подмножества операций из раздела 10 (стр. 69) и изменяется схема применения этого алгоритма — вместо простой дихотомии по номерам слоев используется схема формирования слоев в порядке возрастания рангов их номеров. Использование свойств рангов целых чисел (полученных в разделе 17) позволило улучшить условие (26.1), а более детальная проработка подалгоритмов, используемых на отдельных шагах алгоритма $\mathcal{A}_{10.2}$ нахождения базы семейства ребер графа (из раздела 10, стр. 52), позволило понизить итоговую трудоемкость алгоритма.

Теорема 26.1 *Существует алгоритм трудоемкости $O(nm^2 \log m)$, который для любого примера задачи open shop с m машинами и n работами строит расписание длины $L_{\max}$. Расписание является допустимым (а следовательно, оптимальным) при выполнении условия*

$$L_{\max} \geq \left(\frac{16}{3} m \log_2 m' + \frac{13}{9} m - \frac{4m}{9m'} (-1)^k \right) p_{\max}, \tag{26.3}$$

где $k = \lceil \log_2 m \rceil$, $m' = 2^k$. □

Доказательство. В этом разделе для обозначения операций, их длительностей, их работ и исполняющих их машин нам удобнее будет пользоваться обозначениями из раздела 10 (стр. 69). Для величины $p_{\max}$ введем более короткое обозначение K.

Алгоритм $\mathcal{A}_{26.1}$ (построения расписания длины $L_{\max}$)

Шаг 1 (выравнивание нагрузок машин).
От длительностей операций p_i переходим к длительностям p'_i, удовлетворяющим свойствам:

$$p_i \leq p'_i \leq p_{\max}, \ \forall\ o_i \in \mathcal{O}, \tag{26.4}$$

$$L_j \doteq \sum_{o_i \in F_j} p'_i = L_{\max}, \ \forall\ j \in \mathbb{N}_m, \tag{26.5}$$

где $\mathcal{O}$ — множество всех операций, F_j — множество операций машины j. Как отмечалось в п.8 раздела 22, такие величины p'_i могут быть найдены с трудоемкостью $O(nm)$. Далее под p_i будем понимать их новые значения p'_i.

Шаг 2 (разбиение множества работ на два подмножества).
Применяя алгоритм из леммы 10.8 (стр. 60) при $\lambda_i \equiv \frac{1}{2},\ \forall\ i \in \mathbb{N}_n$, и норме $s = l_\infty$ с трудоемкостью $O(nm^2)$ находим разбиение множества работ $\mathcal{J}$ на два подмножества $\mathcal{J}_1$ и $\mathcal{J}_2$ такие, что:

$$\left| \sum_{o_i \in \mathcal{O}_\nu \cap F_j} p_i - \frac{1}{2} L_{\max} \right| \le \frac{m}{2} K,\ \forall\ j \in \mathbb{N}_m,\ \nu \in \{1,2\}, \tag{26.6}$$

где $\mathcal{O}_\nu = \{o_i \,|\, J(o_i) \in \mathcal{J}_\nu\}$.

Шаг 3 (формирование "черных" и "белых" слоев будущего "пирога").
Для каждого $\nu \in \{1,2\}$ формируем вложенную последовательность подмножеств операций $\varnothing = \mathcal{O}^{\nu 0} \subset \mathcal{O}^{\nu 1} \subset \ldots \subset \mathcal{O}^{\nu m} = \mathcal{O}_\nu$, удовлетворяющую свойствам:

$$|\mathcal{O}^{\nu l} \cap D_j| = l,\ \forall\ j \in \mathbb{N}_n,\ l \in \mathbb{N}_m,\ \nu \in \{1,2\}; \tag{26.7}$$

$$\left| \sum_{o_i \in \mathcal{O}^{\nu l} \cap F_j} p_i - \frac{l}{m} \sum_{o_i \in \mathcal{O}_\nu \cap F_j} p_i \right| \le \delta_m(l)\, K,\ \forall\ j \in \mathbb{N}_m,\ l \in \overline{\mathbb{N}}_m,\ \nu \in \{1,2\}, \tag{26.8}$$

где функция $\delta_m(l)$ определена согласно (17.1), (17.2). Для каждого $\nu \in \{1,2\}$ множества $\mathcal{O}^{\nu l}$ определяем (и находим) в порядке убывания рангов $r_m(l)$ индексов l:

$\boldsymbol{r_m(l) = k = \lceil \log_2 m \rceil}$. $\mathcal{O}^{\nu 0} := \varnothing;\ \mathcal{O}^{\nu m} = \mathcal{O}_\nu$.
$\boldsymbol{r_m(l) = r < k}$**.** Предполагаем, что для чисел $\{l'\}$ с рангом $r_m(l') > r$ множества $\mathcal{O}^{\nu l'}$ уже определены. В частности, для "родителей" числа l, — чисел $l^+ \doteq j_m^+(l),\ l^- \doteq j_m^-(l)$, — известны множества $\mathcal{O}^{\nu l^+}$ и $\mathcal{O}^{\nu l^-}$, удовлетворяющие свойствам (26.7), (26.8). Применяя алгоритм из леммы 10.14 для решения задачи ПО$(l^+ - l^-, l - l^-)$ с множеством операций $\mathcal{O}'' \doteq \mathcal{O}^{\nu l^+} \setminus \mathcal{O}^{\nu l^-}$, найдем подмножество операций $\mathcal{O}' \subset \mathcal{O}''$, такое что

$$|\mathcal{O}' \cap D_j| = l - l^-,\ \forall\ j \in \mathbb{N}_n, \tag{26.9}$$

$$\left| \sum_{o_i \in \mathcal{O}' \cap F_j} p_i - \frac{l - l^-}{l^+ - l^-} \sum_{o_i \in \mathcal{O}'' \cap F_j} p_i \right| \le K, \tag{26.10}$$

и положим $\mathcal{O}^{\nu l} := \mathcal{O}^{\nu l^-} \cup \mathcal{O}'$. Тогда (26.7) для l следует из (26.7) для l^- и (26.9), а (26.8) для l легко получить из (26.8) для l^+ и l^-, (26.10) и определения (17.1) функции $\delta_m(l)$.

Оценим трудоемкость шага 3. Нахождение множества $\mathcal{O}^{\nu l}$, согласно лемме 10.14, потребует $O((|\mathcal{O}^{\nu l^+}| - |\mathcal{O}^{\nu l^-}|)m)$ вычислительных операций. Для всех чисел $\{l\}$ ранга $r(l) = r$ это составит в сумме, с учетом утверждений 17.1 и 17.4, $O(|\mathcal{O}_\nu| m)$. Просуммировав по всем рангам $r < k$ и по $\nu \in \{1,2\}$, получим оценку трудоемкости $O(|\mathcal{O}| mk) = O(nm^2 \log m)$.

Определим множества операций: $\mathcal{O}_l^\nu := \mathcal{O}^{\nu l} \setminus \mathcal{O}^{\nu,l-1},\ \mathcal{O}_{lj}^\nu \doteq \mathcal{O}_l^\nu \cap F_j,\ \forall\ l \in \mathbb{N}_m,\ j \in \mathbb{N}_n,\ \nu \in \{1,2\}$.

Шаг 4 (построение расписания длины $L_{\max}$).
Организуем работу машин по выполнению множества операций $\mathcal{O}$ так, чтобы:

а) каждая машина работала непрерывно в интервале $[0, L_{\max}]$;

б) на каждой машине j множество ее операций F_j было упорядочено соотношениями: $\mathcal{O}_{1j}^1 \prec \mathcal{O}_{1j}^2 \prec \mathcal{O}_{2j}^1 \prec \mathcal{O}_{2j}^2 \prec \ldots \prec \mathcal{O}_{mj}^2$, где $\mathcal{O}' \prec \mathcal{O}''$ означает, что все операции из множества $\mathcal{O}'$ предшествуют каждой операции из $\mathcal{O}''$.

Свойства а) и б) однозначно определяют интервал выполнения операций из каждого множества $\mathcal{O}_{lj}^\nu$ и определяют расписание $S = \{s(o_i)\}$ с точностью до перестановки операций внутри каждого множества $\mathcal{O}_{lj}^\nu$. Любое такое расписание может быть определено с трудоемкостью $O(|\mathcal{O}|) = O(nm)$.

Алгоритм $\mathcal{A}_{26.1}$ описан. ■

Лемма 26.2 *При выполнении условия (26.3) расписание S, получаемое на шаге 4 алгоритма $\mathcal{A}_{26.1}$, является допустимым.*

Доказательство. Из (26.7) следует, что в каждом блоке операций $\mathcal{O}_l^\nu$ содержится ровно одна операция каждой работы $j \in \mathcal{J}_\nu$. Таким образом, нам достаточно доказать, что если $o_{i_1} \in \mathcal{O}_l^\nu \cap D_j,\ o_{i_2} \in \mathcal{O}_{l+1}^\nu \cap D_j$, то операция o_{i_1} предшествует операции o_{i_2}. Для этого, в свою очередь, достаточно доказать для всех $l \in \mathbb{N}_{m-1},\ \nu \in \{1,2\}$ и любых машин $j', j'' \in \mathbb{N}_m$ соотношение $\mathcal{O}_{lj'}^\nu \prec \mathcal{O}_{l+1,j''}^\nu$, или

$$\Delta^\nu(l+1, j', j'') \doteq s(\mathcal{O}_{l+1,j'}^\nu) - c(\mathcal{O}_{lj''}^\nu) \geq 0, \tag{26.11}$$

где $s(\mathcal{O}_{lj}^\nu)$ и $c(\mathcal{O}_{lj}^\nu)$ — моменты начала и окончания выполнения операций из блока $\mathcal{O}_{lj}^\nu$.

По построению расписания на шаге 4 для любой машины $j \in \mathbb{N}_m$ имеем

$$s(\mathcal{O}_{l+1,j}^1) = c(\mathcal{O}_{lj}^2) = \sum_{o_i \in \mathcal{O}^{1l} \cap F_j} p_i + \sum_{o_i \in \mathcal{O}^{2l} \cap F_j} p_i, \tag{26.12}$$

$$s(\mathcal{O}_{l+1,j}^2) = c(\mathcal{O}_{l+1,j}^1) = \sum_{o_i \in \mathcal{O}^{1,l+1} \cap F_j} p_i + \sum_{o_i \in \mathcal{O}^{2l} \cap F_j} p_i. \tag{26.13}$$

Отсюда при $\nu = 1$ получаем

$$\Delta^1(l+1, j', j'') = \sum_{o_i \in \mathcal{O}^{1l} \cap F_{j'}} p_i + \sum_{o_i \in \mathcal{O}^{2l} \cap F_{j'}} p_i - \sum_{o_i \in \mathcal{O}^{1l} \cap F_{j''}} p_i - \sum_{o_i \in \mathcal{O}^{2,l-1} \cap F_{j''}} p_i$$

$$\overset{\text{из (26.8)}}{\geq} \frac{l}{m} \sum_{o_i \in \mathcal{O}_1 \cap F_{j'}} p_i - \delta_m(l)\, K + \frac{l}{m} \sum_{o_i \in \mathcal{O}_2 \cap F_{j'}} p_i - \delta_m(l)\, K - \frac{l}{m} \sum_{o_i \in \mathcal{O}_1 \cap F_{j''}} p_i - \delta_m(l)\, K$$

$$- \frac{l-1}{m} \sum_{o_i \in \mathcal{O}_2 \cap F_{j''}} p_i - \delta_m(l-1)\, K \overset{\text{из (26.5)}}{=} \frac{1}{m} \sum_{o_i \in \mathcal{O}_2 \cap F_{j''}} p_i - (3\delta_m(l) + \delta_m(l-1))\, K$$

$$\overset{\text{из (26.6)}}{\geq} \frac{L_{\max}}{2m} - \left(\frac{1}{2} + \psi_m^-(l)\right) K \geq \text{(из леммы 17.10)}$$

$$\geq \frac{L_{\max}}{2m} - \left(\frac{8}{3}\log_2 m' + \frac{13}{18} + (-1)^{k-1}\frac{2}{9m'}\right) K \overset{\text{из (26.3)}}{\geq} 0.$$

Аналогично при $\nu = 2$ оцениваем

$$\Delta^2(l+1, j', j'') \overset{\text{из (26.11)–(26.13)}}{\geq} \sum_{o_i \in \mathcal{O}^{1,l+1} \cap F_{j'}} p_i + \sum_{o_i \in \mathcal{O}^{2,l} \cap F_{j'}} p_i - \sum_{o_i \in \mathcal{O}^{1,l} \cap F_{j''}} p_i$$

$$- \sum_{o_i \in \mathcal{O}^{2,l} \cap F_{j''}} p_i \overset{\text{из (26.8), (26.5)}}{\geq} \frac{1}{m} \sum_{o_i \in \mathcal{O}_1 \cap F_{j'}} p_i - (3\delta_m(l) + \delta_m(l+1)) K$$

$$\overset{\text{из (26.6)}}{\geq} \frac{L_{\max}}{2m} - \left(\frac{1}{2} + \psi_m^+(l)\right) K \geq 0.$$

Лемма 26.2 доказана. □

Очевидно, что расписание S, допустимое для длительностей p'_i, является допустимым и для исходных длительностей p_i. А поскольку длина расписания равна $L_{\max}$, то оно оптимально. Требуемая оценка трудоемкости алгоритма $\mathcal{A}_{26.1}$ складывается из оценок трудоемкости его этапов. Теорема 26.1 доказана. □

Если правую часть (26.3) оценить сверху функцией от m (используя неравенства $m' \leq 2m - 2$, $\ln(1 - \Delta) < -\Delta$, а также $m \geq 3$, — ввиду того, что при $m = 2$ задача полиномиально разрешима [87]), то получим

Следствие 26.3 *Для задачи* OS *с* $m \geq 3$ *машинами и* n *работами существует алгоритм трудоемкости* $O(nm^2 \log m)$*, который строит нормальное расписание для всякого примера, удовлетворяющего условию*

$$L_{\max} \geq \left(\frac{16}{3} m \log_2 m + \frac{61}{9} m - 7.4\right) p_{\max}.$$ □

27 Оценки функций $\eta_n(m)$ и $\eta_e(m)$

Из следствий 25.4 и 26.3 вытекают следующие верхние оценки на функцию $\eta_{en}(m)$, определенную на стр. 161:

$$\eta_{en}(m) \leq m^2 - 1 + \frac{1}{m-1},$$

$$\eta_{en}(m) \leq \frac{16}{3} m \log_2 m + \frac{61}{9} m - 7.4.$$

Нижняя оценка функций $\eta_n(m)$ и $\eta_{en}(m)$ предлагается в следующей теореме.

Теорема 27.1 $\eta_{en}(m) \geq \eta_n(m) \geq 2m - 2$.

Доказательство. Для доказательства теоремы достаточно для любого целого m и любого $\varepsilon > 0$ привести пример задачи open shop с m машинами, удовлетворяющий условию $L_{\max} > 2m - 2 - \varepsilon$ и не обладающий нормальным расписанием.

Пусть задано $\varepsilon > 0$. Множество работ состоит из одной A-работы с m A-операциями единичной длины и из $m(2m-3)$ B-работ, каждая с одной операцией длины $(1-\delta)$, где $0 < \delta < \varepsilon/(2m-3)$. Кроме A-операции каждая машина выполняет $(2m-3)$ B-операций, откуда следует $L_i = 1 + (2m-3)(1-\delta) > -2 - \varepsilon,\ \forall i \in \mathbb{N}_m$.

Пусть для этого множества работ существует расписание длины $L_{\max}$. Занумеруем машины в соответствии с порядком выполнения A-операций. Пусть k_i — количество B-операций, предшествующих A-операции на i-й машине. Поскольку каждая машина работает без простоя в интервале времени $[0, L_{\max}]$, то $k_i - k_{i-1} \geq 2,\ \forall i = 2, \ldots, m$, откуда $k_m \geq 2(m-1)$, что противоречит условиям данного примера □

Из теорем 25.2 и 27.1 вытекает

Следствие 27.2 *Для минимальной функции $\eta_n(m)$, гарантирующей нормальность всякого примера задачи* OS*, удовлетворяющего условию $L_{\max} \geq \eta_n(m)p_{\max}$, при $m = 3$ справедливы оценки:*

$$4 \leq \eta_n(3) \leq 7.$$

Из теоремы 27.1 следует, что множество примеров задачи OS, удовлетворяющих условию $L_{\max} < (2m-2)p_{\max}$, содержит как нормальные примеры, так и те, которые не являются нормальными. Естественно поставить вопрос о построении эффективной процедуры распознавания свойства нормальности на множестве входов задачи open shop. Доказываемая ниже теорема дает частичный ответ на этот вопрос.

Пусть задано рациональное число $\lambda \geq 1$. Через $OS(m, \lambda)$ обозначим класс m-машинных примеров задачи open shop, имеющих целочисленные длительности операций и удовлетворяющих условию $L_{\max} = \lambda p_{\max}$.

Теорема 27.3 *Для любого фиксированного целого $m \geq 3$ и любого фиксированного рационального $\lambda \in (1, 2m-3)$ проблема распознавания свойства нормальности на классе примеров $OS(m, \lambda)$ является NP-трудной.*

Доказательство. Пусть заданы рациональное число $\lambda \geq 1$ и целое $m \geq 3$. Сформулируем задачу **CHP**(m, λ) распознавания *Существования Нормального Расписания* для примера из класса $OS(m, \lambda)$.

CHP$(\boldsymbol{m, \lambda})$.
УСЛОВИЕ: заданы целые m, n', рациональное число $\lambda \geq 1$ и матрица (p_{ji}), $i \in \mathbb{N}_m$, $j \in \mathbb{N}_{n'}$, неотрицательных целых чисел, такие что $\sum_j p_{ji} = \lambda p_{\max},\ i \in \mathbb{N}_m$, где $p_{\max} = \max_{i,j} p_{ji}$.
ВОПРОС: существует ли допустимое расписание задачи OS с m машинами, n' работами и матрицей длительностей (p_{ji}), в котором каждая машина работает без простоев в интервале времени $[0, L_{\max}]$?

Для сведения к задаче CHP(m,λ) будет использована NP-полная задача РАЗБИЕНИЕ ([17, стр.66]).

Пусть заданы n неотрицательных целых чисел $\{c_1,\ldots,c_n\}=C$, целое $m\geq 3$ и рациональное $\lambda\in(1,2m-3)$. Построим пример задачи OS: $I(m,\lambda,C)\in OS(m,\lambda)$, — для которого выполнено свойство

$$C_{\max}(S_{\text{opt}})=L_{\max}, \tag{27.1}$$

если и только если существует подмножество $N'\subset\mathbb{N}_n$ такое, что $\sum_{i\in N'}c_i=\tilde{c}\doteq\sum c_i/2$.

Вначале построим пример $I'(m,\lambda,C)$ задачи OS, удовлетворяющий свойству (27.1), но имеющий рациональные длительности операций. Множество работ состоит из работ трех типов: одной A-работы с m операциями, $(2m-4)m+(m-1)L$ B-работ и n C-работ, каждая с одной операцией. Операции работ будем называть A-,B- и C-операциями соответственно. Имеется m машин, одну из которых будем называть "специальной". A-операции на специальной машине имеют длину $\tilde{\gamma}$, а на остальных машинах — γ. Все C-операции выполняются на "специальной" машине и имеют длительности $\{c_i\mid i\in\mathbb{N}_n\}$. Все B-операции имеют одинаковую длительность $\gamma-\tilde{c}$ и распределены среди m машин следующим образом: $(2m-4)$ из них — на "специальной" машине и по $(2m-4+L)$ — на каждой из оставшихся машин. Параметры L,γ и $\tilde{\gamma}$ определим в зависимости от значения λ.

Будем различать два случая: $\lambda\in(1,m]$ и $\lambda\in(m,2m-3)$. В обоих случаях положим

$$\tilde{\gamma}=(L+1)\gamma-(L+2)\tilde{c}. \tag{27.2}$$

Из (27.2) следует, что все m машин имеют одинаковую нагрузку $(L_{\max})$. Положим:

$$L=\left\lfloor\frac{2m-2}{\lambda-1}\right\rfloor,\ \gamma=\frac{\lambda(L+2)-L-2m+4}{\lambda(L+1)-L-2m+3}\tilde{c},\ \text{— в случае } \lambda\in(1,m];$$
$$L=0,\ \gamma=\frac{2m-4}{2m-3-\lambda}\tilde{c},\ \text{— в случае } \lambda\in(m,2m-3).$$

Предоставим читателю убедиться, что в обоих случаях знаменатель дроби, определяющей γ, положителен, и

$$\gamma>2\tilde{c}. \tag{27.3}$$

Из (27.2) можно заметить, что $L\geq 2$ и $\tilde{\gamma}>\gamma$, — в первом случае, и $\tilde{\gamma}=\gamma-2\tilde{c}<\gamma$, — во втором. Таким образом, мы имеем $p_{\max}=\tilde{\gamma}$, — в первом случае, и $p_{\max}=\gamma$, — во втором. Также предоставим читателю убедиться, что $L_{\max}=\lambda p_{\max}$ в обоих случаях.

Предположим, что пример $I'(m,\lambda,C)$ имеет расписание S длины $L_{\max}$. Покажем, что существует подмножество $N'\subset\mathbb{N}_n$ такое, что $\sum_{i\in N'}c_i=\tilde{c}$.

Занумеруем машины согласно порядку выполнения A-операций в расписании S. Предположим, что "специальная" машина имеет номер m. Поскольку расписание не имеет простоев, то по крайней мере две B-операции предшествуют A-операции на второй машине, по крайней мере 4 — на третьей, и т.д., по крайней мере $2(m-2)$ — на $(m-1)$-й машине и по крайней мере $2m-3$ B-операции — на m-й машине (ввиду

(27.3)), что невозможно. Отсюда заключаем, что A-операция на "специальной" машине не может быть ни первой, ни последней из A-операций, т.е. "специальная" машина имеет номер $i \in \{2, \ldots, m-1\}$.

С помощью тех же рассуждений заключаем, что x B-операций (где $x \geq 2(i-2)+1$) предшествуют A-операции на "специальной" машине и y B-операций (где $y \geq 2(m-i-1)+1$) следуют за ней в расписании. Поскольку отсюда получаем $x+y \geq 2m-4$, то имеем равенства: $x = 2i-3$, $y = 2m-2i-1$. Кроме того, суммарная длина C-операций, предшествующих A-операции на "специальной" машине, должна быть не меньше $\tilde{c}$, и то же самое верно для C-операций, следующих за ней, откуда следует существование такого подмножества $N' \subset \mathbb{N}_n$, что $\sum_{i \in N'} c_i = \tilde{c}$.

Обратная импликация (если $\sum_{i \in N'} c_i = \tilde{c}$, то для примера $I'(m, \lambda, C)$ существует расписание длины $L_{\max}$) очевидна. Таким образом, пример $I'(m, \lambda, C)$ удовлетворяет свойству (27.1).

Чтобы получить пример $I(m, \lambda, C) \in OS(m, \lambda)$ с целочисленными длительностями, достаточно длительности всех операций примера $I'(m, \lambda, C)$ умножить на целое число x, равное удвоенному произведению знаменателей рациональных чисел λ и γ (т.е. на число $x = 2p(L+1)-2q(L+2m-3)$, — в случае $\lambda \in (1, m]$, и на число $x = 2q(2m-3)-2p$, — в случае $\lambda \in (m, 2m-3)$, где p и q суть числитель и знаменатель рационального числа λ). Поскольку $L_{\max}$ и $p_{\max}$ одновременно увеличиваются в x раз, то свойство (27.1) сохраняется. Замечаем, что при фиксированных m и λ чи́сла L и x также фиксированы, а длина записи каждого из чисел γx и $\tilde{\gamma} x$ отличается от длины записи числа $\tilde{c}$ на константу. Кроме того, суммарное количество работ $n' = n+1+(2m-4)m+L(m-1)$ в примере $I(m, \lambda, C)$ полиномиально от n при фиксированных m и λ. Таким образом, длина записи примера $I(m, \lambda, C)$ при фиксированных m и λ является полиномом от длины записи примера C задачи РАЗБИЕНИЕ. Тем самым, доказана полиномиальная сводимость задачи РАЗБИЕНИЕ к задаче CHP(m, λ), и следовательно, NP-трудность последней.

Теорема 27.3 доказана.

Следствие 27.4 *Для минимальной функции $\eta_e(m)$, для которой класс примеров $OS(m, \eta_e(m))$ является полиномиально разрешимым, справедлива нижняя оценка*

$$\eta_e(m) \geq 2m-3. \qquad \square$$

При $\lambda = 1$ (т.е. при $L_{\max} = p_{\max}$) проблема распознавания свойства нормальности сводится к аналогичной проблеме с $m-1$ машинами. Следовательно, в этом случае свойство нормальности полиномиально распознается при $m = 3$ и является NP-трудной проблемой при $m > 3$.

Представляет интерес вопрос о соотношении между функциями $\eta_n(m)$ и $\eta_e(m)$. Так если для некоторого m имеет место соотношение $\eta_e(m) < \eta_n(m)$, то это означает, что при некоторых $\lambda \in (\eta_e(m), \eta_n(m))$ для класса примеров $OS(m, \lambda)$ существует полиномиальный алгоритм, гарантирующий построение оптимального расписания, которое при этом *не обязательно является нормальным*. И такой алгоритм был бы

несомненно интересен. В случае же, когда для некоторого m выполнено обратное соотношение: $\eta_n(m) < \eta_e(m)$, это означало бы, что для каждого примера из класса $OS(m,\lambda),\ \lambda \in (\eta_n(m), \eta_e(m))$ существует нормальное расписание S_{opt}, но не существует полиномиального алгоритма построения нормального расписания для любого примера из класса $OS(m,\lambda)$. В этом случае представляет интерес доказательство существования нормального расписания для любого примера из класса $OS(m,\lambda),\ \lambda \in (\eta_n(m), \eta_e(m))$, которое по своей природе должно быть неконструктивно.

28 Жадные алгоритмы построения допустимых расписаний с постоянными приоритетными порядками операций

В трех последующих разделах будет рассматриваться применение к задаче open shop таких жадных алгоритмов, в которых на протяжении всей работы алгоритма поддерживаются постоянные приоритетные порядки на множествах операций каждой машины и каждой работы. (Такие алгоритмы будем называть *жадными алгоритмами с приоритетами*.) Многие известные автору алгоритмы для задачи OS относятся к этому типу жадных алгоритмов и различаются лишь правилами построения приоритетных порядков. Так например, используемые в теоремах 25.3,26.1 алгоритмы точного решения задачи OS являются жадными, так как конструируют приоритетные порядки операций на машинах, задающие допустимые при определенных условиях приоритетные укладки. Таковыми же будут и алгоритмы $\mathcal{A}_{30.1}, \mathcal{A}_{30.2}$, которые мы будем строить в доказательствах теорем 30.2,30.3.

Уточним понятие жадного алгоритма с приоритетами применительно к задаче OS.

Пусть задан некоторый "приоритетный"порядок $(1,\dots,m)$ машин $M_i \in \mathcal{M}$, который для каждой работы J_j определяет приоритетный порядок на множестве ее операций $O^j : o_1^j \succ o_2^j \succ \cdots \succ o_m^j$. Пусть для каждой машины M_i также определен приоритетный порядок на множестве ее операций F_i. Поскольку в системе open shop каждая работа имеет единственную операцию на каждой машине M_i, то для задания приоритетного порядка на F_i достаточно указать "приоритетный" порядок выполняемых ею работ: $q^i = (q_1^i,\dots,q_n^i)$. *Жадным алгоритмом* для задачи OS будем называть алгоритм, строящий расписание на следующих принципах.

1. Машина не простаивает, если есть работа, готовая на ней выполняться.
2. Выбор очередной операции для освободившейся машины M_i определяется приоритетным порядком ее работ (q^i).
3. Выбор очередной операции освободившейся работы определяется приоритетным порядком машин $(1,\dots,m)$.

Таким образом, всякий жадный алгоритм можно представить состоящим из двух этапов, на первом из которых определяются приоритетный порядок машин $(1,2,\dots,m)$

и приоритетные порядки $\{q^i \mid i \in \mathbb{N}_m\}$ работ на машинах, а на втором конструируется само жадное расписание. Первый этап специфичен для каждого алгоритма и не поддается единому описанию. Идеология второго этапа для всех жадных алгоритмов идентична, однако возможны различные варианты его реализации, имеющие различные характеристики трудоемкости и требуемого объема памяти. (При этом возможен учет специфики приоритетных порядков, полученных на первом этапе.) Нетрудно предложить реализацию второго этапа алгоритма с трудоемкостью $O(nm^3)$, где для каждой из nm операций время ее старта в расписании будет находиться в худшем случае за $O(m^2)$ вычислительных операций. Наши дальнейшие усилия будут направлены на то, чтобы показать, что *для любых исходных приоритетных порядков* нахождение времени старта каждой операции можно реализовать с трудоемкостью $O(m)$, что даст оценку трудоемкости всего второго этапа $O(nm^2)$[1]. Объем требуемой для этого дополнительной памяти незначителен (существенно меньше объема входной информации).

Далее *вектором приоритетов* будем называть вектор $q = (q^1, \ldots, q^m)$, компонентами которого являются приоритетные порядки работ на машинах. Перейдем к детальному описанию работы жадного алгоритма на втором этапе.

Схема $\mathcal{A}_{\mathrm{GP}}$ жадного алгоритма
(построение жадного расписания для заданных приоритетных порядков работ на машинах)

В силу двойственности понятий "работа"–"машина" в задаче OS, не ограничивая общности можем считать выполненным соотношение $m \leq n$. Машины предполагаем занумерованными в соответствии с заданным на них приоритетным порядком.

На каждом шаге алгоритма будет известна *граница определенности* текущего (частичного) расписания S, т.е. такая величина $\hat{t} \geq 0$, что на последующих шагах алгоритма в интервале $[0, \hat{t})$ не будет происходить никаких изменений в расписании S. Это означает, что никаким операциям не будут приписываться времена их начала s_{ji} в интервале $[0, \hat{t})$. (К этому добавим, что алгоритм не переопределяет однажды назначенные времена s_{ji}.) В процессе работы алгоритма граница $\hat{t}$ будет скачкообразно перемещаться от нуля в положительном направлении, пока не достигнет момента $C_{\max}(S)$ окончания последней операции.

Через $j(i,t)$ будем обозначать номер работы, выполняемой в расписании S на машине M_i в момент времени t, $j(i,t) = 0$ означает, что машина M_i простаивает. Множество $j(t) = \cup_{i=1}^{m}\{j(i,t)\} \setminus \{0\}$ будем называть *фронтом работ в момент времени* t. Множество $j(\hat{t}) \doteq \hat{j}$ будем называть также *текущим фронтом работ*; номер работы текущего фронта, выполняемой на машине M_i, будем обозначать $\hat{j}(i)$. Алгоритм состоит из предварительного шага и nm последующих шагов, где в конце шага $k \geq 1$ будут известны k операций, заканчивающихся в расписании S до текущей границы $\hat{t}$. Кроме величины $\hat{t}$ и фронта $\hat{j}$ в конце каждого шага для каждой машины M_i, $i \in \mathbb{N}_m$, будет известен список $U_i(\hat{t}) = (u_1^i, u_2^i, \ldots, u_{a_i}^i)$ номеров работ (упорядоченных в соответствии

[1]Приведенная в монографии [61, стр.22] оценка трудоемкости жадного алгоритма $O(nm)$ представляется ошибочной.

с их приоритетным порядком q^i), моменты начала операций которых на машине M_i еще не определены; a_i — текущая длина списка $U_i(\hat{t})$.

Для уменьшения трудоемкости алгоритма введем два дополнительных массива. Целый массив $\Phi[1..n]$ при $\Phi[j]=0$ сообщает, что работа J_j не является работой текущего фронта $\hat{j}$. В противном случае $\Phi[j]$ сообщает номер машины, на которой выполняется работа $J_j \in \hat{j}$. Элементы битового массива $B[1..m, 1..m]$ принимают значения 0 или 1. Если $\hat{j}(i)=0$ (т.е. машина M_i простаивает), то $B[i,i']=1$, если и только если работа фронта $\hat{j}(i')$ входит в список U_i. (Таким образом, в случае $\hat{j}(i)=0$ строка i матрицы B содержит ровно a_i единиц.) Если $\hat{j}(i)\neq 0$, то значения $B[i,i']$ могут быть произвольны.

Предварительный шаг. Полагаем $a_i := n;\ U_i = (u_1^i,\ldots,u_n^i) := (q_1^i,\ldots,q_n^i) = q^i,\ i \in \mathbb{N}_m;\ \Phi \equiv 0;\ B \equiv 1;\ \hat{t} := 0.$

Для $i=1,\ldots,m$ в списке U_i находим работу $j=u_k^i$ с наименьшим номером k такую, что $\Phi[j]=0$, и включаем ее во фронт, исключая из списка U_i: $\hat{j}(i) := j;\ s_{ji} := 0;\ U_i := U_i \setminus \{j\};\ a_i := a_i - 1;\ \Phi[j] := i;\ B[i,i] := 0$. (Ясно, что порядок просмотра машин на этом шаге является существенным.)

Отметим свойство $\hat{t} \le s_{\hat{j}(i),i} + p_{\hat{j}(i),i}$, которое на каждом шаге алгоритма будет выполняться для любой работы текущего фронта $\hat{j}(i) \in \hat{j}$.

Трудоемкость предварительного шага равна $O(n+m^2)$.

Шаг k ($k=1,\ldots,nm$) состоит из трех логических действий.

Действие 1. Среди непростаивающих машин $\{i \mid \hat{j}(i)\neq 0\}$ находим машину с наименьшим номером $i=i'$, на которой достигается величина $t' = \min_i(s_{\hat{j}(i),i} + p_{\hat{j}(i),i})$, т.е. на которой раньше других заканчивается работа фронта $j' \doteq \hat{j}(i')$. Ясно, что $\hat{t} \le t'$. Положим $\hat{t} := t'$. Так как освободились одновременно работа (j') и машина (i'), то нужно найти очередную операцию как для работы j', так и для машины i', чему посвящены два последующих действия.

Действие 2. (Нахождение очередной операции работы j'.) Положим $\Phi[j'] := 0$. Просматриваем столбец i' матрицы B и отыскиваем все машины i такие, что

$$\hat{j}(i)=0,\ a_i>0,\ B[i,i']=1. \tag{28.1}$$

Если таковых нет, то переходим к действию 3. Пусть такие машины нашлись, и первая из них — машина i''. Это означает, что только что закончившаяся работа j' входит в список $U_{i''}$ простаивающей машины i''. Назначаем j' на машину $M_{i''}$: $\hat{j}(i'') := j';\ \Phi[j'] := i'';\ s_{j'i''} := \hat{t};\ U_{i''} := U_{i''} \setminus \{j'\}$ (так как машина $M_{i''}$ простаивала, то $a_{i''} \le m-1$, следовательно, работу j' в списке $U_{i''}$ мы найдем за $O(m)$ операций); $a_{i''} := a_{i''} - 1$. Для остальных машин M_i со свойством (28.1) полагаем $B[i,i'] := 0;\ B[i,i''] := 1$ (в строке i остается ровно a_i единиц).

Действие 3. (Нахождение очередной операции машины $M_{i'}$.) Положим $\hat{j}(i') := 0;\ B[i',i] := 0,\ \forall\ i \in \mathbb{N}_m$. Если $a_{i'} > 0$, то последовательно просматриваем список $U_{i'}$: $j := u_1^{i'}, u_2^{i'}, \ldots$. Если $\Phi[j] \neq 0$, то $B[i', \Phi[j]] := 1$. Если $\Phi[j]=0$, то прекращаем просмотр списка (нашлась работа j, не входящая во фронт $\hat{j}$). Полагаем $\Phi[j] := i';\ \hat{j}(i') := j;\ s_{ji'} := \hat{t};\ U_{i'} := U_{i'} \setminus \{j\};\ a_{i'} := a_{i'} - 1$. Наконец, если весь список $U_{i'}$ просмотрен и работа J_j с $\Phi[j]=0$ не найдена, то машина $M_{i'}$ становится

простаивающей, а строка i' в матрице B в точности перечисляет номера машин, по чьей вине $M_{i'}$ простаивает.

Заметим, что в последнем случае имеем $a_{i'} \leq m-1$, поэтому действие 3 требует $O(m)$ вычислений. Заметим также, что факт включения работы J_j во фронт не требует каких-то изменений в массиве B. Это включение могло бы добавить единицу в какую-то строку i массива B (если $j \in U_i$), но для простаивающей машины M_i этого произойти не может (так как количество единиц в такой строке i уже максимально — равно a_i), а для непростаивающих машин M_i значения $B[i,\cdot]$ для нас несущественны.

Мы завершили описание Этапа 2 жадного алгоритма.

Нетрудно убедиться, что на любом шаге $k = 1, \ldots, nm$ каждое из трех действий требует не более $O(m)$ вычислений, что дает общую оценку трудоемкости Этапа 2, равную $O(nm^2)$.

Объем памяти, отводимой в машине для входной информации (mn чисел $\{p_{ji}\}$), для готового расписания (mn чисел $\{s_{ji}\}$) и, возможно, для вектора приоритетов q и списка приоритетов машин, с добавлением массивов Φ и B возрастает незначительно (на n целых ячеек и m^2 битовых, что с учетом принятого нами соглашения $m \leq n$ не превосходит по порядку объема исходной информации). Таким образом, нами доказана

Лемма 28.1 *Для любых приоритетных порядков, полученных на Этапе 1 жадного алгоритма решения задачи* OS*, возможна реализация его второго этапа с трудоемкостью $O(nm^2)$ при объеме памяти, равном (по порядку) объему входной информации* □

Заметим, что без использования массивов Φ и B для назначения новой работы на машину $M_{i'}$ (т.е. для отыскания первой работы в списке $U_{i'}$, отличной от всех работ фронта) пришлось бы затратить $O(m^2)$ операций, сравнивая каждую очередную работу списка $U_{i'}$ со всеми работами фронта. Точно так же, отыскание простаивающей машины, имеющей в своем списке работу $J_{j'}$, потребовало бы $O(m^2)$ операций, что привело бы к общей оценке трудоемкости $O(nm^3)$.

29 Свойства жадных расписаний

Пусть задано жадное расписание S, определенное вектором приоритетов q. Для двух операций $o_i^j, o_i^{j'}$ машины M_i запись $o_i^j \succ_p o_i^{j'}$ означает больший приоритет операции o_i^j. Интервал выполнения операции o' в расписании S обозначим $I(o')$. Для двух интервалов $I_1 = [a_1, b_1), I_2 = [a_2, b_2)$ отношение $I_1 \prec I_2$ означает $b_1 \leq a_2$. Если для двух операций машины M_i выполняется $o_i^j \succ_p o_i^{j'} \& I(o_i^j) \succ I(o_i^{j'})$, т.е. операция $o_i^{j'}$ выполняется раньше, чем более приоритетная операция o_i^j, то интервал $I(o_i^{j'})$ будем называть o_i^j-*задержкой*. Операцию $o_{i'}^j$ работы J_j, предшествующую операции o_i^j, назовем o_i^j-*предоперацией*.

Если интервал $I(o_{i'}^j)$ некоторой o_i^j-предоперации начинается раньше некоторой o_i^j-задержки $I(o_i^{j'})$, то последнюю будем называть *покрываемой* — в случае $I(o_i^{j'}) \subseteq I(o_{i'}^j)$,

и *полупокрываемой* — в случае $I(o_i^{j'}) \not\subseteq I(o_{i'}^j) \& I(o_i^{j'}) \cap I(o_{i'}^j) \neq \emptyset$. (В последнем случае лишь начальный отрезок o_i^j-задержки содержится в $I(o_{i'}^j)$.)

Интервал простоя машины M_i, предшествующий интервалу $I(o_i^j)$, назовем *o_i^j-простоем*.

В приводимых ниже простых утверждениях формулируются свойства жадного расписания.

Утверждение 29.1 *Всякий o_i^j-простой покрывается интервалами выполнения o_i^j-предопераций* □

Утверждение 29.2 *Всякая o_i^j-задержка покрывается или полупокрывается единственной o_i^j-предопорацией* □

Утверждение 29.3 *Для всякой o_i^j-предопорации существует не более одной полупокрываемой ею o_i^j-задержки* □

Далее для упрощения формулировок считаем $p_{\max} = 1$. Обозначим:
$k'_{ji}(t)$ — количество o_i^j-предопераций, начинающихся строго до момента t;
$k''_{ji}(t)$ — количество o_i^j-предопераций, заканчивающихся строго до момента t.
Из утверждений 29.1–29.3 вытекает

Утверждение 29.4 *Для любой операции o_i^j и любого момента $t \leq s_{ji}$ суммарная длина o_i^j-задержек и o_i^j-простоев в их пересечении с интервалом $[0, t)$ не превосходит суммарной длины o_i^j-предопераций в их пересечении с интервалом $[0, t)$ плюс $k''_{ji}(t)$.*

Доказательство. Суммарная длина o_i^j-простоев и покрываемых o_i^j-задержек, очевидно, не превосходит суммарной длины покрывающих их o_i^j-предопераций. Суммарная длина полупокрываемых o_i^j-задержек (ввиду соглашения $p_{\max} = 1$) не превосходит их количества, которое, в силу утверждения 29.2, не превосходит $k''_{ji}(t)$ □

Оценивая суммарную длину o_i^j-предопераций в интервале $[0, t)$ через их количество, из утверждения 29.4 получаем

Утверждение 29.5 *Для любой операции o_i^j и любого момента $t \leq s_{ji}$ суммарная длина o_i^j-задержек и o_i^j-простоев в их пересечении с интервалом $[0, t)$ не превосходит $k'_{ji}(t) + k''_{ji}(t)$* □

Обозначим: $D_i(t)$ — величина простоя машины M_i в интервале $[0, t]$; $w_i(t)$ — величина чистой работы машины M_i после момента t; C_i — момент завершения машиной M_i последней операции; $D_i \doteq D_i(C_i)$. Тогда, очевидно,

$$C_{\max}(S) = \max_i C_i = \max_i (D_i + L_i).$$

Из утверждения 29.1 вытекают

Утверждение 29.6 $D_i \le m-1,\ \forall\ i \in \mathbb{N}_m$ □

Утверждение 29.7 $C_i = L_i + D_i \le L_i + (m-1),\ \forall\ i \in \mathbb{N}_m$ □

Из последнего утверждения следует оценка (24.8) на длину любого жадного расписания. Для удобства формулировок последующих утверждений дадим несколько определений.

Машину, заканчивающую работу последней, будем называть *критической* и обозначать $M_{i^*}:\ C_{\max}(S) = C_{i^*}$. Через $\tilde{t}_i$ обозначим момент начала первого простоя на машине M_i. Операции, выполняемые на машине M_i после момента $\tilde{t}_i$, назовем *постоперациями*. Операции, являющиеся постоперациями на критической машине (равно как и работы, которым они принадлежат), назовем *критическими*. Операции критических работ, покрывающие начальный ε-интервал $[\tilde{t}_{i^*}, \tilde{t}_{i^*} + \varepsilon)$ простоя критической машины, назовем *предкритическими*. Из утверждения 29.7 непосредственно следует

Утверждение 29.8 *Если $\delta(2) \ge m-1$, то машина M_1 является критической.* □

По утверждению 29.1, для каждой критической работы существует предкритическая операция, причем единственная. Поскольку интервалы выполнения любых двух предкритических операций пересекаются, то все предкритические операции выполняются на разных машинах, следовательно, имеется не более $(m-1)$ предкритических операций и предкритических работ. А поскольку в системе open shop каждая работа имеет на каждой машине ровно одну операцию, то справедливо

Утверждение 29.9 *На каждой машине в жадном расписании S может быть не более $(m-1)$ постопераций* □

Из утверждения 29.9 следует

Утверждение 29.10 $\tilde{t}_i \ge L_i - (m-1),\ \forall\ M_i \in \mathcal{M}$ □

В следующих трех утверждениях мы исследуем вопрос, насколько время старта s_{ji} какой-либо операции o_i^j в жадном расписании S, построенном для заданных приоритетных порядков $(q^1, \ldots, q^m) = q$, может отличаться от времени старта этой операции в приоритетной укладке $S^q = \{\hat{s}_i(j)\}$, определенной для тех же приоритетных порядков.

Отклонение в меньшую сторону ($s_{ji} < \hat{s}_i(j)$) возможно в том случае, когда операция o_i^j обгоняет несколько более приоритетных операций машины M_i. Как следует из описания этапа 2 жадного алгоритма, приведенного в разделе 28, выбор очередной операции на освободившуюся машину M_i делается из не более чем m первых операций списка U_i, поэтому операция o_i^j может обогнать не более $(m-1)$ более приоритетных операций. Все остальные более приоритетные операции к моменту назначения времени s_{ji} уже выполнены, поэтому справедливо

Утверждение 29.11 $s_{ji} \ge \hat{s}_i(j) - (m-1)$ □

Отмечая, что за вычетом o_i^j-задержек и o_i^j-простоев все остальное время в интервале $[0,t)$ для $t \le s_{ji}$ заполнено интервалами выполнения более приоритетных чем o_i^j операций, с учетом утверждения 29.5 можем выписать следующую оценку момента начала операции o_i^j в приоритетной укладке.

Утверждение 29.12 *Для любой операции o_i^j и момента времени $t \le s_{ji}$ справедлива оценка:* $\hat{s}_i(j) \ge t - k'_{ji}(t) - k''_{ji}(t)$ □

Из утверждения 29.12 при $t = s_{ji}$ вытекает следующее соотношение между s_{ji} и $\hat{s}_i(j)$.

Утверждение 29.13 $s_{ji} \le \hat{s}_i(j) + 2m - 2$ □

Следующее утверждение потребует небольшого доказательства.

Утверждение 29.14 *Пусть $m \le 3$ и задано произвольное жадное расписание S. Пусть в этом расписании какая-то из машин (M_i) имеет $m-1$ постопераций. Тогда одна из оставшихся машин до момента $\tilde{t}_i$ работает без простоев.*

Доказательство. Пусть, для определенности, $i = 1$ и $o_1^2, \ldots, o_1^m$ — постоперации машины M_1. Предшествующий им простой на машине M_1 в интервале $[\tilde{t}_1, \tilde{t}_1 + \varepsilon)$ покрывается предкритическими операциями работ $J_2, \ldots, J_m$. Пусть это операции $o_2^2, \ldots, o_m^m$, и предположим, что в расписании S имеются интервалы простоя, предшествующие моменту $\tilde{t}_1$. Из них рассмотрим тот интервал I', что оканчивается последним. Пусть он принадлежит машине M_2. Из утверждения 29.1 и соотношений

$$I' \prec I(o_2^2) \prec I(o_1^2) \tag{29.1}$$

следует, что

$$I' \subseteq I(o_3^2) \cup \ldots \cup I(o_m^2), \tag{29.2}$$

что невозможно при $m = 2$, поэтому при $m = 2$ утверждение 29.14 доказано. При $m = 3$ из (29.1) и (29.2) следует, что операция o_3^2 является первой операцией работы J_2 в расписании S, поэтому, по утверждению 29.1, на машине M_3 не существует простоев, предшествующих $I(o_3^2)$. Так как их нет и в интервале между $I(o_3^2)$ и $\tilde{t}_1$ (по определению I'), то до момента $\tilde{t}_1$ машина M_3 не простаивает □

30 Жадные алгоритмы точного и приближенного решения задачи open shop

Замечание 30.1 *При построении описываемых ниже алгоритмов $\mathcal{A}_{30.1}$ и $\mathcal{A}_{30.2}$ будут использоваться жадные алгоритмы. Их реализация на втором этапе предполагается по схеме $\mathcal{A}_{\mathrm{GP}}$ изложенной в разделе 28, поэтому для каждого из них достаточно будет описать лишь первый этап — построение приоритетных порядков.*

Теорема 30.2 *Класс примеров задачи* OS *с* m *машинами и* n *работами, задаваемый соотношениями*

$$\delta(2) \geq m-1, \tag{30.1}$$

$$L_{\max} \geq 5.45m - 7, \tag{30.2}$$

является эффективно нормальным. Оптимальное расписание для каждого из этих примеров находится жадным алгоритмом $\mathcal{A}_{30.1}$ *с трудоемкостью* $O(n^2m^2)$.

(*При нарушении какого-либо из условий* (30.1), (30.2) *алгоритм* $\mathcal{A}_{30.1}$, *как и всякий жадный алгоритм, гарантирует оценку точности* (24.8).)

Доказательство. При $m = 2$ и $m = 3$ утверждение теоремы следует, соответственно, из свойств расписания Гонзалеза и Сани и теоремы 24.1. Пусть $m \geq 4$. Покажем, что описываемый ниже алгоритм $\mathcal{A}_{30.1}$ строит допустимое расписание S длины $L_{\max}$. Ввиду замечания 30.1, достаточно описать лишь первый этап алгоритма.

Этап 1 алгоритма $\mathcal{A}_{30.1}$

Для каждой работы $J_j,\ j \in \mathbb{N}_n$ определен вектор $x_j = (p_{j1}, \ldots, p_{jm}) \in [0,1]^m$ Ясно,что $\sum_{j=1}^{n} x_j = (L_1, \ldots, L_m)$. Согласно следствию 11.6 (стр. 74) для вектора $a = 0$ за $O(n^2m^2)$ операций находим перестановку $\pi = (\pi_1, \ldots, \pi_n)$ такую, что

$$\sum_{j=1}^{l} p_{\pi_j i} = \frac{(l-m+1)}{n} L_i + \lambda_i^l, \tag{30.3}$$

где

$$\lambda_i^l \in \left[-\frac{1}{m}, m-1\right],\ \forall\ l \in \mathbb{N}_n; i \in \mathbb{N}_m. \tag{30.4}$$

Далее считаем, что работы занумерованы согласно перестановке π, т.е. $\pi = (1, 2, \ldots, n)$. Для каждой из машин $M_i \in \mathcal{M}$ выбираем вектор приоритетов:

$$q^1 := (1, 2, \ldots, n); \quad q^i = (q_1^i, \ldots, q_n^i) := (n, n-1, \ldots, 1),\ \forall\ i \neq 1.$$

Алгоритм $\mathcal{A}_{30.1}$ описан. ■

Из утверждения 29.8 и соотношения (30.1) следует, что машина M_1 является критической в расписании S, т.е. $C_{\max}(S) = C_1$. Таким образом, для доказательства свойства нормальности оптимального расписания достаточно будет показать, что $D_1 = 0$, т.е. что M_1 не простаивает.

Предположим противное, и пусть $\tilde{t} = \tilde{t}_1$ — начало простоя на M_1; J_j — произвольная критическая работа; $o_1^j, o_{i'}^j$ — критическая и предкритическая операции этой работы. Так как $k'_{j1}(\tilde{t}) \leq m-1,\ k''_{j1}(\tilde{t}) \leq m-2$, то по утверждению 29.12,

$$\hat{s}_1(j) \geq \tilde{t} - (2m-3) \doteq t^*. \tag{30.5}$$

Из $k'_{ji'}(s_{ji'}) \le m-2$, $k''_{ji'}(s_{ji'}) \le m-2$ и $s_{ji'} > \tilde{t}-1$ также по утверждению 29.12 получаем:

$$\hat{s}_{i'}(j) \ge s_{ji'} - (2m-4) > \tilde{t} - (2m-3) = t^*. \tag{30.6}$$

Пусть k' — такое максимальное число k, что

$$\hat{s}_i(q^i_k) = \sum_{j=n-k+2}^{n} p_{ji} \le t^*, \ \forall\ i \ne 1. \tag{30.7}$$

По условию (30.6), работы $\{q^i_1, \ldots, q^i_{k'}\} = \{n-k'+1, \ldots, n\}$ не могут быть критическими, следовательно, таковыми могут быть лишь работы $\{1, 2, \ldots, n-k'\}$. По выбору k' известно, что при $k = k'+1$ для некоторого $i = i'$ условие (30.7) нарушается, т.е. выполнено $\sum_{j=n-k'+1}^{n} p_{ji'} > t^*$, или

$$\sum_{j=1}^{n-k'} p_{ji'} < L_{i'} - t^*. \tag{30.8}$$

С другой стороны, так как из (30.5) каждая критическая операция o^j_1 начинается в приоритетной укладке S^q не раньше t^* и их суммарная длина равна $(L_{\max} - \tilde{t})$, то последняя из них заканчивается в S^q не раньше момента $t^* + (L_{\max} - \tilde{t})$, и тем более не раньше этого момента заканчивается в S^q последняя из операций множества $\{o^1_1, \ldots, o^{n-k'}_1\}$, включающего все критические операции. Таким образом,

$$\sum_{j=1}^{n-k'} p_{j1} \ge t^* + (L_{\max} - \tilde{t}) = L_{\max} - (2m-3). \tag{30.9}$$

Из (30.3) для $l = n-k'$, $i \in \{i', 1\}$ и из (30.8) получаем

$$\sum_{j=1}^{l} p_{ji'} = \frac{l-m+1}{n} L_{i'} + \lambda^l_{i'} < L_{i'} - t^*,$$

$$\frac{l-m+1}{n} < 1 - \frac{t^* + \lambda^l_{i'}}{L_{i'}};$$

$$\sum_{j=1}^{l} p_{j1} = \frac{l-m+1}{n} L_{\max} + \lambda^l_1 < L_{\max} + \lambda^l_1 - \frac{L_{\max}}{L_{i'}}(t^* + \lambda^l_{i'}).$$

С учетом (30.9), это дает

$$L_{\max} - (2m-3) < L_{\max} + \lambda^l_1 - \frac{L_{\max}}{L_{i'}}(t^* + \lambda^l_{i'}),$$
$$L_{\max}(t^* + \lambda^l_{i'}) < L_{i'}(\lambda^l_1 + 2m - 3),$$

откуда с учетом (30.4),(30.5),(30.1) и утверждения 29.10 получаем

$$L_{\max}(\tilde{t}-(2m-3)-\frac{1}{m})<(L_{\max}-(m-1))(3m-4),$$
$$L_{\max}(L_{\max}-(3m-4+\frac{1}{m}))<(L_{\max}-(m-1))(3m-4),$$
$$L_{\max}^2-(6m-8+\frac{1}{m})L_{\max}+(m-1)(3m-4)<0.$$

Из решения квадратного уравнения при $m\geq 4$ вытекает необходимое требование

$$\begin{aligned} L_{\max} &< 3m-4+\frac{1}{2m}+\sqrt{\left(3m-4+\frac{1}{2m}\right)^2-(3m-4)(m-1)} < \\ &< 3m-4+\frac{1}{2m}+\sqrt{6}\left(m-\frac{17}{12}+\frac{3}{12m-17}\right) < 5.45m-7, \end{aligned}$$

которое не выполняется ввиду (30.2). Таким образом, наше предположение о существовании простоя на машине M_1 оказалось неверным, что доказывает свойство нормальности построенного алгоритмом расписания S.

Теорема 30.2 доказана.

Теорема 30.3 *Для задачи open shop с n работами и m машинами существует жадный алгоритм $\mathcal{A}_{30.2}$, который с трудоемкостью $O(nm^2)$ для любого входа задачи строит расписание S с оценкой длины*

$$C_{\max}(S)\leq L_{\max}+\left(\left\lceil\frac{(3m^2-2m)p_{\max}}{L_{\max}+2mp_{\max}}\right\rceil-1\right)p_{\max}. \tag{30.10}$$

Доказательство. Будем предполагать выполненными (22.9) и (22.10). В силу замечания 30.1, достаточно описать лишь первый этап алгоритма $\mathcal{A}_{30.2}$.

Этап 1 алгоритма $\mathcal{A}_{30.2}$

Пусть $F_i=\{o_i^j\mid j\in\mathbb{N}_n\}$ — множество операций машины M_i,

$$l\doteq\left\lceil\frac{3m^2-2m}{L_{\max}+2m}\right\rceil-1. \tag{30.11}$$

Если $L_{\max}<m-2$, то оценке (30.10) будет удовлетворять любое расписание S, полученное жадным алгоритмом (в силу (24.8)). Поэтому далее будем предполагать $L_{\max}\geq m-2$, что обеспечит свойство $m-l-1\geq 0$, необходимое для корректности выражений (30.12), (30.13).

Все множество операций ($\mathcal{O}$) разобьем на два подмножества: $\mathcal{O}=\mathcal{O}_1\cup\mathcal{O}_2$, — таким образом, что

$$\left|\mathcal{O}_1\cap O^j\right|=m-l-1,\ \forall\ j\in\mathbb{N}_n; \tag{30.12}$$

$$\left| \sum_{o_i^j \in \mathcal{O}_1 \cap F_i} p_{ji} - \frac{m-l-1}{m} L_{\max} \right| < 1, \ \forall\ i \in \mathbb{N}_m. \tag{30.13}$$

(Согласно лемме 10.14, такое разбиение можно получить с трудоемкостью $O(nm^2)$.) Для каждой машины $M_i,\ i \in \mathbb{N}_m$, в приоритетном порядке q^i идут сначала все операции из $\mathcal{O}_1$ (в произвольном порядке), а затем — все остальные.

Алгоритм $\mathcal{A}_{30.2}$ описан. ■

Из описания алгоритма $\mathcal{A}_{30.2}$ и леммы 28.1 следует, что его трудоемкость на обоих этапах равна $O(nm^2)$, а значит, такова и трудоемкость алгоритма в целом. Остается доказать, что работающий на определенных выше приоритетных порядках жадный алгоритм строит расписание S, удовлетворяющее оценке (30.10).

Предположим противное. Тогда $D_{i^*} > l$, где M_{i^*} — критическая машина. Пусть $o_{i^*}^j$ — операция, выполняемая последней в расписании S. По утверждению 29.1, не менее $(l+1)$ $o_{i^*}^j$-предопераций участвует в покрытии $o_{i^*}^j$-простоев машины M_{i^*}. Назовем такие операции "*покрывающими*", и пусть $o_{i'}^j$ — предкритическая $o_{i^*}^j$-предоперация. По определению, она покрывает первый $o_{i^*}^j$-простой и для нее выполнено

$$\tilde{t} - 1 < s_{ji'} \le \tilde{t} < s_{ji'} + p_{ji'}. \tag{30.14}$$

Так как $k''_{ji^*}(s_{ji'}) \le m-l-2,\ k'_{ji^*}(s_{ji'}) \le m-l-2$, то по утверждению 29.12, критическая операция $o_{i^*}^j$ и все покрывающие операции начинаются в приоритетной укладке не раньше момента

$$\dot{t} \doteq s_{ji'} - k'_{ji^*}(s_{ji'}) - k''_{ji^*}(s_{ji'}) \ge s_{ji'} - 2(m-l-2). \tag{30.15}$$

В то же время, согласно (30.12), (30.13) и правилу построения приоритетных порядков, $m-l-1$ операций работы J_j заканчивается в S^q раньше момента $\frac{m-l-1}{m} L_{\max} + 1 \doteq \bar{t}$. Так как сумма мощностей этих двух множеств операций не меньше $(l+2)+(m-l-1) = m+1 > m$, то существует "общая" операция $(o_{i_0}^j)$, входящая в оба эти множества, т.е. начинающаяся в S^q не раньше $\dot{t}$ и заканчивающаяся раньше $\bar{t}$. Очевидно, ее длина p_{ji_0} меньше длины интервала $(\dot{t}, \bar{t})$, поэтому справедлива цепочка соотношений:

с учетом (30.11), (30.14) оценивается величиной

$$p_{ji_0} < \bar{t} - \dot{t} \le L_{\max} - \frac{l+1}{m} L_{\max} + 1 + 2(m-l-2) - s_{ji'} \le (\text{из (30.11)})$$

$$\le L_{\max} - (m-1) - s_{ji'} = \tilde{t} + \Sigma_{cr} - (m-1) - s_{ji'} \le \tilde{t} - s_{ji'} < (\text{из (30.14)}) < 1. \tag{30.16}$$

Здесь Σ_{cr} — сумма длительностей критических операций, которых, как известно по утверждению 29.9, не более $m-1$.

Так как разница между крайними левой и правой частями (30.16) меньше 1, то оценка (30.15) на величину $\dot{t}$ точна, т.е. покрывающих операций ровно $l+1$. Далее

покажем, что "общей" операцией не может быть критическая операция $o^j_{i^*}$. Действительно, предполагая противное и обозначая через Σ'_{cr} сумму длительностей остальных критических операций, из (30.16) получим

$$p_{ji^*} < \tilde{t} + p_{ji^*} + \Sigma'_{cr} - (m-1) - s_{ji'} \le \tilde{t} + p_{ji^*} - s_{ji'} - 1,$$

откуда $\tilde{t} - s_{ji'} > 1$, что противоречит (30.14).

Из (30.16) и (30.14) следует, что операция $o^j_{i'}$ также не может быть "общей".

Наконец, пусть $o^j_{i_0}$ — покрывающая операция, отличная от $o^j_{i'}$. Так как в покрытии $o^j_{i^*}$-простоев участвует лишь часть интервала $I(o^j_{i'})$, не превосходящая по длине $1 - (\tilde{t} - s_{ji'})$, а длина покрывающей операции $o^j_{i_0}$ меньше $\tilde{t} - s_{ji'}$, то суммарная длина всех $o^j_{i^*}$-простоев меньше l, что противоречит предположению $D_{i^*} > l$. Таким образом, предположение оказалось неверным, что доказывает оценку (30.10).

Теорема 30.3 доказана.

Отметим несколько частных случаев теоремы 30.3, представляющих интерес.

Следствие 30.4

1. Если для целого $i \in \mathbb{N}_{m-1}$ *выполняется*

$$L_{\max} \ge m - 2 + 3i + \frac{(3i-2)i}{m-i}, \ \textit{то} \quad C_{\max}(S) \le L_{\max} + m - 1 - i.$$

В частности, если

$$L_{\max} \ge m + 1 + \frac{1}{m-1}, \ \textit{то} \quad C_{\max}(S) \le L_{\max} + m - 2,$$

что на единицу улучшает оценку (24.8). Для трех машин это дает условие:

если $L_{\max} \ge 4\frac{1}{2}$, *то* $C_{\max}(S) \le L_{\max} + 1$.

Если $L_{\max} \ge m + 4 + \frac{8}{m-2}$, *то* $C_{\max}(S) \le L_{\max} + m - 3$.

2. Если $L_{\max} \ge 4m - 4$, *то* $C_{\max}(S) \le L_{\max} + (m-1)/2$.

3. Если $L_{\max} \ge 7m - 6$, *то* $C_{\max}(S) \le L_{\max} + (m-1)/3$.

4. Если $L_{\max} \ge \frac{3}{4}m^2 - \frac{5}{2}m$, *то* $C_{\max}(S) \le L_{\max} + 3$ □

31 Схема $\mathcal{A}_{\mathrm{GC}}$ жадной достройки расписания многопроцессорной системы открытого типа

Описываемая ниже схема $\mathcal{A}_{\mathrm{GC}}$ построения допустимого расписания также активно использует жадные принципы, определенные в п.9 введения к данной главе (стр. 162). Однако в отличие от схемы $\mathcal{A}_{\mathrm{GP}}$, изложенной в разделе 28, мы не будем затрачивать усилий на поддержание постоянных приоритетных порядков на множествах операций

каждой работы и каждой машины, что позволит нам сэкономить на трудоемкости алгоритма. С другой стороны, построение расписания ведется в более сложных условиях, когда приходится учитывать наличие уже имеющегося частичного расписания. (В схеме $\mathcal{A}_{\mathrm{GC}}$ предполагается, что множество операций, для которых задано частичное расписание, может быть произвольным.) Это накладывает дополнительные ограничения на способы построения искомого расписания, и соответственно, требует дополнительных затрат трудоемкости. Однако благодаря предложенной ниже схеме реализации жадных принципов построения расписания удается сохранить низкую трудоемкость алгоритма.

В последующих разделах схема $\mathcal{A}_{\mathrm{GC}}$ будет применяться дважды: в разделе 32 — в алгоритме последовательной достройки нормального расписания, и в несколько модифицированной версии — в разделе 35, при построении аппроксимационной схемы для многопроцессорной задачи open shop. В разделе 32 частичное расписание будет предполагаться заданным для первых k машин, и мы достраиваем его на следующих $(l-k)$ машинах за время $O(nl(l-k))$. (Последовательное применение этого метода до построения полного расписания на m машинах дает суммарную оценку трудоемкости $O(nm^2)$, совпадающую с оценкой трудоемкости построения расписания по схеме $\mathcal{A}_{\mathrm{GP}}$.) В разделе же 35 частичное расписание задается для некоторого подмножества работ (так называемых “больших” работ), и мы достраиваем его на всех машинах для остальных работ.

Перейдем к детальному описанию работы алгоритма по схеме $\mathcal{A}_{\mathrm{GC}}$. (Аббревиатура “GC” происходит от английского “greedily completing”, — жадно достраивающий.)

* * *

Предположим, что у нас имеется r-стадийная многопроцессорная система открытого типа с n работами и m_i параллельными идентичными процессорами на i-й стадии ($i \in \mathbb{N}_r$), $m = \sum m_i$ — общее число машин, и пусть задано частичное расписание для этой системы, т.е. для некоторых операций $\{o_i^j\}$ (будем называть их “старыми”) уже заданы обслуживающие их процессоры $M(o_i^j)$ и моменты $\{s_i^j\}$ начала их выполнения. Требуется назначить исполнителей и моменты начала для оставшихся (“новых”) операций (будем называть это “назначить операцию”) без изменения уже имеющегося расписания, с тем чтобы длина полученного расписания была по возможности минимальной. Здравой идеей является как можно более полное заполнение промежутков простоя между старыми операциями, и мы предлагаем делать это следующим алгоритмом жадного типа.

Текущее (частичное) расписание будем хранить в двух матрицах размера $r \times n$: в матрице M будет задаваться процессор, обслуживающий операцию o_i^j, $(i \in \mathbb{N}_r,\ j \in \mathbb{N}_n)$, а в матрице s — момент начала операции o_i^j. (Если для операции o_i^j расписание еще не назначено, то $M(o_i^j) = \mathbf{nul}$.) Кроме того, все уже назначенные в расписание операции будем хранить в виде "списка операций"$\mathcal{L}$. Предполагаем, что возможен прямой доступ к любому элементу списка $\mathcal{L}$, если известен его абсолютный адрес. Кроме того, при каждом элементе из $\mathcal{L}$ будут храниться ссылки на адреса других элементов списка. В частности, для каждой операции o_i^j в списке $\mathcal{L}$ хранится номер работы j, которой принадлежит эта операция, номер обслуживающего процессора $M(o_i^j)$, тип операции

(старая или новая), момент ее окончания $c(o_i^j)$, а также адреса́ предыдущей и последующей операций работы J_j в списке $\mathcal{L}$, адреса́ предыдущей и последующей операций машины $M(o_i^j)$ в списке $\mathcal{L}$ и адреса́ предыдущего и последующего элементов списка $\mathcal{L}$ по ключу $c(o_i^j)$.

В процессе работы алгоритма мы один раз просматриваем весь список $\mathcal{L}$ по ключу c, в подходящие моменты назначая в расписание какие-то новые операции и добавляя их в список $\mathcal{L}$. Текущую просматриваемую операцию списка $\mathcal{L}$ называем *актуальной* операцией. Процесс просмотра сопровождается скачкообразным неубывающим изменением значения переменной τ, которую называем *текущим моментом*. Значение τ всегда совпадает с моментом окончания актуальной операции. *Текущей операцией* работы J_j (машины M_i) называем ближайшую (по ключу c) еще не просмотренную операцию работы J_j (машины M_i) из списка $\mathcal{L}$, — не имеет значения, выполняется ли она в момент τ или еще не начата. Ближайшую (по ключу c) непросмотренную старую (новую) операцию из списка $\mathcal{L}$ будем коротко называть *ближайшей старой/новой операцией*. В ходе работы алгоритма мы храним адреса обеих ближайших операций, а также поддерживаем массивы $\Phi[1\ldots n]$ и $\hat{j}[1\ldots m]$ адресов текущих операций каждой работы J_j $(j \in \mathbb{N}_n)$ и каждой машины M_i $(i \in \mathbb{N}_m)$.

Информацию о новых, уже назначенных в расписание (и поставленных в список $\mathcal{L}$), но еще не просмотренных операциях (будем называть их "полуопределенными") храним в виде "кучи" $\mathcal{H}$, полуупорядоченной по ключу c: в корневой вершине этой кучи всегда находится операция с наименьшим значением ключа, и по каждой цепочке, идущей из корня вниз, значение ключа неубывает. При каждой вершине кучи хранится пара: адрес операции в списке $\mathcal{L}$ и значение ключа. "Полуопределенность" операции в списке $\mathcal{L}$ означает, что у нее определены все параметры, кроме двух: ссылки на предыдущую и ссылки на следующую операции списка $\mathcal{L}$ по ключу c. Эти ссылки становятся известными после "просмотра" операции, т.е. после назначения ее актуальной операцией.

В каждый момент времени все новые неназначенные операции хранятся в трех списках: списке A_i операций, *разрешенных* для назначения на машину M_i $(i \in \mathbb{N}_m)$, а также списках R_j^J $(j \in \mathbb{N}_n)$ и R_i^M $(i \in \mathbb{N}_m)$ операций работы J_j (машины M_i), временно запрещенных для назначения их в расписание. Операция o' попадает в список запрещенных после ее удаления из какого-либо списка A_i по причине неизбежного перекрытия o' с текущей операцией работы J_j (текущей старой операцией машины M_i) в случае назначения o' в текущий момент. Для каждой операции o', перемещенной из списка A_i в какой-то список R_j^J, запоминается машина $M(o')$, из списка которой она была взята. (Для операций, помещенных в список R_i^M, этого не требуется, так как все операции в R_i^M взяты из A_i.) Поскольку алгоритм работает по жадному принципу, то простой машины M_i в течение какого-то интервала времени возможен лишь при пустом списке A_i. (Отсюда следует, что если в какой-то момент τ список A_i не пуст, то машина M_i не простаивает.)

Алгоритм состоит из нулевого шага (или шага *инициализации*) и цикла пошагового просмотра элементов списка $\mathcal{L}$ по неубыванию ключа c.

На шаге **Инициализация** задаются начальные значения списков и переменных. Полагается $\tau := 0$. Список $\mathcal{L}$ состоит только из старых операций. При этом в $\mathcal{L}$ добавляют-

ся m "старых"операций $\tilde{o}_1, \ldots, \tilde{o}_m$, относящихся к фиктивным работам $J_{n+1}, \ldots, J_{n+m}$, выполняемых соответственно на машинах $M_1, \ldots, M_m$ и имеющих моменты окончания $c(\tilde{o}_i) = 0$. Все новые операции каждого цеха $\mathcal{M}_\nu$ ($\nu \in \mathbb{N}_r$) находятся в списке $\mathcal{O}_\nu$. Для каждой машины $M_i \in \mathcal{M}_\nu$ полагается $A_i := \mathcal{O}_\nu$. (Таким образом, каждая операция $o^j_\nu \in \mathcal{O}_\nu$ представлена в m_ν независимых списках A_i.) Списки R^J_j ($j \in \mathbb{N}_n$), R^M_i ($i \in \mathbb{N}_m$) и $\mathcal{H}$ сначала пусты. Находятся адреса́ текущих операций всех работ (массив $\Phi[1 \ldots n]$) и всех машин (массив $\hat{j}[1 \ldots m]$), а также адрес ближайшей старой операции. Для получения всей этой информации достаточно одного просмотра списка $\mathcal{L}$ по ключу c.

Цикл просмотра списка $\mathcal{L}$ состоит из шагов, на каждом из которых просматривается только одна (актуальная) операция из этого списка. Каждый шаг начинается с определения новой актуальной операции. Она выбирается среди двух операций: ближайшей старой и ближайшей новой, адрес которой хранится в корневой вершине кучи $\mathcal{H}$. (На первом шаге цикла, когда куча $\mathcal{H}$ еще пуста, выбор ограничивается единственной ближайшей старой операцией.) Одновременно изменяется значение τ. Если выбрана старая операция, то она сразу же перестает быть “ближайшей старой”, а в качестве таковой берется следующая (по ключу c) операция из списка $\mathcal{L}$. Если же выбрана новая операция, то сначала доопределяются ее ссылки на предыдущую и последующие операции из списка $\mathcal{L}$ (в качестве таковых берутся предыдущая актуальная и ближайшая старая операции, заодно корректируются ссылки у этих двух операций), а затем операция удаляется из корня кучи $\mathcal{H}$ и куча перестраивается (что может быть сделано за время $O(\log n_{\mathcal{H}})$, где $n_{\mathcal{H}}$ — максимально возможная мощность кучи $\mathcal{H}$), одновременно определяется новое значение для “ближайшей новой” операции. Кроме того, актуальная операция перестает быть текущей операцией как своей работы, так и своей машины, и эти титулы переходят соответственно к следующей (в списке $\mathcal{L}$) операции этой работы и следующей операции этой машины.

На каждом шаге также принимается решение о назначении в расписание в момент τ каких-то (не более двух) новых операций. Это назначение производят две процедуры, которые называются “назначение работы” и “загрузка машины”. Пусть актуальная операция $o^{j'}_{k'}$ выполняется на машине $M_{i'}$.

Назначение работы предназначено для постановки в расписание в момент τ одной из еще не назначенных операций работы $J_{j'}$. Поиск такой операции производится в списке $R^J_{j'}$. (Использовать для этой цели списки A_i не имеет смысла, так как если список A_i не пуст, то согласно сделанному выше замечанию машина M_i не простаивает. Следовательно, никакая операция не может быть назначена на эту машину в текущий момент.) Последовательно просматриваются операции из списка $R^J_{j'}$ и для каждой операции o' выполняются следующие действия.

- Если o' перекрывается с текущей операцией машины $M_i = M(o')$ и эта текущая операция — новая, то o' перемещается в список A_i.
- Если o' перекрывается с текущей операцией машины $M_i = M(o')$ и эта операция — старая, то o' перемещается в список R^M_i.

- Если o' не перекрывается с текущей операцией машины $M(o')$, но перекрывается с текущей операцией работы $J_{j'}$, то o' остается в списке $R^J_{j'}$.
- Наконец, если o' не перекрывается ни с текущей операцией машины $M(o')$, ни с текущей операцией работы $J_{j'}$, то операция o' назначается в расписание в момент τ и одновременно выполняются следующие действия:
  - o' удаляется из списка $R^J_{j'}$;
  - o' назначается текущей операцией работы $J_{j'}$ и машины $M(o')$ и определяются соответствующие ссылки на предыдущие и последующие операции по обеим цепочкам в списке $\mathcal{L}$;
  - o' помещается в список $\mathcal{L}$ и в кучу $\mathcal{H}$ и за время $O(\log n_{\mathcal{H}})$ выполняется необходимая перестройка кучи.

Процедура заканчивается, как только какая-то операция из списка $R^J_{j'}$ назначается в расписание, либо когда все операции из списка $R^J_{j'}$ просмотрены. Ясно, что во время текущего шага будет назначено не более одной операции из списка $R^J_{j'}$.

Загрузка машины предназначена для постановки в расписание в момент τ одной из еще не назначенных операций машины $M_{i'}$. Вначале проверяется, является ли актуальная операция старой, и если "да", то всё содержимое списка $R^M_{i'}$ перемещается в список $A_{i'}$. Затем (независимо от типа актуальной операции: старая она или новая) последовательно просматривается список $A_{i'}$, и для каждой операции $o^j_k \in A_{i'}$ выполняются следующие действия.

- Если o^j_k уже назначена в расписание (такое возможно, в силу относительной независимости списков $\{A_i\}$), то o^j_k удаляется из списка $A_{i'}$.
- Если o^j_k перекрывается с текущей операцией машины $M_{i'}$ (очевидно, текущая операция является "старой"), то o^j_k перемещается из $A_{i'}$ в $R^M_{i'}$.
- Если o^j_k перекрывается с текущей операцией работы J_j, то o^j_k перемещается из $A_{i'}$ в R^J_j.
- Если o^j_k не перекрывается с обеими операциями, то она назначается в расписание в момент τ. При этом, как и в процедуре "назначение работы", выполняются все действия, положенные для только что назначенной операции, и просмотр списка $A_{i'}$ прекращается.

Описание алгоритма $\mathcal{A}_{\mathrm{GC}}$ завершено. ■

Лемма 31.1 *Пусть в r-стадийной m-процессорной системе open shop задано частичное расписание S, в котором для S_k операций цеха $\mathcal{M}_k$ ($k \in \mathbb{N}_r$) расписание полностью определено, а для n_k ("новых") операций цеха $\mathcal{M}_k$ его предстоит доопределить. Пусть также информация о частичном расписании S задана в том виде,*

как это требуется в алгоритме $\mathcal{A}_{\mathrm{GC}}$. Тогда алгоритм $\mathcal{A}_{\mathrm{GC}}$ достраивает исходное частичное расписание до полного допустимого расписания для n работ на m машинах за время $O\left(\sum_{k=1}^{r} n_k(rm_k + S_k + \log n_{\mathcal{H}})\right)$, где $n_{\mathcal{H}}$ — максимально возможное число одновременно выполняемых новых операций.

Доказательство. Работа алгоритма на каждом шаге состоит из нахождения новой актуальной операции и выполнения процедур *Назначение работы* и *Загрузка машины*. Если в качестве актуальной операции выбрана ближайшая старая, то это требует лишь $O(1)$ действий, тогда как при выборе ближайшей новой операции в качестве актуальной мы должны $O(\log n_{\mathcal{H}})$ действий потратить на перестройку кучи $\mathcal{H}$. Следовательно, суммарная трудоемкость выбора актуальной операции по всем шагам не превосходит $O(nr + n_{\mathrm{new}} \log n_{\mathcal{H}})$, где $n_{\mathrm{new}} = \sum_{k=1}^{r} n_k$ — общее число новых операций.

Трудоемкость алгоритма внутри процедур складывается из трудоемкости действий, связанных с перемещениями новых операций между списками A_i, R_j^J и R_i^M, и действий, связанных с назначением новых операций в расписание. Первая пропорциональна количеству перемещений. Для каждой машины $M_i \in \mathcal{M}_k$ каждая из n_k новых операций, первоначально входящих в список A_i, может не более $r-1$ раз быть удалена (в соответствующий список R_j^J) по причине ее перекрытия с другой операцией той же работы, — и не более $r-1$ раз возвращена, — и не более s_i раз удалена (возвращена) по причине ее перекрытия с одной из старых операций на машине M_i, где s_i — количество старых операций на машине M_i. Последнее удаление происходит при назначении операции в расписание. Таким образом, общее число перемещений каждой новой операции из списка A_i есть $O(r + s_i)$, что в сумме по всем новым операциям и по всем машинам дает

$$O\left(\sum_{k=1}^{r} \sum_{M_i \in \mathcal{M}_k} n_k(r + s_i)\right) = O\left(\sum_{k=1}^{r} n_k(rm_k + S_k)\right).$$

Наконец, при назначении новой операции в расписание самой трудоемкой процедурой является ее добавление в кучу $\mathcal{H}$ с последующей перестройкой кучи, что требует $O(n_{\mathrm{new}} \log n_{\mathcal{H}})$ действий. В итоге получаем требуемую оценку трудоемкости. □

32 Метод последовательной достройки нормального расписания

Определение 32.1 Расписание S для m-машинной системы open shop называется *k-нормальным* ($k \le m$), если оно нормально для первых k машин; m-машинная система open shop называется k-нормальной, если для нее существует k-нормальное расписание.

Предположим, что у нас имеется система открытого типа с m машинами, занумерованными по невозрастанию их нагрузок. Предположим также, что для первых k машин $M_1, \ldots, M_k$ (которые будем называть “начальными”) уже задано допустимое расписание S', и мы хотели бы достроить его до допустимого расписания на l машинах

$(0 \le k < l \le m)$ так, чтобы его длина была по возможности минимальной. Если предположить, что расписание S' задано в том виде, как это требуется в схеме $\mathcal{A}_{\mathrm{GC}}$ (т.е. помимо матриц M и S, задающих исполнителя и время начала каждой операции, имеется еще список $\mathcal{L}$ уже назначенных операций, упорядоченных по неубыванию времен их завершения), то искомая достройка расписания может быть реализована по схеме $\mathcal{A}_{\mathrm{GC}}$.

Через $\mathcal{A}_{k,l}$ обозначим алгоритм достройки расписания на k машинах до расписания на l машинах, работающий по схеме $\mathcal{A}_{\mathrm{GC}}$. В доказываемой ниже лемме формулируется основное свойство расписания, полученного алгоритмом $\mathcal{A}_{k,l}$. Согласно этому свойству, расписание каждой из "дополнительных" машин $M_{k+1}, \ldots, M_l$ будет иметь сравнительно небольшое количество простоя, что позволяет (в предположении нормальности исходного расписания S' и при определенных условиях на вектор разностей) гарантировать нормальность полученного расписания на l машинах. Последовательное применение серии алгоритмов $\{\mathcal{A}_{k,l}\}$ с различными параметрами $\{k, l\}$ позволяет строить нормальные расписания для m машин, если *VOD* исходного примера удовлетворяет определенным соотношениям.

Лемма 32.2 *Пусть задан пример системы открытого типа для n работ и m машин и заданы целые k и l такие, что $0 \le k < l \le m$, и пусть для первых k машин уже построено частичное допустимое расписание S'. Тогда алгоритм $\mathcal{A}_{k,l}$ достраивает расписание S' до допустимого расписания S для l машин таким образом, что каждая из дополнительных машин $M_{k+1}, \ldots, M_l$ простаивает в расписании S не более $(l + k - 1)$ единиц времени. Если при этом информация о расписании S' задана в том виде, как это требуется в схеме $\mathcal{A}_{\mathrm{GC}}$, то трудоемкость алгоритма $\mathcal{A}_{k,l}$ не превосходит $O(nl(l-k))$.*

Доказательство. Рассмотрим произвольную дополнительную машину M_i, $k+1 \le i \le l$, и операцию o_i^j, выполняемую на этой машине последней в расписании S. В каждый момент времени, предшествующий операции o_i^j, когда машина M_i простаивала, операция o_i^j была "запрещена". Т.к. на дополнительных машинах нет "старых" операций, то единственная причина запрета на выполнение операции — ее перекрытие с другой операцией той же работы. Нетрудно видеть, что для каждой из k "старых" операций o_ν^j работы J_j (т.е. выполняемых на машинах M_ν, $\nu \in \mathbb{N}_k$) операция o_i^j не может начинаться в интервале $(s_\nu^j - p_i^j, s_\nu^j + p_\nu^j$, длина которого не превосходит 2, а для каждой из $(l-k-1)$ "новых" операций $\{o_\nu^j \mid \nu \ne i\}$, назначенных в расписание раньше операции o_i^j, запретным является интервал $[s_\nu^j, s_\nu^j + p_\nu^j$ длины не больше 1. Суммарная длина объединения всех запретных для o_i^j интервалов не превосходит $2k + l - k - 1 = k + l - 1$.

Для вывода оценки трудоемкости алгоритма достаточно в формулу трудоемкости схемы $\mathcal{A}_{\mathrm{GC}}$, полученную в лемме 31.1, подставить $r = l$, $n_i = 0$ для $i = 1, \ldots, k$ и $n_i = n, m_i = 1, S_i = 0$ для $i = k+1, \ldots, l$. Получим $O\left(\sum_{i=k+1}^{l} n_i(l + \log n_{\mathcal{H}})\right)$. Т.к. $n_{\mathcal{H}}$ — максимальное число одновременно выполняемых новых операций — не превосходит $l - k$, то получаем требуемую оценку. □

Из леммы 32.2 вытекает следующая

Теорема 32.3 *Пусть задана k-нормальная m-машинная система open shop с вектором разности, удовлетворяющим соотношению $\delta(k+1) \geq l+k-1$ для некоторого l $(k < l \leq m)$, и пусть его k-нормальное расписание S_k задано в виде списка операций, упорядоченных по неубыванию времен их завершения. Тогда расписание S_k может быть достроено до l-нормального расписания за время $O(nl(l-k))$.* □

Заметим, что требование задания исходного расписания S_k в виде списка $\mathcal{L}$ во многих случаях не является ограничивающим. Например, такой список получается автоматически, если мы строили S_k также жадным алгоритмом. Благоприятным также является случай, если нам известен порядок выполнения операций на каждой из машин (как в алгоритме Гонзалеза и Сани и многих других алгоритмах).

33 k-нормализующие и k-достраивающие векторы в $\mathbb{R}^m$

Определение 33.1 Вектор $\Delta \in \mathbb{R}^m$ называется *k-нормализующим*, если для любого примера с *VOD* $= \Delta$ существует k-нормальное расписание.

Вектор $\Delta \in \mathbb{R}^m$ называется *k-эффективно нормализующим* (коротко, *k-EN* вектором), если существует эффективный алгоритм, который для любого примера с *VOD* $= \Delta$ строит его k-нормальное расписание.

Определение 33.2 Вектор $\Delta \in \mathbb{R}^m$ называется *k-жадно достраивающим вектором* (коротко, *k-GC* вектором), если существует последовательность целых чисел $\{k_i\}$ такая, что $k = k_0 < k_1 < \ldots < k_\nu = m$ и

$$\Delta(k_{i-1}+1) \geq k_{i-1} + k_i - 1, \ i \in \mathbb{N}_\nu. \tag{33.1}$$

С учетом последнего определения, мы можем обобщить теорему 32.3 следующим образом.

Теорема 33.3 *Пусть задан пример с k-GC VOD и его k-нормальное расписание S_k. Тогда расписание S_k может быть эффективно достроено до нормального расписания для m машин с помощью последовательности алгоритмов $\mathcal{A}_{k,l}$.* □

Легко понять, что кроме описанного выше жадного алгоритма достройки нормального расписания могут существовать и другие (как эффективные, так и неэффективные) процедуры достройки расписания, гарантирующие его нормальность при определенных условиях на *VOD* . Чтобы обобщить наш подход к нахождению нормальных и эффективно нормальных классов задачи open shop, введем следующее определение.

Определение 33.4 Вектор $\Delta \in \mathbb{R}^m$ называется *k-эффективно достраивающим вектором* (коротко, *k-EC* вектором), если существует эффективный алгоритм, который для любого k-нормального примера задачи open shop с *VOD* $= \Delta$ и его k-нормального частичного расписания достраивает это расписание до нормального расписания для m машин.

Вектор $\Delta \in \mathbb{R}^m$ называется *k-достраивающим вектором*, если существует (не обязательно эффективный) алгоритм, который для любого k-нормального примера задачи open shop с $VOD = \Delta$ и его k-нормального частичного расписания достраивает это расписание до нормального расписания для m машин.

По теореме 33.3, каждый *k-GC* вектор является, очевидно, *k-EC* вектором.

В следующей теореме формулируются достаточные условия для вектора из $\mathbb{R}^m$ быть нормализующим (эффективно нормализующим).

Теорема 33.5 *Если вектор $\Delta \in \mathbb{R}^m$ является k-нормализующим (k-эффективно нормализующим) и k-достраивающим (k-эффективно достраивающим), то он является нормализующим (эффективно нормализующим).* □

Таким образом, при проверке нормальности вектора Δ мы должны интересоваться двумя вещами: для каких k он является k-нормализующим (k-эффективно нормализующим) и для каких k он является k-достраивающим (k-эффективно достраивающим).

Лемма 33.6 *Если $\Delta', \Delta'' \in \mathbb{R}^m$, Δ' является k-достраивающим (k-эффективно достраивающим) и $\Delta'' \geq \Delta'$, то Δ'' также является k-достраивающим (k-эффективно достраивающим).*

Доказательство. Пусть у нас имеется k-нормальный пример I'' с $VOD = \Delta''$ и известно его k-нормальное расписание $S''_k,\ k < m$. Нашей целью является:

1. преобразовать пример I'' в "расширенный" пример I' с $VOD = \Delta'$ и длительностью каждой операции, не меньшей чем длительность соответствующей операции в примере I'' и не большей 1;

2. преобразовать расписание S''_k в k-нормальное расписание S'_k для примера I' таким образом, чтобы S'_k было "расширением" расписания S''_k в том смысле, что интервал выполнения любой операции o в расписании S''_k содержится в соответствующем интервале выполнения этой операции в расписании S'_k.

Если нам удалось выполнить такое преобразование, то последующие шаги очевидны: во-первых, достроить k-нормальное расписание S'_k до нормального расписания S' для примера I' (это возможно, поскольку пример I' имеет k-достраивающий VOD), и во-вторых, преобразовать расписание S' в нормальное расписание S'' для исходного примера I'' так, чтобы частичное расписание S'_k преобразовалось в частичное расписание S''_k. Второй шаг легко реализуем: достаточно вернуться к исходным (более коротким) длительностям операций, сохраняя моменты начала всех операций.

Чтобы получить требуемое преобразование со свойствами 1–2, мы сначала увеличиваем длительности операций на "дополнительных" машинах $(M_{k+1}, \ldots, M_m)$ — не более чем до $p_{\max} = 1$ каждую. Таким способом мы сможем увеличить загрузки дополнительных машин, пока их разности с $L_{\max}$ не станут равными $\Delta'(i),\ i = k+1, \ldots, m$. Для первых k ("начальных") машин этот прием, вообще говоря, не проходит, поскольку мы

должны дополнительно сохранять расписание S''_k "внутри" нового расписания S'_k. Чтобы достичь нашей цели, мы добавляем новые работы, каждая из которых имеет только одну операцию (на одной из начальных машин). Каждая новая операция на машине M_i (за исключением, быть может, операции, добавляемой последней) либо добавляет единицу к машиной загрузке, либо заполняет какой-то пустой промежуток в расписании S''_k на машине M_i. Поскольку итоговая загрузка машины M_i должна быть не больше $L_{\max} \leq n$, и в расписании S''_k каждой машины может быть не более $n+1$ "дыр" в интервале $[0, L_{\max}]$, отсюда следует, что для каждой из начальных машин M_i будет добавлено не более $2n$ новых работ. Таким образом, мы добавляем не более $2nk$ новых работ, что сохраняет эффективность k-достраивающей процедуры для примера I' (если таковая имелась) в терминах числа работ (n) исходного примера.

Таким образом, мы доказали, что если Δ' является k-достраивающим вектором, то таковым же является и вектор Δ'', и если Δ' является k-эффективно достраивающим, то таковым же является и вектор Δ''. □

Определение 33.7 Вектор Δ называется *минимальным k-достраивающим (минимальным k-эффективно достраивающим, минимальным k-жадно достраивающим)*, если он является k-достраивающим (k-эффективно достраивающим, k-жадно достраивающим) и не существует k-достраивающего (k-эффективно достраивающего, k-жадно достраивающего) вектора Δ', такого что $\Delta' < \Delta$.

Мы хотели бы описать множество всех k-достраивающих (k-эффективно достраивающих, k-жадно достраивающих) векторов. Для этого, благодаря лемме 33.6, достаточно описать полное семейство минимальных (и следовательно, попарно несравнимых) k-достраивающих (k-эффективно достраивающих, k-жадно достраивающих) векторов. Далее мы сконцентрируемся на k-жадно достраивающих векторах.

В следующей лемме формулируются необходимые условия для вектора быть *k-GC* вектором.

Лемма 33.8 *Каждый k-GC вектор $\Delta \in \mathbb{R}^m$ удовлетворяет следующим соотношениям*

$$\sum_{j=k+1}^{i} \Delta(j) \geq (i-k)(i+k-1),\ i = k+1, \ldots, m. \tag{33.2}$$

Доказательство. По определению *k-GC* векторов, существует последовательность $\{k_i\}$, такая что $k = k_0 < k_1 < \ldots < k_\nu = m$ и

$$\Delta(k_l + 1) \geq k_l + k_{l+1} - 1,\ l = 0, \ldots, \nu - 1. \tag{33.3}$$

Для каждого $i,\ k_l < i \leq k_{l+1}$, из (23.2) и (33.3) получаем, что

$$\sum_{j=k_l+1}^{i} \Delta(j) \geq (i-k_l)\Delta(k_l+1) \geq (i-k_l)(k_l+k_{l+1}-1). \tag{33.4}$$

В частности,

$$\sum_{j=k_q+1}^{k_{q+1}} \Delta(j) \geq (k_{q+1}-k_q)(k_{q+1}+k_q-1) = (k_{q+1}^2-k_{q+1})-(k_q^2-k_q),\ q=0,\ldots,\nu-1. \quad (33.5)$$

Складывая первые l соотношений (33.5), получаем

$$\sum_{j=k+1}^{k_l} \Delta(j) \geq (k_l^2-k_l)-(k^2-k). \quad (33.6)$$

Наконец, складывая (33.6) и (33.4), получаем

$$\begin{aligned}\sum_{j=k+1}^{i} \Delta(j) &\geq k_l^2-k_l-k^2+k+i(k_l+k_{l+1}-1)-k_l^2+k_l-k_lk_{l+1}\\ &= i^2-i-k^2+k+i(k_l+k_{l+1})-k_lk_{l+1}-i^2=(i-k)(i+k-1)\\ &+ (k_{l+1}-i)(i-k_l)\geq(i-k)(i+k-1).\end{aligned}$$

Лемма 33.8 доказана. □

Лемма 33.9 *Вектор $\Delta \in \mathbb{R}^m$ является минимальным k-GC вектором, если и только если*

$$\Delta(j)=0,\ j\in\mathbb{N}_k, \quad (33.7)$$

и существует последовательность целых чисел $\{k_i\}$ такая, что $k=k_0<k_1<\ldots<k_\nu=m$ и

$$\Delta(j)=k_{l-1}+k_l-1,\ j=k_{l-1}+1,\ldots,k_l;\ l=1,\ldots,\nu. \quad (33.8)$$

Доказательство. Пусть Δ является минимальным k-*GC* вектором. Тогда, во-первых, выполняются соотношения (33.7); во-вторых, все неравенства (33.3) выполняются как равенства, и в-третьих, все $\Delta(j)$, не входящие в соотношения (33.3) (т.е. для $k_{l-1}+1<j\leq k_l$) достигают своих минимально возможных значений, равных $\Delta(k_{l-1}+1)$, что приводит к условию (33.8).

Докажем обратное утверждение. Пусть равенства (33.8) выполняются для некоторой последовательности $\{k_i\}$. Тогда очевидно, что вектор Δ является k-жадно достраивающим. Докажем, что он является минимальным, т.е. не существует другого k-*GC* вектора Δ' (соответствующего другой последовательности $\{k_i\}$), такого что $\Delta'<\Delta$.

Из равенств (33.8) следует, что (33.4)–(33.6) также выполняются как равенства (поскольку получаются из сложения равенств (33.8)). В частности, для $l=\nu$ в (33.6) имеем

$$\sum_{j=k+1}^{m} \Delta(j) = m^2-m-k^2+k=(m-k)(m+k-1),$$

откуда следует, что последнее из неравенств (33.2) (содержащее все компоненты $\Delta(j)$, $k+1\leq j\leq m$) выполняется как равенство. Следовательно, никакая из компонент

$\Delta(j),\ k{+}1 \le j \le m$, не может быть уменьшена без нарушения последнего из неравенств (33.2) (которое, как мы знаем из леммы 33.8, является необходимым условием, чтобы вектор был k-*GC* вектором). Отсюда и из (33.7) вытекает минимальность вектора Δ. □

Из леммы 33.9 следует, что каждый минимальный k-*GC* вектор Δ задается единственной последовательностью чисел $\{k_i\}$, удовлетворяющей (33.8). Таким образом, приходим к следующему утверждению.

Утверждение 33.10 *Существует в точности 2^{m-k-1} минимальных k-GC векторов $\Delta \in \mathbb{R}^m$.* □

Если бы условия (33.2) в лемме 33.8 были не только необходимыми, но и достаточными условиями для вектора быть k-жадно достраивающим, это дало бы нам эффективный способ определения, является ли заданный вектор k-*GC* вектором. К сожалению, это не так, и условия (33.2) не являются достаточными. Например, вектор $(0,0,5,6,7)$ является решением системы (33.2) при $k = 2$, но не является 2-*GC* вектором. (Возможно, он все же является нормализующим, но мы пока не можем этого доказать.) Описываемый ниже алгоритм $\mathcal{A}_{\rm com}$ предоставляет эффективную процедуру проверки, является ли заданный вектор k-*GC* вектором.

Определим логическую функцию $\mathrm{COMPL}(\Delta, m, k)$, которая равна **TRUE**, если и только если вектор $\Delta \in \mathbb{R}^m$ является k-*GC* вектором. Алгоритм $\mathcal{A}_{\rm com}$ вычисляет все значения функции COMPL для $k \in \mathbb{N}_m$ по рекуррентной формуле (33.9).

Алгоритм $\mathcal{A}_{\rm com}$

$\mathrm{COMPL}(\Delta, m, m) :=$ **TRUE**;
for $k := m-1$ downto 1 do

$$\mathrm{COMPL}(\Delta, m, k) := \bigvee_{i=k+1}^{m} (\Delta(k+1) \ge k+i-1) \& \mathrm{COMPL}(\Delta, m, i). \quad \blacksquare \qquad (33.9)$$

Для каждого истинного значения функции $\mathrm{COMPL}(\Delta, m, k)$ запоминаем номер i, который обеспечивает значение **TRUE** для правой части соотношения (33.9). Это дает нам возможность для любого вектора Δ, который оказался k-*GC* вектором, восстановить обратным ходом последовательность $\{k_i\}$, удостоверяющую, что Δ является k-*GC* вектором.

Лемма 33.11 *Существует алгоритм $\mathcal{A}_{\rm com}$, который по любому заданному вектору $\Delta \in \mathbb{R}^m$ для каждого $k \in \mathbb{N}_m$ проверяет, является ли Δ k-GC вектором, и если "да", то выводит последовательность чисел $\{k_i\}$, удостоверяющую его принадлежность к k-GC векторам. Алгоритм имеет трудоемкость $O(m^2)$.* □

Таким образом, благодаря лемме 33.11, теореме 33.3 и алгоритмам $\mathcal{A}_{\rm com}$ и $\mathcal{A}_{k,l}$ мы можем предложить следующую эффективную процедуру достройки частичного k-нормального расписания до нормального.

Алгоритм $\mathcal{A}_{\rm GC}$

Искомый алгоритм состоит из четырех шагов.

1. Для заданного примера вычисляем его *VOD* $\Delta \in \mathbb{R}^m$.
2. Используя алгоритм $\mathcal{A}_{\rm com}$, для каждого $i = m-1, m-2, \ldots, 1$ проверяем, является ли вектор Δ i-*GC* вектором.
3. Если Δ оказался i-*GC* вектором для некоторого $i \le k$, восстанавливаем (обратным ходом) последовательность $\{k_i\}$, $i = k_0 < k_1 < \ldots < k_\nu = m$, удостоверяющую этот факт.
4. Применяя последовательность алгоритмов $\mathcal{A}_{k_{i-1},k_i}$, $i = 1, \ldots, \nu$, достраиваем нормальное расписание для i машин до нормального расписания для m машин ■

Главные свойства данного алгоритма формулируются в следующей лемме.

Теорема 33.12 *Существует алгоритм трудоемкости $O(nm^2)$, который для любого примера задачи open shop с n работами и m машинами и его частичного k-нормального расписания определяет, является ли VOD данного примера i-GC вектором для какого-либо $i \le k$, и в случае положительного ответа достраивает частичное i-нормальное расписание до нормального расписания для m машин.* □

Алгоритм $\mathcal{A}_{\rm GC}$ нуждается в небольшом комментарии. Может показаться, что поскольку мы уже имеем нормальное расписание для k машин, то достаточно только проверить, что *VOD* является k-*GC* вектором. В действительности, этого недостаточно. Пусть нам дан пример с *VOD* $\Delta = (0, 4, 4, 4, 8)$ и известно его 3-нормальное расписание для машин M_1, M_2, M_3. Тогда функция COMPL будет иметь значения (вычисленные согласно алгоритму $\mathcal{A}_{\rm com}$) представленные в таблице 33.1.

k	5	4	3	2	1
COMPL$(\Delta, 5, k)$	T	T	F	F	T

Таблица 33.1: Значения функции COMPL$(\Delta, 5, k)$ для $\Delta = (0, 4, 4, 4, 8)$.

Из таблицы 33.1 видно, что Δ не является ни 3-*GC*, ни 2-*GC* вектором, и следовательно, нормальное расписание, уже построенное для машин M_2, M_3, оказывается для нас бесполезным. В то же время, начиная с 1-нормального расписания, мы легко достраиваем его до нормального, используя алгоритмы $\mathcal{A}_{1,4}$ и $\mathcal{A}_{4,5}$.

34 Эффективно нормализующие векторы в $\mathbb{R}^m$

Утверждение 34.1 (см. [54]) *Вектор $\Delta = (0, 3, 3, 3)$ является эффективно нормализующим в $\mathbb{R}^4$. Существует алгоритм, который за время $O(n)$ находит нормальное расписание для любого примера с VOD $\geq \Delta$.*

Следующая теорема дает достаточные условия для получения семейства эффективно нормализующих векторов.

Теорема 34.2 *Для любого целого $k \leq \sqrt{m}$ каждый k-GC вектор $\Delta \in \mathbb{R}^m$ является эффективно нормализующим.*

Доказательство. Пусть задан k-*GC* вектор Δ. Если $k = 1$, то он является эффективно нормализующим вследствие теоремы 33.5 и того, что всякий вектор является 1-эффективно нормализующим. Пусть $k > 1$.

Поскольку Δ является k-*GC* вектором, из определения k-*GC* векторов следует, что для некоторого k' $(k \leq k' < m)$ выполняются соотношения

$$L_{\max} \geq \Delta(k'+1) \geq m + k' - 1 \geq m \geq k^2 \geq k^2 - 1 + \frac{1}{k-1},$$

из которых, ввиду следствия 25.4, вытекает, что Δ является k-*EN* вектором. С учетом предположения, что Δ является k-*GC* вектором, и ввиду теоремы 33.5 заключаем, что Δ является эффективно нормализующим. □

В следующем утверждении сформулировано несколько достаточных условий, чтобы вектор $\Delta \in \mathbb{R}^m$ был k-эффективно нормализующим.

Утверждение 34.3
(1) Каждый вектор $\Delta \in \mathbb{R}^m$ является 1-EN вектором.
(2) Если $\Delta(2) \geq 1$ или $\Delta(m) \geq 2$, то Δ является 2-EN вектором.
(3) Если $\Delta(m) \geq 7$, то Δ является 3-EN вектором.
(4) $\Delta = (0, 3, 3, 3, \Delta')$ является 4-EN вектором.
(5) Если $\Delta(m) \geq \eta_1(k)$ для некоторой верхней оценки $\eta_1(k) \geq \eta_n(k)$ функции $\eta_n(k)$, то Δ является k-нормализующим.
(6) Если $\Delta(m) \geq \eta_2(k)$ для некоторой верхней оценки $\eta_2(k) \geq \eta_{en}(k)$ функции $\eta_{en}(k)$, то Δ является k-эффективно нормализующим.
(7) Если $\Delta(2) \geq k - 1$ и $\Delta(m) \geq 5.45k - 7$, то Δ является k-EN вектором.
(8) Если $\Delta(2) \geq k$, то Δ является k-EN вектором.

Доказательство. (1) Достаточно расписать операции на машине M_1 в интервале $[0, L_1]$ в произвольном порядке.
(2) Следует из алгоритма Гонзалеза и Сани для двух машин [87].
(3) Является следствием теоремы 25.2.
(4) См. утверждение 34.1.

(5) По определению функции $\eta_n(k)$.
(6) По определению функции $\eta_{en}(k)$.
(7) Является следствием теоремы 30.2.
(8) Поскольку вектор Δ является 1-*EN* вектором и 1-жадно достраивающим для k машин (согласно определению 33.2), то по теореме 33.5 он является k-*EN* вектором. □

Наименьшие из известных *EN*-векторов в $\mathbb{R}^4$ перечислены в следующем утверждении. В случаях (2)–(4) эффективная нормальность вектора следует из теоремы 33.5 и факта его принадлежности к k-*EN* и k-*GC* векторам для некоторого k.

Утверждение 34.4
Следующие векторы являются эффективно нормализующими в $\mathbb{R}^4$*:*
(1) $(0,3,3,3)$ *(из утверждения 34.3(4));*
(2) $(0,0,5,5)$ *(2-EN, по утверждению 34.3(2); 2-GC, по определению 33.2);*
(3) $(0,0,2,6)$ *(3-EN, по теореме 24.1; 3-GC, по определению 33.2);*
(4) $(0,0,0,7)$ *(3-EN, по утверждению 34.3(3); 3-GC).* □

В следующей теореме представлено несколько наиболее простых случаев *EN*-векторов в пространстве $\mathbb{R}^m$ для $m \geq 5$.

Теорема 34.5
Следующие векторы $\Delta \in \mathbb{R}^m$ *являются эффективно нормализующими при* $m \geq 5$*:*
(1) $(0, m, \ldots, m)$ *(1-EN; 1-GC);*
(2) $(0, m-2, \ldots, m-2, 5.45m-12.45)$ *($(m-1)$-EN, по утверждению 34.3(7); $(m-1)$-GC);*
(3) $(0,0,m+1,\ldots,m+1)$ *(2-EN; 2-GC);*
(4) $(0,0,m,\ldots,m,2m-2)$ *(2-EN; 2-GC);*
(5) $(0,0,0,m+2,\ldots,m+2)$ *(3-EN, по утверждению 34.3(3); 3-GC);*
(6) $(0,3,3,3,m+3,\ldots,m+3)$ *(4-EN, по утверждению 34.3(4); 4-GC);*
(7) $(0,0,5,5,m+3,\ldots,m+3)$ *(2-EN; 2-GC);*
(8) $(0,0,0,6,m+3,\ldots,m+3)$ *(3-EN; 3-GC);*
(9) $(0,\ldots,0,k+m-1,\ldots,k+m-1)$, $\forall\, k \leq \sqrt{m}$ *или* $\forall\, m \geq k^2-k+\frac{1}{k-1}$, *где* $\Delta(i)=0,\ i \in \mathbb{N}_k$ *(по теореме 34.2);*
(10) $(0,\ldots,0,m^2-2m+\frac{1}{m-2})$, *($(m-1)$-EN, по утверждению 34.3(6) и следствию 25.4; $(m-1)$-GC);*
(11) $(0,\ldots,0,\frac{16}{3}(m-1)\log_2(m-1)+\frac{61}{9}m-14\frac{8}{45})$, *($(m-1)$-EN, по утверждению 34.3(6) и следствию 26.3; $(m-1)$-GC);*
(12) $(0,k,\ldots,k,k+m-1,\ldots,k+m-1)$, $\forall\, k=5,\ldots,m-1$, *где* $\Delta(i)=k,\ i=2,\ldots,k;$ $\Delta(i)=k+m-1,\ i=k+1,\ldots,m;$ *(1-EN; 1-GC);*
(13) $(0,k-1,\ldots,k-1,k+m-2,\ldots,k+m-2,\max\{5.45k-7,2m-2\})$, $\forall\, k \leq m-2$, *где* $\Delta(i)=k-1,\ i=2,\ldots,k;\ \Delta(i)=k+m-2,\ i=k+1,\ldots,m-1;$ *(k-EN, по утверждению 34.3(7); k-GC);*
(14) $(0,0,0,6,8,10,\ldots,2m-2)$ *(3-EN, по утверждению 34.3(3); 3-GC).* □

Возможности этой теоремы иллюстрирует следующее

Следствие 34.6 *Следующие 7 векторов являются наименьшими из известных эффективно нормализующих векторов в пространстве* $\mathbb{R}^5$*:*
$(0,5,5,5,5)$ *(по теореме 34.5(1));*
$(0,3,3,3,8)$ *(по теореме 34.5(6));*
$(0,0,6,6,6)$ *(по теореме 34.5(3));*
$(0,0,5,5,8)$ *(по теореме 34.5(4));*
$(0,0,0,7,7)$ *(по теореме 34.5(5));*
$(0,0,0,6,8)$ *(по теореме 34.5(14));*
$(0,0,0,0,15\frac{1}{3})$ *(по теореме 34.5(10)).*

Пока ни для какого из *EN* векторов в $\mathbb{R}^m$ при $m \geq 4$ не доказана его принадлежность к *MEN*-векторам. С достоверностью справедливо лишь следующее

Утверждение 34.7 *В пространстве* $\mathbb{R}^4$ *существует по крайней мере два различных MN-вектора и два различных MEN-вектора.*

Доказательство. Из утверждения 34.4 следует, что существует *MN*-вектор вида $\Delta_1 = (0,0,0,\Delta_1(4)) \in \mathbb{R}^4$, и из теоремы 27.1 (при $m = 3$) следует, что $\Delta_1(4) \geq 4$. С другой стороны, поскольку вектор $(0,3,3,3)$ — эффективно нормализующий в $\mathbb{R}^4$, то существует *MN*-вектор $\Delta_2 \in \mathbb{R}^4$ с компонентой $\Delta_2(4) \leq 3$. Таким образом, существует по крайней мере два различных *MN*-вектора в $\mathbb{R}^4$.

Теперь что мы можем сказать о *MEN*-векторах в $\mathbb{R}^4$? Легко убедиться, что вектор $(0,0,0,3)$ не является *EN*-вектором. (Нетрудно определить семейство $\mathcal{F}$ примеров 4-машинной задачи open shop с $(3n+1)$ работами и свойствами $L_1 = L_2 = L_3 = 3,\ L_4 = 0$, такое что для любого примера из $\mathcal{F}$ проверка существования его нормального расписания не легче решения задачи PARTITION для n работ.) Следовательно, существует *MEN*-вектор $\Delta_3 \in \mathbb{R}^4$ с компонентой $\Delta_3(4) > 3$. С другой стороны, мы знаем, что существует *MEN*-вектор $\Delta_4 \in \mathbb{R}^4$ с $\Delta_4(4) \leq 3$. □

35 Полиномиальная аппроксимационная схема для многопроцессорной задачи open shop с фиксированным числом машин

В этом разделе мы покажем, что для многопроцессорной задачи open shop с n работами, фиксированным числом стадий (r) и ограниченным константой числом машин на каждой стадии ($m_i,\ i \in \mathbb{N}_r$) существует полиномиальная аппроксимационная схема линейной от n трудоемкости.

Пусть $L_i = \sum_{j=1}^{n} p_i^j$ обозначает суммарную нагрузку i-го цеха; $L_{\max} = \max_i L_i$; $\hat{L}_{\max} = \max_i L_i/m_i$; $d_j = \sum_{i=1}^{r} p_i^j$ в этом разделе обозначает длину работы J_j, а $d_{\max} = \max d_j$. Ясно, что для задачи $O(P)||C_{\max}$ обе величины $\hat{L}_{\max}$ и $d_{\max}$ являются нижними оценками оптимума:

$$C^*_{\max} \geq \max\{\hat{L}_{\max},\ d_{\max}\}. \tag{35.1}$$

В [105] было показано, что любой жадный алгоритм $\mathcal{A}_G$ (в том числе, алгоритмы, достраивающие пустое расписание по схеме $\mathcal{A}_{\mathrm{GC}}$) обеспечивает построение *плотного* расписания S для задачи $O(P)||C_{\max}$ с оценкой

$$C_{\max}(S) \le \hat{L}_{\max} + d_{\max} \tag{35.2}$$

Таким образом, из (35.1) и (35.2) следует, что алгоритм $\mathcal{A}_G$ строит приближенное расписание для задачи $O(P)||C_{\max}$ с оценкой относительной точности, равной 2.

Наша аппроксимационная схема для $Or(Pm)||C_{\max}$ (где m — общее число машин) использует идею "полужадного" алгоритма, строящего "полуплотные" расписания. В отличие от плотного расписания, где простои машины возможны лишь в такие моменты, когда нет готовых выполняться на ней операций, в полуплотном расписании будет допускаться некоторое — относительно небольшое — количество "насильственных" простоев.

Другой решающей идеей является разбиение всего множества работ на три подмножества: "больших", "средних" и "малых" работ. Для двух вещественных чисел $\alpha' > \alpha'' > 0$ определяем три подмножества работ: $L = \{J_j \in \mathcal{J} \mid d_j \ge \alpha' \hat{L}_{\max}\}$, $M = \{J_j \in \mathcal{J} \mid \alpha'' \hat{L}_{\max} \le d_j < \alpha' \hat{L}_{\max}\}$ и $S = \{J_j \in \mathcal{J} \mid d_j < \alpha'' \hat{L}_{\max}\}$, которые будем называть соответственно *большими, средними* и *малыми*. Операции больших, средних и малых работ также будем называть *большими, средними* и *малыми* (независимо от их действительных размеров). Выбранные числа α', α'' должны удовлетворять следующим требованиям:

(a) число $|L|$ больших работ должно быть ограничено константой (для заданного ε);

(b) суммарная длина средних работ должна быть не больше $\varepsilon \hat{L}_{\max}$;

(c) отношение α''/α' должно быть достаточно малым, чтобы обеспечить неравенство

$$\alpha' + \alpha'' + \alpha''|L| \le \varepsilon. \tag{35.3}$$

Схема представляет собой семейство алгоритмов $\mathcal{A}_\varepsilon$, обеспечивающих полиномиальное построение $(1+\varepsilon)$-приближенного расписания при любом фиксированном $\varepsilon > 0$. Опишем в общем виде работу произвольного алгоритма $\mathcal{A}_\varepsilon$.

Алгоритм $\mathcal{A}_\varepsilon$

Искомое расписание строится за 4 шага.

Шаг 1. Если $\varepsilon \ge 1$ или $d_{\max} \le \varepsilon \hat{L}_{\max}$, то искомое расписание S строится с помощью жадного алгоритма $\mathcal{A}_G$. В противном случае приступаем к шагу 2.

Шаг 2. Находим разбиение множества работ на три подмножества: L, M и S ("больших", "средних" и "малых" работ), удовлетворяющее свойствам 1–3.

Шаг 3. Находим оптимальное расписание S^L для работ из множества L.

Шаг 4. С помощью полужадной модификации схемы $\mathcal{A}_{\mathrm{GC}}$ достраиваем расписание S^L для остальных работ (из множеств M и S). ■

Теорема 35.1 *Для любого вещественного $\varepsilon > 0$, целых r и m и любого примера задачи $Or(Pm)||C_{\max}$ с n работами алгоритм $\mathcal{A}_\varepsilon$ строит расписание S с оценкой*

$$C_{\max}(S) \leq (1+\varepsilon)C^*_{\max}. \tag{35.4}$$

Трудоемкость алгоритма равна $O(n)$, причем мультипликативная константа полиномиальна от m и не зависит от ε.

Доказательство. Если $\varepsilon \geq 1$ или $d_{\max} \leq \varepsilon\hat{L}_{\max}$, то благодаря (35.2), требуемое расписание S строится на первом шаге с помощью алгоритма $\mathcal{A}_G$. Пусть $\varepsilon \in (0,1)$ и $d_{\max} > \varepsilon\hat{L}_{\max}$, и пусть на втором шаге алгоритма $\mathcal{A}_\varepsilon$ мы нашли числа α', α'', удовлетворяющие требованиям 1–3. (Чуть позже мы опишем этот шаг более подробно.) Шаг 3 не нуждается в детальном описании. Ясно, что нетрудно предложить какой-либо алгоритм переборного типа для построения оптимального расписания S^L работ из множества L. Трудоемкость этого алгоритма будет зависеть от числа "больших" работ $|L|$ и числа машин m. Поскольку оба числа ограничены константой, то трудоемкость шага 3 даст вклад в оценку трудоемкости всего алгоритма в виде аддитивной константы.

Для достройки расписания для средних и малых работ на шаге 4 мы могли бы применить схему $\mathcal{A}_{\mathrm{GC}}$. Тогда согласно лемме 31.1, трудоемкость шага 4 оценивалась бы величиной $O\left(\sum_{k=1}^{r} n_k(rm_k + S_k + \log n_{\mathcal{H}})\right) \leq O((rnm + rn|L|)$, а значит, мультипликативная константа при n зависела бы от ε. Покажем, как избавиться от этой неприятности, слегка пожертвовав свойством плотности расписания. Откроем страницу 208 и освежим в памяти описание процедуры *Загрузка машины*.

Произведем следующее небольшое изменение в работе процедуры. До начала просмотра списка $A_{i'}$ мы оценим длину оставшегося интервала простоя на текущей машине, т.е. — от момента τ до начала выполнения текущей (старой!) операции на данной машине, и если эта длина меньше величины $\alpha''\hat{L}_{\max}$, то заканчиваем работу процедуры и переходим к следующему шагу (невзирая на то, что какие-то разрешенные операции из списка $A_{i'}$ вполне умещаются в данном интервале простоя и могли бы быть назначены в момент τ). Это гарантирует нам, что никакая операция малой работы никогда не будет перемещена из списка $A_{i'}$ в список $R^M_{i'}$, а значит, для каждой "малой" операции из цеха i суммарное количество ее перемещений не превосходит $O(rm_i)$ (напомним, что каждая операция представлена в m_i независимых списках A_i), что в сумме по всем операциям малой работы дает $O(rm)$, а по всем малым работам — $O(nrm)$. Для средних операций все остается по-старому, поэтому суммарное количество их перемещений оценивается величиной $O(r|M|(m+|L|))$. Поскольку, однако, и количество больших, и количество средних работ ограничены константами (зависящими от ε), то эта часть трудоемкости дает вклад лишь в аддитивную константу. Таким образом, трудоемкость шага 4 в новой версии оценивается величиной $O(nrm)$.

Покажем, что относительная погрешность описанного алгоритма действительно не превосходит ε. Пусть S^A обозначает полученное расписание, $C^A_{\max}$ — его длину, а $C^L_{\max}$ — длину расписания S^L. Очевидно, что $C^*_{\max} \geq C^L_{\max}$. Без ограничения общности можем предполагать, что *критической машиной*, которая заканчивает работу в момент $C^A_{\max}$, является машина $M_1 \in \mathcal{M}_1$.

Если последней операцией на машине M_1 является "большая" операция, то $C^A_{\max} = C^L_{\max} \le C^*_{\max}$, и расписание S^A оптимально. Далее предполагаем, что последняя операция на машине M_1 в расписании S^A принадлежит работе J_k, которая не является большой. Отсюда следует, что $d_k \le \alpha' \hat{L}_{\max}$. Пусть t — время начала операции o^k_1, а t^*_i обозначает время завершения последней большой операции в расписании S^L на машине $M_i \in \mathcal{M}_1$; $t^* = \max_{M_i \in \mathcal{M}_1} t^*_i$. Ясно, что

$$t^* \le C^*_{\max} \le C^A_{\max} = t + p^k_1,$$

как и то, что никакая большая операция не выполняется в интервале $[t^*, C^A_{\max}]$ на какой-либо машине $M_i \in \mathcal{M}_1$. Сначала предположим, что никакая малая операция (даже частично) не выполняется на машине M_1 в интервале $[t^*, C^A_{\max}]$. (А следовательно, J_k — средняя работа.) Поскольку все простои машины M_1 в интервале $[t^*, C^A_{\max}]$ обусловлены выполнением операций работы J_k в других цехах, то суммарный простой на M_1 в интервале $[t^*, C^A_{\max}]$ не превосходит $d_k - p^k_1$, а суммарная нагрузка M_1 в этом же интервале не превосходит $p^k_1 + \sum_{j \in M \setminus \{k\}} d_j$. Таким образом, длина интервала $[t^*, C^A_{\max}]$ не превосходит суммарной длины средних работ, которая по сделанному выше предположению относительно выбора α' и α'' не превосходит $\varepsilon \hat{L}_{\max}$. Отсюда

$$C^A_{\max} \le t^* + \varepsilon \hat{L}_{\max} \le (1+\varepsilon) C^*_{\max}.$$

Теперь рассмотрим случай, когда существует малая операция o^j_1, которая завершается на машине M_1 в момент $t' > t^*$. Простой на машине $M_i \in \mathcal{M}_1$ в интервалах $[0, t^*_i]$ и $[t^*_i, t]$ будем называть *внутренним* и *внешним*, соответственно. Оценим суммарное внутреннее и внешнее время простоя (I_1 и I_2, соответственно) по всем машинам $M_i \in \mathcal{M}_1$. Прежде всего заметим, что каждый интервал внутреннего простоя либо покрывается интервалом выполнения другой операции работы J_j, либо является тем самым "насильственным" интервалом простоя, который предшествует началу выполнения какой-то большой операции на данной машине и не превосходит "контрольной" величины $\alpha'' \hat{L}_{\max}$. Суммарная длина первых по всем машинам $M_i \in \mathcal{M}_1$ не превосходит $m_1 d_j \le m_1 \alpha'' \hat{L}_{\max}$, а суммарная длина вторых — $|L| \alpha'' \hat{L}_{\max}$. Таким образом, $I_1 < (m_1 + |L|) \alpha'' \hat{L}_{\max}$.

Внешние простои обусловлены только выполнением операций работы J_k в других цехах, поэтому суммарный внешний простой по всем машинам $M_i \in \mathcal{M}_1$ не превосходит $I_2 \le m_1 (d_k - p^k_1)$. Принимая во внимание, что каждая машина $M_i \in \mathcal{M}_1$ в любой момент времени в интервале $[0, t]$ либо занята, либо простаивает, и учитывая, что все машины цеха $\mathcal{M}_1$ могут быть заняты не более L_1 единиц времени в интервале $[0, t]$, приходим к оценке

$$t m_1 \le L_1 + I_1 + I_2 < L_1 + (m_1 + |L|) \alpha'' \hat{L}_{\max} + m_1 (d_k - p^k_1).$$

Отсюда следует, что

$$C^A_{\max} = t + p^k_1 < L_1/m_1 + d_k + (1 + |L|/m_1) \alpha'' \hat{L}_{\max} \le (1 + \alpha' + \alpha'' + \alpha'' |L|) \hat{L}_{\max} \le (1+\varepsilon) C^*_{\max}.$$

Таким образом, алгоритм $\mathcal{A}_\varepsilon$ всегда гарантирует ε-оптимальность получаемых решений.

Опишем алгоритм нахождения чисел α', α'', которые гарантируют разбиение всего множества работ на большие, средние и малые с выполнением свойств 1.–3.

Вычислим значения величин $\{d_j \mid j \in \mathbb{N}_n\}$, $d_{\max} = \max d_j$, $\hat{L}_{\max}$. Напомню, что мы предполагаем выполненным $d_{\max} > \varepsilon \hat{L}_{\max}$ и $\varepsilon < 1$. Можем считать, что $\varepsilon = m/K$ для некоторого целого $K > m$. Сначала будет представлен алгоритм трудоемкости $O(n \log n)$, а затем мы покажем, как получить тот же результат за линейное от n время.

- Все работы нумеруются по невозрастанию d_j.
- Полагается $\alpha_1 = \frac{\varepsilon}{2}$; $\alpha_0 = \infty$. $E_1 = [\alpha_1 \hat{L}_{\max}, \alpha_0 \hat{L}_{\max})$;
  Просматривается список работ, пока не выполнится условие $d_j < \alpha_1 \hat{L}_{\max}$; вычисляется $D_1 \doteq \sum_{d_j \in E_1} d_j$ и полагается $n'_1 = \frac{D_1}{\alpha_1 \hat{L}_{\max}} - 1$.
- Для $k = 2, 3, \ldots$ выполняются следующие действия.
  - Находится α_k из равенства
  $$\alpha_{k-1} + \alpha_k (1 + n'_{k-1}) = \varepsilon. \tag{35.5}$$
  - Определяется $E_k = [\alpha_k \hat{L}_{\max}, \alpha_{k-1} \hat{L}_{\max})$. Продолжается просмотр списка работ, пока не выполнится неравенство $d_j < \alpha_k \hat{L}_{\max}$. Находится $D_k \doteq \sum_{d_j \in E_k} d_j$ и полагается
  $$n'_k = n'_{k-1} + \frac{D_k}{\alpha_k \hat{L}_{\max}}. \tag{35.6}$$
  - Если $D_k \le \varepsilon \hat{L}_{\max}$, то $\{\alpha'' := \alpha_k;\ \alpha' := \alpha_{k-1};\ \textbf{stop}\}$.
- Конец цикла по k.

Сначала убедимся, что для любого $i \ge 2$ выполнено неравенство $\alpha_i < \alpha_{i-1}$ (т.е. величины $\{\alpha_i\}$ определяют непустые и **непересекающиеся** интервалы $\{E_i\}$). Предположим противное, т.е. для некоторого $i \ge 2$ оказалось $\alpha_i \ge \alpha_{i-1}$. Тогда при $i = 2$ получаем $\varepsilon = \alpha_1 + \alpha_2(1 + n'_1) \ge \alpha_1 + \frac{\alpha_1 D_1}{\alpha_1 \hat{L}_{\max}} \ge \alpha_1 + \frac{d_{\max}}{\hat{L}_{\max}} \ge \frac{3}{2}\varepsilon$, — противоречие. При $i \ge 3$ получаем $\varepsilon = \alpha_{i-1} + \alpha_i(1 + n'_{i-1}) \ge \alpha_{i-1}\left(2 + n'_{i-2} + \frac{D_{i-1}}{\alpha_{i-1}\hat{L}_{\max}}\right) > \alpha_{i-1}(2 + n'_{i-2}) + \varepsilon > \varepsilon$, — противоречие.

Пусть $n_i \doteq |\{J_j \mid d_j \ge \alpha_i \hat{L}_{\max}\}|$. Индукцией по i докажем неравенство $n_i \le n'_i$.

Так как в сумму D_1 в качестве слагаемого входит длина работы J_{j^*} ($d_{j^*} = d_{\max} > \varepsilon \hat{L}_{\max}$), а длины остальных $(n_1 - 1)$ работ, представленных в сумме D_1, оцениваются снизу величинами $d_j \ge \frac{\varepsilon}{2}\hat{L}_{\max}$, то

$$D_1 > \varepsilon \hat{L}_{\max} + (n_1 - 1)\frac{\varepsilon}{2}\hat{L}_{\max} = (n_1 + 1)\frac{\varepsilon}{2}\hat{L}_{\max} = (n_1 + 1)\alpha_1 \hat{L}_{\max},$$

откуда $n_1 \le n'_1$.

Пусть неравенство $n_{i-1} \le n'_{i-1}$ доказано. Тогда

$$n_i \le n_{i-1} + \frac{D_i}{\alpha_i \hat{L}_{\max}} \le n'_{i-1} + \frac{D_i}{\alpha_i \hat{L}_{\max}} = n'_i,$$

что и требуется.

Далее убедимся, что оператор **stop** сработал и величины α', α'' определены. Из $D_1 \ge d_{\max} \ge \varepsilon \hat{L}_{\max}$ и соотношений

$$\sum_{k=1}^{K} D_k \le \sum_{j=1}^{n} d_j = \sum_{i=1}^{r} L_i \le \hat{L}_{\max} \sum_i m_i = m\hat{L}_{\max} = K\varepsilon \hat{L}_{\max}$$

следует, что для одной из величин $D_2, \ldots, D_k$ выполнено неравенство $D_k \le \varepsilon \hat{L}_{\max}$, которое является условием срабатывания оператора **stop**.

Наконец убедимся, что величины α', α'', определенные согласно описанной выше процедуре, удовлетворяют требованиям (a)–(c). Требование (b) является условием срабатывания **stop** и, как уже отмечалось, выполнено. Требование (c) вытекает из соотношений

$$\alpha_{k-1} + \alpha_k(1 + n_{k-1}) \le \alpha_{k-1} + \alpha_k(1 + n'_{k-1}) = \varepsilon.$$

Требование (a) будет непосредственно вытекать из верхней оценки на n_{k-1}, которую мы сейчас получим.

Для удобства выкладок примем $\varepsilon \hat{L}_{\max}$ за единицу и обозначим $\beta_i = \alpha_i/\varepsilon$. Тогда для $i = 3, \ldots, k$ из (35.5) и (35.6) имеем

$$1 - \beta_{i-1} = \beta_i(1 + n'_{i-1}) = \beta_i(1 + n'_{i-2} + D_{i-1}/\beta_{i-1}) = \beta_i(1 - \beta_{i-2} + D_{i-1})/\beta_{i-1},$$

или $\beta_i(1 - \beta_{i-2} + D_{i-1}) = \beta_{i-1}(1 - \beta_{i-1})$.

При $i = 2$ получаем аналогичное соотношение
$1 - \beta_1 = \beta_2(1 + n'_1) = \beta_2 D_1/\beta_1$, или $\beta_2(1 - \beta_0 + D_1) = \beta_1(1 - \beta_1)$, где $\beta_0 = 1$. Таким образом, величины β_i находятся из рекуррентных соотношений

$$\beta_0 = 1;\ \beta_1 = 1/2; \tag{35.7}$$

$$\beta_i(1 - \beta_{i-2} + D_{i-1}) = \beta_{i-1}(1 - \beta_{i-1}),\ i = 2, \ldots, k, \tag{35.8}$$

как функции от параметров $\{D_i\}$, удовлетворяющих соотношениям

$$D_i \ge 1;\ \sum_{i=1}^{k-1} D_i \le N.$$

Поскольку число n_{k-1} больших работ удовлетворяет неравенству

$$n_{k-1} \le n'_{k-1} = \frac{1 - \beta_{k-1}}{\beta_k} - 1 \le \frac{1}{\beta_k} - 2,$$

то при получении верхней оценки для n_{k-1} достаточно оценить снизу величину β_k при $k \in [2, N]$. Так как величина D_{k-1} присутствует лишь в последнем рекуррентном соотношении из (35.8), т.е.

$$\beta_k(1 - \beta_{k-2} + D_{k-1}) = \beta_{k-1}(1 - \beta_{k-1}),$$

то β_k принимает минимальное значение при максимально возможном D_{k-1}. Таким образом, мы можем считать выполненным равенство $\sum_{i=1}^{k-1} D_i = N$. Для каждого $k \in \{2, \ldots, N\}$ найдем нижнюю оценку для величины β_k при соотношениях (35.7), (35.8) и

$$D_i \geq 1;\ \sum_{i=1}^{k-1} D_i = N. \tag{35.9}$$

Вначале покажем, что при любом $i = 0, 1, 2, \ldots$ величина β_i удовлетворяет неравенству

$$\beta_i \leq 1/2^i. \tag{35.10}$$

Для β_0 и β_1 неравенство выполнено. Пусть оно выполнено для $i \leq k$, где $k \geq 1$. Из (35.8) и $D_i \geq 1$ получаем

$$\beta_{k+1} = \frac{\beta_k(1 - \beta_k)}{1 - \beta_{k-1} + D_k} \leq \frac{\beta_k(1 - \beta_k)}{2 - 1/2^{k-1}}.$$

Выражение $\beta_k(1 - \beta_k)$ как функция от β_k возрастает на отрезке $[0, 1/2]$ и с учетом неравенства $\beta_k \leq 1/2^k$ принимает максимальное значение при $\beta_k = 1/2^k$. Поэтому

$$\beta_{k+1} \leq \frac{\frac{1}{2^k}\left(1 - \frac{1}{2^k}\right)}{2(1 - 1/2^k)} = \frac{1}{2^{k+1}},$$

т.е. (35.10) выполнено при $i = k + 1$.

Последовательно воспользовавшись рекуррентной формулой (35.8), получаем

$$\begin{aligned}
\beta_k &= \frac{(1 - \beta_{k-1})(1 - \beta_{k-2}) \cdots (1 - \beta_1)\beta_1}{(1 - \beta_{k-2} + D_{k-1})(1 - \beta_{k-3} + D_{k-2}) \cdots (1 - \beta_1 + D_2)D_1} \\
&= (\text{ибо } \beta_1 = \tfrac{1}{2}) = \frac{1 - \beta_{k-1}}{2D_1} \prod_{i=1}^{k-2} \frac{1 - \beta_i}{1 - \beta_i + D_{i+1}} \geq (\text{см. (35.10)}) \\
&\geq \frac{1 - 2^{-k+1}}{2D_1} \prod_{i=1}^{k-2} \frac{1 - 2^{-i}}{1 - 2^{-i} + D_{i+1}} = \frac{1}{2} \prod_{i=1}^{k-1} \frac{1 - 2^{-i}}{1 - 2^{-i+1} + D_i}.
\end{aligned} \tag{35.11}$$

Сначала оценим снизу величину $\prod_{i=1}^{k-1}(1 - 2^{-i})$. Имеем

$$\ln \prod_{i=1}^{k-1}(1 - 2^{-i}) > \ln \prod_{i \geq 1}(1 - 2^{-i}) = \sum_{i \geq 1} \ln(1 - 2^{-i})$$

$$= -\sum_{i\geq 1}\sum_{j\geq 1}\frac{1}{j(2^i)^j} = -\sum_{j\geq 1}\sum_{i\geq 1}\frac{1}{j(2^j)^i} = -\sum_{j\geq 1}\frac{1}{j(2^j-1)}.$$

Так как $2j(2^j-1) < (j+1)(2^{j+1}-1)$ при любом $j \geq 5$, то

$$-\sum_{j\geq 5}\frac{1}{j(2^j-1)} > -\frac{2}{5(2^5-1)}.$$

Поэтому

$$\ln\prod_{i=1}^{k-1}(1-2^{-i}) > -\sum_{j=1}^{4}\frac{1}{j(2^j-1)} - \frac{2}{5(2^5-1)} > -1,25$$

Следовательно,

$$\prod_{i=1}^{k-1}(1-2^{-i}) > e^{-1,25}. \tag{35.12}$$

Теперь оценим сверху величину

$$\prod_{i=1}^{k-1}(1-2^{-i+1}+D_i). \tag{35.13}$$

Нетрудно убедиться, что при фиксированной сумме величин D_i, равной N, максимум выражения (35.13) достигается в случае, когда все сомножители равны, т.е. при $D_1 = 1-2^{-1}+D_2 = 1-2^{-2}+D_3 = \cdots = 1-2^{-k+2}+D_{k-1}$, или $D_{i+1} = D_1 - 1 + 2^{-i},\ i \in \mathbb{N}_{k-2}$. (Здесь мы игнорируем ограничение $D_i \geq 1$.) При таких $D_1, \ldots, D_{k-1}$ имеем

$$N = \sum_{i=1}^{k-1} D_i = D_1(k-1) - (k-2) + \sum_{i=1}^{k-2} 2^{-i}.$$

Следовательно,

$$D_1 = \left(N + k - 2 - \sum_{i=1}^{k-2} 2^{-i}\right)/(k-1) < \frac{N+k-2}{k-1},$$

а выражение (35.13) не превосходит $D_1^{k-1} \leq \left(\frac{N+k-2}{k-1}\right)^{k-1}$.

Нетрудно убедиться, что функция $\left(\frac{N-1+x}{x}\right)^x$ от x возрастает при $x > 0$, поскольку производная ее логарифма, равная $\ln\frac{N-1+x}{x} - \frac{N-1}{N-1+x}$, положительна. (Это следует из неравенства $\ln\frac{1}{1-\Delta} > \Delta$ при $\Delta \in (0,1)$.) Отсюда следует, что величина $\left(\frac{N+k-2}{k-1}\right)^{k-1}$ достигает максимума при $k = N$, т.е. выражение (35.13) не превосходит 2^{N-1}. Подставляя эту оценку, а также оценку (35.12) в (35.11), получим $\beta_k > e^{-1,25}\cdot 2^{-N}$. Таким образом, число больших работ не превосходит $n_{k-1} < 1/\beta_k < e^{1,25}\cdot 2^{m/\varepsilon}$.

Описанный выше алгоритм трудоемкости $O(n\log n)$ легко преобразуется в линейный. Поскольку для любого заданного примера гарантируется оценка $\alpha'' > \varepsilon/(e^{1,25}\cdot 2^N)$,

то мы можем сначала за линейное время отобрать заведомо "малые" работы J_j, для которых выполняется неравенство $d_j \leq \frac{\varepsilon}{e^{1,25} \cdot 2^N} \hat{L}_{\max}$, а затем уже для оставшихся работ (число которых ограничено сверху константой $\frac{m}{\varepsilon} \cdot e^{1,25} \cdot 2^{m/\varepsilon}$) с помощью описанного выше алгоритма отыскивать числа α', α'' и распределение оставшихся работ по множествам L, M и S.

Таким образом, трудоемкость шага 2 есть $O(rn)$, а трудоемкость всего алгоритма $\mathcal{A}_\varepsilon$ оценивается величиной $O(nrm)$, в которой от ε зависит лишь аддитивная константа. Теорема 35.1 доказана.

Об эффективности полиномиальной схемы

Рассмотрим пример с параметрами $m = 3, \varepsilon = 1/3$. Алгоритм, описанный в [124], гарантирует нахождение 4/3-приближенного решения задачи $O3||C_{\max}$ с линейной от n трудоемкостью, где в качестве аддитивной константы в оценке трудоемкости присутствует трудоемкость нахождения приближенного решения для примера с 5 работами с абсолютной оценкой точности

$$C_{\max}(S) \leq \frac{4}{3} L_{\max}.$$

(Доказывается, что такое расписание существует для любого примера задачи $O3||C_{\max}$.) В описанной выше аппроксимационной схеме к линейной от n трудоемкости добавляется аддитивная константа, в которой содержится трудоемкость построения расписания (причем, оптимального) для существенно большего числа работ (при $m = 3, \varepsilon = 1/3$ число "больших" работ может быть порядка тысячи). Таким образом, с практической точки зрения построенная нами схема оказывается неработоспособной.

Выход из этой ситуации состоит в дальнейшем совершенствовании схемы. В основу более совершенной схемы могли бы быть положены более глубокие знания о свойствах оптимальных расписаний задачи open shop. Так например, мы знаем, что в любом плотном расписании количество внутренних интервалов простоя на любой машине не превосходит $r - 1$. И хотя для некоторых примеров может не существовать плотных оптимальных расписаний (один из таких примеров — для трех работ на трех машинах — приведен в [123]), это не исключает возможности, что для одного из оптимальных расписаний любого примера выполняется аналогичное свойство, т.е. количество внутренних интервалов простоя ограничено величиной, не зависящей от числа работ. И если бы такое свойство удалось показать, мы могли бы предложить более эффективную аппроксимационную схему для рассматриваемой задачи, поскольку количество внутренних простоев в получаемом расписании не пришлось бы оценивать через число больших работ.

36 Барьер точности для задачи open shop с нефиксированным числом машин

Несуществование аппроксимационной схемы для задачи open shop (при условии P≠NP) будет вытекать из несуществования для любой константы $C < 5/4$ полиномиального C-

приближающего алгоритма. Таким образом, для задачи open shop с нефиксированным числом машин существует *барьер приближаемости*, который лежит где-то в интервале от 5/4 до 2. В свою очередь, для доказательства несуществования полиномиального C-приближающего алгоритма при $C < 5/4$ достаточно доказать NP-полноту следующей проблемы распознавания.

Существование расписания длины 4 для задачи $O||C_{\max}$
Пример: Пример системы open shop с n работами, m машинами и целочисленными длительностями операций.
Вопрос: Существует ли для данного примера допустимое расписание S длины $C_{\max}(S) \leq 4$?

Для доказательства NP-полноты этой задачи к ней будет сведена известная NP-полная проблема

Not-All-Equal-3Sat [17, стр. 332]
Пример: Множество булевых переменных U и набор C дизъюнкций над U, такой что каждая дизъюнкция $c \in C$ состоит не более чем из трех литералов (каждый литерал является либо переменной из U, либо ее отрицанием).
Вопрос: Существует ли назначение переменных, доставляющее значение **истина** всем дизъюнкциям $c \in C$, и такое, что в каждой дизъюнкции найдется по крайней мере один истинный и один ложный литерал?

Лемма 36.1 *Проблема распознавания* Существование расписания длины 4 для задачи $O||C_{\max}$ *является NP-полной.*

Доказательство. Пусть на множестве переменных $U = \{x_1, \ldots, x_u\}$ задано семейство дизъюнкций $C = \{c_1, \ldots, c_\nu\}$, где $c_i = l_{i1} \vee \ldots \vee l_{ik_i}$; литерал l_{ij} совпадает либо с некоторой переменной $x_p \in U$, либо с ее отрицанием $\bar{x}_p$; $k_i \leq 3$, $\forall i$. Без ограничения общности можем считать, что $k_i = 3$, $\forall i$. (В противном случае добавляем в дизъюнкцию недостающие литералы, совпадающие с одним из уже имеющихся в данной дизъюнкции литералов.)

По заданному семейству C построим пример задачи open shop с $(7\nu - u)$ работами и 6ν машинами следующим образом.

Пусть $\mathcal{L} = \{l_{ij} \mid i \in \mathbb{N}_\nu;\ j = 1, 2, 3\}$ — семейство литералов, входящих в дизъюнкции $c_i \in C$; функции $c: \mathcal{L} \to C$ и $x: \mathcal{L} \to U$ задают принадлежность каждого литерала $l \in \mathcal{L}$ к какой-то дизъюнкции $c(l) \in C$ и переменной $x(l) \in U$, а функция $d: \mathcal{L} \to \{+, -\}$ указывает, является ли литерал l вхождением переменной $x(l)$ без отрицания (если $d(l) = \text{“}+\text{”}$) или с отрицанием (если $d(l) = \text{“}-\text{”}$).

Будем определять работы трех типов: α, β и γ. Для каждого литерала $l \in \mathcal{L}$ определим две машины: $A(l)$ и $B(l)$, а также α-работу $\alpha(l)$, состоящую из двух операций: $\alpha_A(l)$ и $\alpha_B(l)$; операции выполняются, соответственно, на машинах $A(l)$ и $B(l)$ за 2 единицы времени. Очевидно, что внутри интервала $[0, 4]$ каждая α-работа $\alpha(l)$ может выполняться одним из двух способов: либо $\alpha_A(l)$ выполняется в интервале $[0, 2]$, а $\alpha_B(l)$ — в

интервале $[2,4]$, либо — наоборот. Сопоставим эти два способа со значениями литерала l:

$$l := \text{“}\alpha_A(l) \text{ предшествует операции } \alpha_B(l)\text{”}.$$

Пусть $p_j = |x^{-1}(x_j) \cap d^{-1}(+)|$ — число вхождений переменной x_j без отрицания, а $n_j = |x^{-1}(x_j) \cap d^{-1}(-)|$ — число ее вхождений с отрицанием. Занумеровав элементы каждого из двух пересечений произвольным образом, первые будем обозначать через $x_{j1}, \ldots, x_{jp_j}$, а вторые — через $\bar{x}_{j1}, \ldots, \bar{x}_{jn_j}$. Для переменной x_j определим (p_j+n_j-1) так называемых β-работ: $\beta(x_{j1}), \ldots, \beta(x_{j,p_j-1}), \beta(\bar{x}_{j1}), \ldots, \beta(\bar{x}_{j,n_j-1}), \hat{\beta}(x_j)$, — которые призваны при любом допустимом расписании длины 4 согласовать значения литералов, относящихся к различным вхождениям одной и той же переменной x_j. А именно, все литералы $x_{j1}, \ldots, x_{jp_j}$ должны получить одно и то же значение, отличное от значений всех литералов $\bar{x}_{j1}, \ldots, \bar{x}_{jn_j}$. С этой целью даем каждой β-работе по две операции: для литерала x_{ji} $(i = 1, \ldots, p_j - 1)$ работа $\beta(x_{ji})$ получает операцию $\beta_B(x_{ji})$ длины 2 на машине $B(x_{ji})$ и операцию $\beta_A(x_{ji})$ длины 1 на машине $A(x_{j,i+1})$; аналогично, для литерала $\bar{x}_{ji}$ $(i = 1, \ldots, n_j - 1)$ работа $\beta(\bar{x}_{ji})$ получает операцию $\beta_B(\bar{x}_{ji})$ длины 2 на машине $B(\bar{x}_{ji})$ и операцию $\beta_A(\bar{x}_{ji})$ длины 1 на машине $A(\bar{x}_{j,i+1})$. Наконец, работа $\hat{\beta}(x_j)$ определяется в примере лишь для переменных x_j, имеющих в C как вхождения с отрицанием, так и без отрицания; работа $\hat{\beta}(x_j)$ состоит из операций $\hat{\beta}_p(x_j)$ и $\hat{\beta}_n(x_j)$, выполняемых на машинах $B(x_{jp_j})$ и $B(\bar{x}_{jn_j})$, соответственно, и требующих по 2 единицы времени каждая. Нетрудно убедиться, что если $x_{ji} =$**истина**, то $\alpha_B(x_{ji})$ выполняется в интервале $[2,4]$, $\beta_B(x_{ji})$ — в интервале $[0,2]$, $\beta_A(x_{ji})$ — на машине $A(x_{j,i+1})$ в одной из единиц интервала $[2,4]$, $\alpha_A(x_{j,i+1})$ — в интервале $[0,2]$ на той же машине, и значит, $x_{j,i+1} =$**истина**. Аналогично обеспечивается согласованность значений остальных вхождений переменной x_j (см. рис. 36.1).

Наконец, для каждой дизъюнкции $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$ определим γ-работу $\gamma(c_i)$, состоящую из трех операций единичной длины: γ_{i1}, γ_{i2} и γ_{i3}, — выполняемых, соответственно, на машинах $A(l_{i1}), A(l_{i2})$ и $A(l_{i3})$. Наличие работы $\gamma(c_i)$ обеспечивает истинность дизъюнкции c_i (при условии существования расписания длины 4) и свойство *Not-All-Equal*, т.е. то, что хотя бы один из трех входящих в дизъюнкцию литералов получит значение **истина**, и хотя бы один — значение **ложь**. Действительно, все три литерала, составляющие c_i, не могут быть истинны одновременно, т.к. в этом случае все три операции работы $\gamma(c_i)$ должны выполняться в интервале $[2,4]$ длины 2. Аналогично, все три литерала не могут быть ложны.

Таким образом, мы доказали, что если для примера I_C, построенного по семейству C, существует расписание длины 4, то семейство C имеет *Not-All-Equal* назначение переменных, при котором все его дизъюнкции принимают значение **истина** (это назначение однозначно определяется из расписания α_A-операций). Нетрудно убедиться, что верна и обратная импликация, т.е. если существует истинное *Not-All-Equal* назначение переменных семейства дизъюнкций C, то для примера I_C существует расписание длины 4. Действительно, по значению каждой переменной x_j однозначно определяется расписание α-операций, а также операций $\beta_B(x_{ji}), \beta_B(\bar{x}_{ji})$ и операций работы $\hat{\beta}(x_j)$ (если таковая имеется). Далее устраиваем расписание для γ-операций. Поскольку из условия

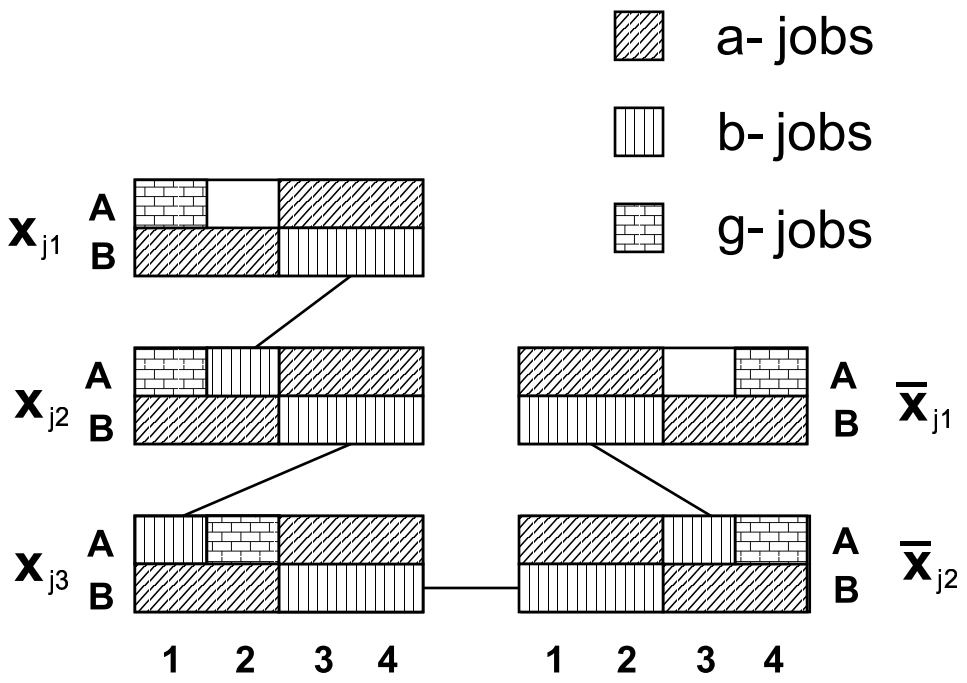


Рис. 36.1: Расписание для операций и машин, соответствующих переменной x_j.

Not-All-Equal следует, что для каждой γ-работы в каждый из интервалов $[0,2],[2,4]$ попадает не более двух ее операций (единичной длины), то нетрудно устроить допустимое расписание для операций каждой γ-работы. Наконец, допустимое расположение оставшихся $\beta_A(x_{ji})$- и $\beta_A(\bar{x}_{ji})$-операций в интервале $[0,4]$ определяется однозначно.

Поскольку сведение проблемы NOT-ALL-EQUAL-3SAT к проблеме распознавания СУЩЕСТВОВАНИЕ РАСПИСАНИЯ ДЛИНЫ 4 ДЛЯ ЗАДАЧИ $O||C_{\max}$ может быть выполнено за время, полиномиальное от длины записи первой проблемы (имеется в виду время, затрачиваемое на построение примера I_C по семейству C и на нахождение значения переменных семейства C по готовому расписанию для примера I_C), и поскольку длина записи примера I_C не превосходит полинома от длины записи семейства C, то из NP-полноты первой проблемы следует NP-полнота второй.

Лемма 36.1 доказана.

Теорема 36.2 *Для любой фиксированной константы $C < 5/4$ не существует полиномиального алгоритма $\mathcal{A}$ решения задачи $O||C_{\max}$ с гарантированной относительной оценкой точности $C_{\max}(S)/C^*_{\max} \leq C$ (при условии $P \neq NP$).*

Доказательство. Пусть для некоторой константы $C < 5/4$ существует алгоритм $\mathcal{A}$ решения задачи $O||C_{\max}$ с гарантированной оценкой $C_{\max}(S_{\mathcal{A}})/C^*_{\max} \leq C$. Покажем,

что такой алгоритм можно использовать для полиномиальной проверки существования расписания S длины $C_{\max}(S) \leq 4$ для заданного примера I. В самом деле, пусть $S_{\mathcal{A}}$ — расписание, построенное алгоритмом $\mathcal{A}$. Если $C_{\max}(S_{\mathcal{A}}) \leq 4$, то мы в явном виде нашли расписание длины не больше 4 (ответ на наш вопрос: "да"). Если же $C_{\max}(S_{\mathcal{A}}) \geq 5$, то из оценки $C_{\max}(S_{\mathcal{A}}) \leq C \cdot C^*_{\max}(I)$ при $C < 5/4$ следует $C^*_{\max}(I) > \frac{4}{5} C_{\max}(S_{\mathcal{A}}) \geq 4$, т.е. для данного примера не существует расписания длины не больше 4 (ответ "нет") □

37 Точный и приближенный алгоритмы для многопроцессорных систем открытого типа

В этом разделе мы рассмотрим наиболее общую систему открытого типа, как она определялась в п.2 главы 22. От системы OS ее отличает два существенных момента:

- количество операций (r_j) каждой работы $J_j \in \mathcal{J}$ является параметром, не зависящим от числа машин (m);
- множество $\mathcal{M}^j_q \subseteq \mathcal{M}$ допустимых исполнителей каждой операции o^j_q может быть произвольным подмножеством $\mathcal{M}$.

Второй из моментов не оказывает существенного влияния на точность получаемых решений, увеличивая лишь итоговую трудоемкость решения задачи. Действительно, применение алгоритма $\mathcal{A}_{19.1}$ из второй части позволяет избавиться от альтернативности исполнителей и дает такое допустимое распределение операций по исполнителям, при котором нагрузка каждой машины отличается от таковой в оптимальном расписании менее чем на величину $p_{\max}$. Таким образом, в случае произвольных длительностей мы получаем соотношение

$$L_{\max} < L^{\text{opt}}_{\max} + p_{\max}, \tag{37.1}$$

а в случае одинаковых (единичных) длительностей операций получаем оптимальное распределение: $L_{\max} = L^{\text{opt}}_{\max}$. Трудоемкость алгоритма $\mathcal{A}_{19.1}$ равна $O(R^2 m^2)$, где $R = \sum_{J_j \in \mathcal{J}} r_j$, — общее количество операций.

Полученная в результате система отличается от системы OS тем, что на каждой машине может быть более одной операции каждой работы. Вспоминая работу жадного алгоритма из раздела 28, мы видим, что этот факт может повлиять на трудоемкость жадного алгоритма, поскольку для отыскания очередной работы на освободившуюся машину M_i теперь может потребоваться просмотр не m, а $(m-1)(r-1)+1$ операций списка U_i, где $r = \max_j r_j$. Однако от этой неприятности мы легко избавимся, если на каждой машине все операции одной работы склеим в одну большую операцию (трудоемкость этой процедуры — $O(R + nm)$) и к полученной системе open shop применим произвольный жадный алгоритм (трудоемкости $O(nm^2)$). Длину получаемого этим алгоритмом расписания мы будем оценивать не через новое значение $p'_{\max}$ максимальной длительности операции (которое может значительно превосходить прежнее значение

$p_{\max}$), а опираясь на утверждение 29.1, из которого следует, что склейка операций не влияет на оценку величины простоя каждой машины:

$$D_i(C_i) \leq (r-1)p_{\max},\ \forall M_i \in \mathcal{M},$$

что с учетом (37.1) позволяет сформулировать следующий результат.

Теорема 37.1 *Для открытой системы общего вида с m машинами и R операциями существует алгоритм трудоемкости $O(R^2m^2)$, строящий приближенное расписание S с оценкой точности*

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) < rp_{\max},$$

где $r = \max_j r_j$ □

В случае одинаковых (единичных) длительностей операций после распределения операций по исполнителям исходная задача сводится к задаче нахождения оптимальной реберной раскраски двудольного мультиграфа $G = (\mathcal{J}, \mathcal{M};\ U)$, где каждой операции работы J_j на машине M_i соответствует ребро $(j,i) \in U$. Из теоремы Кёнига (см., например, Следствие 4 в [7], стр. 107) следует, что ребра двудольного мультиграфа можно покрасить в Δ цветов, где Δ — максимальная степень вершины. В нашем случае Δ равна максимуму величин r и $L_{\max} = L^{\text{opt}}_{\max}$. Так как обе величины являются нижними оценками оптимума исходной задачи, то расписание, получающееся из раскраски мультиграфа, является оптимальным. А поскольку трудоемкость $O(|U|\log^2(n+m))$ одного из алгоритмов, предложенных Габоу и Каривом в [85], мажорируется трудоемкостью распределения операций по машинам, то это дает оценку трудоемкости $O(R^2m^2)$ всего алгоритма построения оптимального расписания в исходной модели, что и формулируется в следующей теореме.

Теорема 37.2 *Для открытой системы общего вида с m машинами и R операциями одинаковой длины существует алгоритм построения оптимального расписания трудоемкости $O(R^2m^2)$* □

Заметим, что в случае единичных длительностей операций функция $O(R^2m^2)$ все же является полиномом от длины записи входной информации для системы рассматриваемого вида, т.к. для записи каждого из множеств $\mathcal{M}^j_q$ требуется m-битовое число, что в сумме по всем операциям составляет Rm битов.

38 Заключительные замечания и открытые вопросы

Много нерешенных открытых вопросов остается в области исследования нормальных классов примеров задачи open shop.

1. Было бы интересно установить какое-либо соотношение между функциями $\eta_n(m)$ и $\eta_e(m)$, определенными в пункте 6 введения к данной главе. Например, неравенство

$\eta_n(m) < \eta_e(m)$ для какого-то m означало бы, что мы можем доказать нормальность любого примера с $L_{\max} \in (\eta_n(m), \eta_e(m))$, но не можем создать полиномиального алгоритма (если $P \neq NP$) построения оптимального расписания для любого примера из данного класса. Но если бы такое соотношение между $\eta_n(m)$ и $\eta_e(m)$ действительно имело место, оставалось бы неясным, как доказывать NP-трудность задачи для данного класса примеров. Действительно, в этом случае мы не можем использовать традиционное сведение оптимизационной задачи к задаче распознавания "верно ли, что $C^*_{\max} \leq L_{\max}$?" (Мы бы заранее знали, что ответ на этот вопрос — "да".)

С другой стороны, соотношение $\eta_n(m) > \eta_e(m)$ для каких-то значений m означало бы, что для класса примеров со свойством $L_{\max} \in (\eta_e(m), \eta_n(m))$ мы можем эффективно строить их оптимальные расписания, несмотря на то, что они не обязательно являются нормальными. (И такой класс примеров представлял бы несомненный интерес.)

2. Первый вопрос может быть обобщен следующим образом: существует ли нормализующий вектор, который не является эффективно нормализующим?

3. Можно ли нижнюю оценку функции $\eta_n(m)$, представленную в теореме 27.1, усилить до $\eta_n(m) \geq 2m$? Утвердительный ответ на этот вопрос позволил бы заключить, что вектор $(0, \ldots, 0, \eta_n(m-1))$ является *MN*-вектором в $\mathbb{R}^m$ для любого $m \geq 2$.

4. Существуют ли *MN*-векторы с нецелочисленными компонентами?

5. Сколько на самом деле *MN*- и *MEN*-векторов в пространстве $\mathbb{R}^4$? (Мы знаем, что существует не менее двух *MN*- и двух *MEN*-векторов в $\mathbb{R}^4$.)

Глава 2

Системы поточного типа

39 Приближенное решение задачи flow shop с использованием нестрогого суммирования векторов

Теорема 39.1 *Пусть имеется алгоритм $\mathcal{A}$, который для некоторого $m \geq 2$ для любого $\hat{s}$-семейства векторов $\{x_1, \ldots, x_n\} \subset \mathbb{R}^{m-1}$ с трудоемкостью $\mathcal{T}_{\mathcal{A}}(m-1, n)$ решает задачу* HCB2$(m-1)$ *с гарантированной оценкой*

$$\theta_2(b_{m-1}) \leq \theta_2^*(m-1).$$

Тогда существует алгоритм $\mathcal{A}_{39.1}$, который для всякого примера задачи FS *с m машинами и n работами с трудоемкостью $O(\mathcal{T}_{\mathcal{A}}(m-1, n) + mn)$ находит перестановочное расписание S_π с оценкой длины*

$$C_{\max}(S_\pi) \leq L_{\max} + (m - 1 + \theta_2^*(m-1))p_{\max}.$$

Доказательство. Алгоритм построения искомого расписания состоит из двух этапов.

Алгоритм $\mathcal{A}_{39.1}$

Этап 1. С помощью алгоритма $\mathcal{A}$ решаем задачу HCB2$(m-1)$ для семейства векторов $D = \{d_1, \ldots, d_n\} \subset \mathbb{R}^{m-1}$, определенного согласно (22.11), с гарантированной оценкой функционала θ_2, заявленной в теореме. В результате получаем перестановку $\pi = (\pi_1, \ldots, \pi_n)$, задающую нестрогое суммирование векторов из D в таком семействе полупространств

$$\{P(e_1 - e_2, \beta_1), \ldots, P(e_{m-1} - e_m, \beta_{m-1})\}$$

пространства $\mathbb{R}^{m-1}$ (здесь предполагаем, что $e_m = 0$), что

$$\sum_{i=1}^{m-1} \beta_i \leq \theta_2^*(m-1)p_{\max}. \tag{39.1}$$

Этап 2. По перестановке π строим перестановочное расписание S_π.
Алгоритм $\mathcal{A}_{39.1}$ описан.

Докажем, что построенное расписание S_π удовлетворяет требуемой оценке точности. Используя обозначения (22.12) и (22.13), преобразуем формулу (22.16) в верхнюю оценку на $C_{\max}(S_\pi)$:

$$C_{\max}(S_\pi) = \max_{1\le k_1\le\dots\le k_{m-1}\le n}((P_\pi^{k_1}(1) - P_\pi^{k_1-1}(2)) + (P_\pi^{k_2}(2) - P_\pi^{k_2-1}(3)) + \dots$$

$$+(P_\pi^{k_{m-1}}(m-1) - P_\pi^{k_{m-1}-1}(m))) + L_m \le \sum_{i=1}^{m-1}\max_k(P_\pi^k(i) - P_\pi^{k-1}(i+1)) + L_{\max} =$$

(используя (39.1) и замечание 22.1)

$$= L_{\max} + \sum_{i=1}^{m-1}\max_k(P_\pi^k(i,i+1) \le$$

$$\le L_{\max} + (m-1)p_{\max} + \sum_{i=1}^{m-1}\beta_i \le L_{\max} + (m-1+\theta_2^*(m-1))p_{\max}.$$

Теорема 39.1 доказана.

Из теоремы 39.1 и леммы 14.4 непосредственно вытекает

Теорема 39.2 *Пусть имеется алгоритм $\mathcal{A}$, который при некотором $m \ge 3$ для любого $\hat{s}$-семейства векторов $\{x_1,\dots,x_n\} \subset \mathbb{R}^{m-2}$ с трудоемкостью $\mathcal{T}_\mathcal{A}$ решает задачу* HCB1$(m-2)$ *с гарантированной оценкой*

$$\theta_1(b_{m-1}) \le \theta_1^*(m-2).$$

Тогда существует алгоритм $\mathcal{A}'$ который для любого примера задачи FS *с m машинами и n работами с трудоемкостью $O(\mathcal{T}_\mathcal{A} + mn)$ находит перестановочное расписание S_π с оценкой длины*

$$C_{\max}(S_\pi) \le L_{\max} + (m-1+\theta_1^*(m-2))p_{\max}.$$ □

С учетом последней теоремы и следствия 14.7 получаем следующий результат.

Теорема 39.3 *Существует алгоритм трудоемкости $O(n\log n)$, который для любого примера задачи flow-shop с четырьмя машинами и n работами строит перестановочное расписание S_π с оценкой длины*

$$C_{\max}(S_\pi) \le L_{\max} + 6p_{\max}.$$ □

В случае трех машин задача FS, как это явствует из теоремы 39.2, сводится к решению задачи НСВ1 в одномерном пространстве, т.е. к нахождению нестрогого суммирования $\hat{s}$-семейства чисел (по модулю не превосходящих 1 и с нулевой суммой) в семействе одномерных полупространств $\{P(-e_2, \beta_2), P(e_2, \beta_1)\}$ с минимальным значением $\beta_1 + \beta_2$. Это означает, что требуется нестрого просуммировать числа внутри отрезка минимальной длины. Нетрудно понять, что для решения такой задачи можно предложить приближенный алгоритм линейной трудоемкости, гарантирующий нахождение нестрогого суммирования любого заданного $\hat{s}$-семейства чисел внутри любого отрезка длины 1, содержащего точку нуль. Таким образом, для задачи FS(3) мы получаем следующий результат.

Теорема 39.4 *Существует алгоритм трудоемкости $O(n)$, который для любого примера задачи flow-shop с тремя машинами и n работами строит перестановочное расписание S_π с неулучшаемой оценкой длины*

$$C_{\max}(S_\pi) \le L_{\max} + 3p_{\max}.$$

Неулучшаемость этой оценки вытекает из теоремы 40.1, доказываемой в следующем разделе.

Для приближенного решения задачи flow shop с числом машин большим четырех требуется найти приближенное решение задачи НСВ1(m) в пространстве размерности $m \ge 3$. Используя в качестве алгоритма решения задачи НСВ1(m) какой-либо алгоритм компактного суммирования векторов, приходим к следующему результату.

Теорема 39.5 *Пусть имеется алгоритм $\mathcal{A}$, который при любых натуральных n и m для любого $\hat{s}$-семейства векторов $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$ с трудоемкостью $\mathcal{T}_{\mathcal{A}}(m, n)$ решает задачу* КСВ *с гарантированной оценкой*

$$f_{\hat{s},X}(\pi) \le C^{\mathcal{A}}_{\hat{s},m}.$$

Тогда существует алгоритм $\mathcal{A}'$, который для любого примера задачи FS *с m машинами и n работами с трудоемкостью $O(\mathcal{T}_{\mathcal{A}}(m-2, n)+mn)$ находит перестановочное расписание S_π с оценкой длины*

$$C_{\max}(S_\pi) \le L_{\max} + (m-1)(C^{\mathcal{A}}_{\hat{s},m-2} + 1)p_{\max}.$$

Доказательство. Пусть $X = \{x_1, \dots, x_n\}$ — $\hat{s}$-семейство векторов в пространстве $\mathbb{R}^{m-2}$. С помощью алгоритма $\mathcal{A}$ находим перестановку π, обеспечивающую оценку $f_{\hat{s},X}(\pi) \le C^{\mathcal{A}}_{\hat{s},m-2}$. Последняя означает, что для любого $k \in \mathbb{N}_n$ частичная сумма x^k_π лежит в шаре $\hat{s}$-нормы радиуса $C^{\mathcal{A}}_{\hat{s},m-2}$. Отсюда с учетом определения нормы $\hat{s}$ следует, что

$$|x^k_\pi(i_1)| \le C^{\mathcal{A}}_{\hat{s},m-2},\ |x^k_\pi(i_1) - x^k_\pi(i_2)| \le C^{\mathcal{A}}_{\hat{s},m-2},\ \forall i_1, i_2.$$

В частности,

$$(x^k_\pi, e_i - e_{i+1}) \le C^{\mathcal{A}}_{\hat{s},m-2},\ i \in \mathbb{N}_{m-3};$$

$$(x_\pi^k, e_{m-2} \le C^{\mathcal{A}}_{\hat{s},m-2},\ (x_\pi^k, -e_1) \le C^{\mathcal{A}}_{\hat{s},m-2},$$

т.е. перестановка π обеспечивает нестрогое суммирование векторов из X в семействе полупространств

$$\{P(e_1 - e_2, \beta_1), \ldots, P(e_{m-3} - e_{m-2}, \beta_{m-3}), P(e_{m-2}, \beta_{m-2}), P(-e_1, \beta_{m-1})\}$$

со значением $\beta_i = C^{\mathcal{A}}_{\hat{s},m-2},\ i \in \mathbb{N}_{m-1}$. Таким образом, для любого $\hat{s}$-семейства векторов $X \subset \mathbb{R}^{m-2}$ мы получаем решение задачи НСВ1$(m-2)$ с оценкой

$$\theta_1(b_{m-1}) \le \theta_1^*(m-2) \doteq C^{\mathcal{A}}_{\hat{s},m-2},$$

что с учетом теоремы 39.2 позволяет строить перестановочное расписание S_π с оценкой длины

$$C_{\max}(S_\pi) \le L_{\max} + (m-1)(C^{\mathcal{A}}_{\hat{s},m-2} + 1)p_{\max}.$$

Теорема 39.5 доказана.

Из теоремы 39.5 и следствия 11.8 получаем следующее утверждение.

Теорема 39.6 *Существует алгоритм трудоемкости $O(n^2m^2)$, который для любого примера задачи flow shop с m машинами и n работами находит перестановочное расписание с оценкой длины*

$$C_{\max}(S_\pi) \le L_{\max} + (m-1)\left(m - 2 + \frac{1}{m-2}\right)p_{\max} \quad \square \tag{39.2}$$

Отметим, что следствие 14.13 (стр. 117) позволяет строить расписания, не только удовлетворяющие указанной выше оценке, но и обладающие специальными свойствами (они потребуются нам при построении расписаний в разделе 4).

Лемма 39.7 *Для любого примера задачи flow shop с тремя машинами и n работами, удовлетворяющего (22.9) и $p_{\max} = 1$, и любого числа $b \in [1,2]$ с трудоемкостью $O(n \log n)$ находятся перестановочные расписания $S_{\pi'}, S_{\pi''}$, в которых 1-я машина работает непрерывно в интервале $[0, L_{\max}]$, 2-я — в интервале $[b, b + L_{\max}]$, 3-я — в интервале $[3, 3 + L_{\max}]$, и при этом*

$$\sum_{j=1}^{k} p_{\pi'_j 1} \le (b-1) + \sum_{j=1}^{k} p_{\pi'_j 2},\ k \in \mathbb{N}_n; \tag{39.3}$$

$$\sum_{j=1}^{k} p_{\pi''_j 2} \le (2-b) + \sum_{j=1}^{k} p_{\pi''_j 3},\ k \in \mathbb{N}_n \quad \square \tag{39.4}$$

Доказательство. Пусть задано число $b \in [1,2]$. Положим $\beta = b-1$ и применим алгоритм из следствия 14.13 для нахождения перестановок π', π'' векторов семейства $D = \{d_1, \ldots, d_n\}$, определенных согласно (22.11). Поскольку перестановка π', согласно следствию 14.13, задает строгое суммирование векторов из D в полупространстве $P(e_1 - e_2, \beta)$, то

$$(d^k_{\pi'}, e_1 - e_2) \le \beta, \ \forall k \in \mathbb{N}_n,$$

откуда следует $d^k_{\pi'}(1) - d^k_{\pi'}(2) \le \beta$, или

$$\sum_{j=1}^{k} p_{\pi'_j 1} - \sum_{j=1}^{k} p_{\pi'_j 2} \le b-1, \tag{39.5}$$

т.е. имеем (39.3). Из (39.5) получаем также соотношения

$$\sum_{j=1}^{k} p_{\pi'_j 1} - \sum_{j=1}^{k} p_{\pi'_j 2} \le b - 1 + p_{\pi'_k 2} \le b, \ k \in \mathbb{N}_n,$$

из которых следует, что при организации работы первой машины в интервале $[0, L_{\max}]$, а второй — в интервале $[b, b+L_{\max}]$ для любой работы $J_{\pi'_j}$, $j \in \mathbb{N}_n$, не возникает накладок ее операций на машинах M_1 и M_2.

Из нестрогого суммирования векторов D в полупространстве $P(e_2, 1-\beta) = P(e_2 - e_3, 1-\beta)$ (где $e_3 = 0$) и замечания 22.1 (стр. 163) следует оценка

$$P^k_\pi(2,3) \le 2 - \beta = 3 - \beta,$$

или согласно определениям (22.13), (22.12),

$$\sum_{j=1}^{k} p_{\pi'_j 2} \le 3 - b + \sum_{j=1}^{k-1} p_{\pi'_j 3}, \ k \in \mathbb{N}_n.$$

Последние соотношения означают, что если непрерывная работа машин M_2 и M_3 организована, соответственно, в интервалах $[b, b+L_{\max}]$ и $[3, 3+L_{\max}]$, то накладок операций любой работы J_j на машинах M_2 и M_3 также не возникает. Таким образом, перестановочное расписание $S_{\pi'}$, в котором машины M_1, M_2 и M_3 работают, соответственно, в интервалах $[0, L_{\max}]$, $[b, b+L_{\max}]$ и $[3, 3+L_{\max}]$, является допустимым.

Соотношения (39.4) и допустимость перестановочного расписания $S_{\pi''}$, построенного в тех же интервалах, доказывается аналогично □

40 Интервалы локализации оптимумов задачи flow shop

Теорема 39.6 позволяет определить функции:

$$\tilde{\mu}_{FS}(m) = \sup\left(C_{\max}(S_{\text{opt}}) - L_{\max}\right)/p_{\max},$$

где sup берется по всем входам задачи FS с m машинами и нетождественно равными нулю длительностями операций, S_{opt} — оптимальное расписание для заданного входа, $L_{\max}$ и $p_{\max}$ — функции от входа, определенные в п.4 раздела 22;

$\tilde{\mu}'_{FS}(m)$ — аналогично определяемая функция для перестановочных оптимумов задачи FS.

Из определения этих функций следует, что оптимум любого примера задачи FS(m) находится в интервале $[L_{\max}, L_{\max} + \tilde{\mu}_{FS}(m)p_{\max}]$, длина которого не зависит от числа работ n, и границы этого интервала точны. Аналогично, перестановочный оптимум любого примера задачи FS(m) находится в интервале $[L_{\max}, L_{\max} + \tilde{\mu}'_{FS}(m)p_{\max}]$.

Ясно, что $\tilde{\mu}_{FS}(m) \le \tilde{\mu}'_{FS}(m),\ \forall m$. Из известного факта, что при $m \le 3$ оптимум задачи FS достигается на перестановочном расписании, имеем равенство

$$\tilde{\mu}_{FS}(m) = \tilde{\mu}'_{FS}(m),\ \text{при } m \le 3. \tag{40.1}$$

Кроме того, из теоремы 39.6 получаем оценку сверху:

$$\tilde{\mu}'_{FS}(m) \le m^2 - 3m + 3 + \frac{1}{m-2}.$$

В следующей теореме мы получим нижнюю оценку на функцию $\tilde{\mu}'_{FS}(m)$.

Теорема 40.1 $\tilde{\mu}'_{FS}(m) \ge m - 1 + \left\lfloor \frac{m-1}{2} \right\rfloor$.

Доказательство. Пусть задано произвольное $\varepsilon \in (0,1)$. Построим пример m-машинной задачи FS с $p_{\max} = 1$, удовлетворяющей оценке

$$C_{\max}(S_{\text{opt}}) > L_{\max} + m - 1 + \left\lfloor \frac{m-1}{2} \right\rfloor - \varepsilon.$$

Возьмем целые p и q такие, что

$$p \ge \frac{m-1}{\varepsilon} - 1,\ q > \frac{2(m-1)p}{\varepsilon}. \tag{40.2}$$

Семейство работ $\mathcal{J}$ состоит из работ двух типов: q A-работ с одинаковым вектором длительностей операций $a \in \mathbb{R}^m$ и p B-работ с вектором длительностей $b \in \mathbb{R}^m$, где

$$a = \begin{cases} (1-\delta, 1, 1-\delta, 1, \ldots, 1, 1-\delta), & \text{для нечетных m,} \\ (1-\delta, 1, \ldots, 1-\delta, 1, 1-\delta, 1-\delta), & \text{для четных m,} \end{cases}$$
$$b = \begin{cases} (1, 0, 1, 0, \ldots, 0, 1), & \text{для нечетных m,} \\ (1, 0, \ldots, 1, 0, 1, 1), & \text{для четных m,} \end{cases}$$

$\delta = p/q;\ n = p + q$. Ясно, что $p_{\max} = 1$ и $L_{\max} = L_i = q,\ \forall i \in \mathcal{M}$.

Пусть π — оптимальная перестановка номеров работ $\mathcal{J}$ (можем считать, что $\pi = (1, 2, \ldots, n))$; π' обозначает ее подпоследовательность из номеров A-работ. В перестановке π B-работы разрезают последовательность π' на t' непустых сегмента ($t' \le p+1$) :

$\pi_1, \ldots, \pi_{t'}$. Пусть $t = \min\{t', \lfloor \frac{m-1}{2} \rfloor\}$ и пусть $\pi_{i_1}, \ldots, \pi_{i_t}$ $(i_1 < i_2 < \cdots < i_t)$ — t наиболее длинных сегментов, L — их суммарная длина. Тогда в случае $t = t'$ имеем

$$L\delta = q\delta = p \geq \left\lceil \frac{m-1}{\varepsilon} \right\rceil - 1 > m - 1.$$

В противном случае (если $t = \lfloor (m-1)/2 \rfloor$),

$$L\delta \geq \frac{tq}{t}\delta \geq \frac{tq}{p+1} \cdot \frac{p}{q} = t - \frac{t}{p+1} \geq \left\lfloor \frac{m-1}{2} \right\rfloor - \frac{m-1}{2(p+1)} \geq \left\lfloor \frac{m-1}{2} \right\rfloor - \frac{\varepsilon}{2}.$$

Таким образом, неравенство

$$L\delta \geq \left\lfloor \frac{m-1}{2} \right\rfloor - \frac{\varepsilon}{2} \tag{40.3}$$

выполняется в обоих случаях. Пусть $\pi_{i_j} = (s_j, s_j + 1, \ldots, f_j)$. Оценим снизу длину перестановочного расписания S_π, выбрав параметры $\{k_1, \ldots, k_{m-1}\}$ в формуле (22.16) следующим образом:

$$k_{2j-1} = s_j,\ k_{2j} = f_j,\ \forall j \in \mathbb{N}_t,$$
$$k_j = k_{2t},\ \forall j > 2t.$$

Нетрудно проверить, что:

- сумма $(\bar{S})$ элементов $\{p_{ji}\}$ в правой части (22.16) содержит ровно $(n + m - 1)$ элементов;
- $\bar{S}$ не содержит нулевых элементов; это означает, что все ее элементы равны 1 или $1 - \delta$;
- в $\bar{S}$ содержится ровно $p + L$ единичных элементов.

Используя (40.2), (40.3), получаем нижнюю оценку длины расписания:

$$C_{\max}(S_\pi) \geq \bar{S} \geq (q + p + m - 1)(1 - \delta) + (p + L)\delta \geq L_{\max} + m - 1 + p - q\delta$$

$$-p\delta - (m-1)\delta + p\delta + L\delta \geq L_{\max} + m - 1 - (m-1)\delta + \left\lfloor \frac{m-1}{2} \right\rfloor - \frac{\varepsilon}{2}$$

$$> L_{\max} + m - 1 + \left\lfloor \frac{m-1}{2} \right\rfloor - \varepsilon.$$

Теорема 40.1 доказана.

Полученные выше результаты позволяют найти точные интервалы локализации оптимумов задачи FS в случае двух и трех машин.

Следствие 40.2

$$4 \le \tilde{\mu}'_{FS}(4) \le 6, \tag{40.4}$$

$$\tilde{\mu}_{FS}(3) = \tilde{\mu}'_{FS}(3) = 3, \tag{40.5}$$

$$\tilde{\mu}_{FS}(2) = \tilde{\mu}'_{FS}(2) = 1. \tag{40.6}$$

Доказательство. Оценки (40.4) вытекают из теорем 39.3 и 40.1. Равенство (40.5) вытекает из теорем 40.1, 39.4 и равенства (40.1).

При $m = 2$ простым алгоритмом трудоемкости $O(n)$ строится расписание S_π с оценкой $C_{\max}(S_\pi) \le L_{\max} + p_{\max}$. (Он ставит в перестановку π сначала все работы с $p_{j1} \le p_{j2}$, а затем — все остальные.)

Оценка $\tilde{\mu}_{FS}(2) \ge 1$ достигается на любом входе с операциями тождественно единичной длины, что вместе с (40.1) дает равенство (40.6) □

Из следствия 40.2 следует, что оптимум любого входа задачи FS при $m = 2$ лежит в интервале $[L_{\max}, L_{\max} + p_{\max}]$, а при $m = 3$ — в интервале $[L_{\max}, L_{\max} + 3p_{\max}]$, и границы этих интервалов точны.

41 Многопроцессорная задача flow shop

В этом разделе мы рассмотрим поточную систему общего вида, как она определена в п.2 раздела 22. С использованием теоремы 11.5 о суммировании векторов будет построен полиномиальный алгоритм нахождения приближенного расписания с абсолютной оценкой точности, не зависящей от числа работ.

Напомним обозначения из раздела 22:

r — количество последовательных цехов поточной системы, совпадающее с количеством операций каждой работы $J_j \in \mathcal{J}$;

$\mathcal{M}_q$ — множество (эквивалентных) машин q-го цеха. Обозначим также:

$m_q = |\mathcal{M}_q|$ — количество альтернативных машин для q-й операции каждой работы;

$L_q = \sum_{J_j \in \mathcal{J}} p_q^j / m_q$, — величина средней нагрузки машин $M_i \in \mathcal{M}_q$;

$L_{\max} = \max_q L_q$; $m = \sum_{q=1}^{r} m_q$, — общее число машин.

Справедлива нижняя оценка длины любого допустимого расписания поточной системы:

$$C_{\max}(S) \ge L_{\max}.$$

Из доказываемой ниже теоремы вытекает, что оптимум быстродействия любой поточной системы лежит в интервале $[L_{\max},\ L_{\max} + \mu(r)p_{\max}]$, длина которого в худшем случае не превосходит квадратичной функции от r, уменьшается с ростом числа альтернативных исполнителей и становится линейной от r при числе исполнителей на каждой стадии: $m_q = \Omega(r)$, что является рекордом точности среди оценок рассматриваемого типа для задач типа job shop.

Теорема 41.1 *Для r-стадийной многопроцессорной задачи flow shop с n работами и m_q машинами на q-й стадии ($q \in \mathbb{N}_r$) с трудоемкостью $O(n^2r^2)$ строится расписание S с оценкой длины*

$$C_{\max}(S) \le L_{\max} + \left(r^2 - 3r + 3 + \frac{1}{r-2}\right) p_{\max}; \tag{41.1}$$

при $r = 3$ с трудоемкостью $O(n \log m)$ строится расписание с оценкой

$$C_{\max}(S) \le L_{\max} + 4p_{\max}, \tag{41.2}$$

а при $r = 2$ с трудоемкостью $O(n \log m)$ строится расписание S с оценкой

$$C_{\max}(S) \le L_{\max} + \left(2 - \frac{1}{m_{\min}}\right) p_{\max}, \tag{41.3}$$

где $m_{\max} = \max_q m_q$, $m_{\min} = \min_q m_q$.

Доказательство. В последующих выкладках предполагаем выполненным $p_{\max} = 1$.

Расписание S строится согласно описанной ниже схемы, состоящей из 5 шагов.

Шаг 0 (выравнивание нагрузок машин).

Выравниваем величины L_i, оставляя неизменными величины $L_{\max}$ и $p_{\max}$. Для этого находим наиболее загруженный цех $\mathcal{M}_{q^*}$ (для которого выполнено $L_{q^*} = L_{\max}$) и для каждого $q \ne q^*$, такого что $L_q < L_{\max}$, выполняем следующие три процедуры.

A. Уменьшаем m_q до тех пор, пока не выполнится либо неравенство $m_q \le m_{q^*}$, либо соотношения

$$\frac{1}{m_q}\sum_j p_q^j \le L_{\max} < \frac{1}{m_q - 1}\sum_j p_q^j \tag{41.4}$$

(либо оба). Ясно, что изменение величин m_q не повлечет изменения величины $m_{\min}$.

B. Увеличиваем длительности операций $\{p_q^j \mid j \in \mathbb{N}_n\}$ (не превышая $p_{\max} = 1$), пока не выполнится либо

$$L_q = L_{\max}, \tag{41.5}$$

либо $p_q^j = 1$, $\forall\, j \in \mathbb{N}_n$. Если выполнилось последнее, т.е. при максимально возможных длительностях операций не удалось выровнять нагрузки цехов, то применяем процедуру C.

C. Вводим дополнительные (фиктивные) операции q-го цеха максимально возможной длины, пока не выполнится (41.5).

По окончании процедуры выравнивания нагрузок по всем цехам объединяем фиктивные операции разных цехов в фиктивные работы.

Нетрудно видеть, что если для некоторого q либо изначально выполнялось $m_q \le m_{q^*}$, либо после процедуры A было достигнуто равенство $m_q = m_{q^*}$, то нам нет необходимости применять процедуру C, т.к. равенство (41.5) получается сразу после B. Если же после процедуры A мы имеем $m_q > m_{q^*}$, то из (41.4) следует, что в процедуре C будет добавлено более $n/(m_q - 1)$ дополнительных операций, т.е. общее число работ не более чем удвоится.

В последующих выкладках будем предполагать, что величины $n,\ m_q$ и p_q^j принимают свои новые значения, при которых для каждого $q \in \mathbb{N}_r$ выполнено равенство (41.5). (При этом никак не будем различать исходные и фиктивные работы.) Для каждой работы $J_j,\ j \in \mathbb{N}_n$, определим векторы

$$d_j = (d_j(1), \ldots, d_j(r)) = (p_1^j/m_1, \ldots, p_r^j/m_r),$$

$$b_j = (b_j(1), \ldots, b_j(r-1)) = (d_j(1) - d_j(r), \ldots, d_j(r-1) - d_j(r)),$$

и обозначим $D \doteq \sum_{j=1}^n d_j = (L_{\max}, \ldots, L_{\max})$. Тогда будем иметь $\sum_{j=1}^n b_j = 0$.

Шаг 1 (нахождение перестановки работ).

Разнообразие алгоритмов, применяемых для решения данной задачи, в основном определяется разнообразием алгоритмов, используемых на шаге 1. Пока не уточняя, какой из алгоритмов мы здесь используем, обозначим найденную им перестановку работ через π. Для упрощения выкладок будем считать, что работы занумерованы согласно перестановке π, т.е. $\pi = (1, 2, \ldots, n)$.

Шаг 2 (распределение операций цехов $q = 1, \ldots, r-1$ по машинам).

Для каждого $q \in \mathbb{N}_{r-1}$ операции $o_q^1, \ldots, o_q^n$ распределим в этом порядке по m_q машинам из множества $\mathcal{M}_q$ согласно правилу LS (List Scheduling): очередную операцию - на наименее загруженную машину. На каждой из машин $M_i \in \mathcal{M}_q$ распределенные на нее операции выполняем в порядке поступления, т.е. согласно перестановке $\pi = (1, 2, \ldots, n)$.

Шаг 3 (распределение операций цеха r по машинам).

Операции цеха r распределяем по машинам согласно правилу RLS (Reverse List Scheduling, [80, 97]), которое состоит в следующем. Распределяем операции по машинам согласно правилу LS, но в обратном порядке: $o_r^n, o_r^{n-1}, \ldots, o_r^1$. Распределенные на машину $M_i \in \mathcal{M}_r$ операции выполняем в порядке, обратном их поступлению (т.е. вновь согласно перестановке $\pi = (1, 2, \ldots, n)$).

Шаг 4 (построение расписания).

Для найденного порядка выполнения операций на машинах, строим наиболее раннее расписание выполнения операций, согласованное по работам.

Схема построения расписания описана.

Определим векторы $\{a_j = (a_j(1), \ldots, a_j(m)) \mid j \in \mathbb{N}_n\}$, характеризующие загруженность каждой из m машин работой J_j, т.е. $a_j(i) = p_q^j$, если в результате шага 2 или 3 операция o_q^j распределена на машину $M_i \in \mathcal{M}_q$, и $a_j(i) = 0$, — если никакая операция работы J_j не выполняется машиной M_i.

Положим $a^k = \sum_{j=1}^{k} a_j,\ k \in \mathbb{N}_n;\ A = a^n$. Тогда для любых $k \in \mathbb{N}_n,\ q \in \mathbb{N}_r$ и любых двух машин $M_{i'}, M_{i''} \in \mathcal{M}_q,\ i' \neq i''$, справедливы соотношения

$$0 \leq a^k(i') \leq a^k(i'') + 1, \tag{41.6}$$

$$\sum_{M_i \in \mathcal{M}_q} a^k(i) \leq \sum_{M_i \in \mathcal{M}_q} a^{k-1}(i) + 1, \tag{41.7}$$

$$d^k(q) = \frac{1}{m_q} \sum_{M_i \in \mathcal{M}_q} a^k(i), \tag{41.8}$$

с помощью которых получаем

$$d^k(q) \geq \frac{1}{m_q}(m_q a^k(i') - (m_q - 1)) = a^k(i') - \frac{m_q - 1}{m_q}, \tag{41.9}$$

$$d^k(q) \leq \frac{1}{m_q}(m_q a^k(i'') + (m_q - 1)) = a^k(i'') + \frac{m_q - 1}{m_q}, \tag{41.10}$$

$$d^k(q) \leq \frac{1}{m_q}\left(\sum_{M_i \in \mathcal{M}_q} a^{k-1}(i) + 1\right) \leq \frac{1}{m_q}(m_q a^{k-1}(i'') + m_q) = a^{k-1}(i'') + 1, \tag{41.11}$$

$$\begin{aligned} d^{k-1}(q) + 1 = \frac{1}{m_q}\left(\sum_{M_i \in \mathcal{M}_q} a^{k-1}(i) + m_q\right) &\geq \frac{1}{m_q}\left(\sum_{M_i \in \mathcal{M}_q} a^k(i) + m_q - 1\right) \\ &\geq \frac{1}{m_q}(m_q a^k(i')) = a^k(i'). \end{aligned} \tag{41.12}$$

Введем обозначения для дополняющих сумм векторов $\{a_j\}$ и векторов $\{d_j\}$:

$$\bar{a}^k \doteq \sum_{j=n-k+1}^{n} a_j = A - a^{n-k},\ \bar{d}^k \doteq \sum_{j=n-k+1}^{n} d_j = D - d^{n-k}.$$

Нетрудно видеть, что при распределении операций по машинам r-го цеха согласно правилу RLS для "надчеркнутых" величин $\bar{a}^k(i)$ и $\bar{d}^k(q)$ выполняются соотношения, аналогичные (41.6)–(41.8), из которых выводятся аналоги соотношений (41.9)–(41.12). В частности, получаем аналоги для (41.9) и (41.12):

$$\bar{a}^k(i') \leq \bar{d}^k(q) + \frac{m_q - 1}{m_q}, \tag{41.13}$$

$$\bar{a}^k(i') \leq \bar{d}^{k-1}(q) + 1. \tag{41.14}$$

Построим сеть $G(\pi)$, задающую технологию выполнения всего множества операций. Для этого каждую операцию o_q^j, как и ранее для задачи flow shop, будем представлять в G вершиной веса p_q^j, причем операции машины M_i составят i-й горизонтальный ряд

верн. Эти вершины соединим между собой цепочкой дуг, задающих порядок выполнения операций машины M_i согласно перестановке $\pi = (1, 2, \ldots, n)$. (Всего, очевидно, в сети G будет не более $rn - m$ "горизонтальных" дуг.) Порядок выполнения каждой из n работ зададим "вертикальными" дугами вида (o_q^j, o_{q+1}^j), т.е. каждая вертикальная дуга переводит нас из горизонтального ряда множества $\mathcal{M}_q$ в ряд множества $\mathcal{M}_{q+1}$. Всего в G имеется $(r-1)n$ вертикальных дуг, что вместе с горизонтальными составляет $O(rn)$ дуг. Веса всех дуг нулевые.

Как мы знаем из календарного планирования, длина раннего расписания S, построенного на четвертом шаге описанной выше схемы, равна длине критического пути в сети G. Всякий такой путь состоит из $r-1$ вертикальных дуг, соответствующих каким-то работам $k_1, k_2, \ldots, k_{r-1}$, $(1 \le k_1 \le k_2 \le \cdots \le k_{r-1} \le n)$, и из r горизонтальных цепей, соответствующих машинам $M_{i_1}, M_{i_2}, \ldots, M_{i_r}$, где $M_{i_q} \in \mathcal{M}_q$. Таким образом, длина критического пути выражается формулой:

$$C_{\max}(S) = \max_{\{k_q\},\{i_q\}} \left(\sum_{j=1}^{k_1} a_j(i_1) + \sum_{j=k_1}^{k_2} a_j(i_2) + \cdots + \sum_{j=k_{r-1}}^{n} a_j(i_r) \right). \tag{41.15}$$

Далее предполагаем, что $\{k_q\}, \{i_q\}$ — как раз те значения, на которых в формуле (41.15) достигается максимум:

$$\begin{aligned} C_{\max}(S) &= \sum_{j=1}^{k_1} a_j(i_1) + \sum_{j=k_1}^{k_2} a_j(i_2) + \cdots + \sum_{j=k_{r-1}}^{n} a_j(i_r) \\ &\le \sum_{q=1}^{r-1} \left(a^{k_q}(i_q) - a^{k_q-1}(i_{q+1}) \right) + A(i_r). \end{aligned} \tag{41.16}$$

Обозначим $f_q \doteq a^{k_q}(i_q) - a^{k_q-1}(i_{q+1})$ и будем оценивать сверху величины f_q для $q \in \mathbb{N}_{r-2}$, и отдельно — величину

$$f_{r-1} + A(i_r) = a^{k_{r-1}}(i_{r-1}) + (A(i_r) - a^{k_{r-1}-1}(i_r)) = a^{k_{r-1}}(i_{r-1}) + \bar{a}^{n-k_{r-1}+1}(i_r).$$

Для $q \in \mathbb{N}_{r-2}$ из (41.9) и (41.11) получаем

$$f_q \le d^{k_q}(q) - d^{k_q}(q+1) + \frac{m_q - 1}{m_q} + 1 = b^{k_q}(q) - b^{k_q}(q+1) + 2 - \frac{1}{m_q}. \tag{41.17}$$

Из (41.12) и (41.10) получаем другую оценку на f_q:

$$\begin{aligned} f_q &\le d^{k_q-1}(q) - d^{k_q-1}(q+1) + 1 + \frac{m_{q+1} - 1}{m_{q+1}} \\ &= b^{k_q-1}(q) - b^{k_q-1}(q+1) + 2 - \frac{1}{m_{q+1}}. \end{aligned} \tag{41.18}$$

Также двумя способами оцениваем сверху величину $f_{r-1} + A(i_r)$. Из (41.9) и (41.14) получаем

$$f_{r-1} + A(i_r) = a^{k_{r-1}}(i_{r-1}) + \bar{a}^{n-k_{r-1}+1}(i_r) \le d^{k_{r-1}}(r-1) + \frac{m_{r-1}-1}{m_{r-1}} + \bar{d}^{n-k_{r-1}}(r) + 1$$

$$= (d^{k_{r-1}}(r-1) - d^{k_{r-1}}(r)) + 2 - \frac{1}{m_{r-1}} + D(r) = b^{k_{r-1}}(r-1) + 2 - \frac{1}{m_{r-1}} + D(r). \quad (41.19)$$

Из (41.12) и (41.13) получаем

$$f_{r-1} + A(i_r) \le d^{k_{r-1}-1}(r-1) + 1 + \bar{d}^{n-k_{r-1}+1}(r) + \frac{m_r - 1}{m_r} = (d^{k_{r-1}-1}(r-1) - d^{k_{r-1}-1}(r))$$

$$+2 - \frac{1}{m_r} + D(r) = b^{k_{r-1}-1}(r-1) + 2 - \frac{1}{m_r} + D(r). \quad (41.20)$$

Выбирая для каждого $f_q,\ q \in \mathbb{N}_{r-2}$, лучшую из оценок (41.17), (41.18), для $f_{r-1}+A(i_r)$ — лучшую из оценок (41.19), (41.20) и предполагая $b^k(r) = 0$, из (41.16) получаем оценку

$$C_{\max}(S) \le D(r) + 2(r-1)$$

$$+ \sum_{q=1}^{r-1} \min \left\{ (b^{k_q}, e_q - e_{q+1}) - \frac{1}{m_q}, (b^{k_q - 1}, e_q - e_{q+1}) - \frac{1}{m_{q+1}} \right\}. \quad (41.21)$$

Поскольку после шага 0 семейство векторов $\{b_j\}$ представляет собой 0-семейство, на шаге 1 для нахождения перестановки π мы можем применить один из алгоритмов решения задачи СВ2$(r-1)$ о суммировании векторов $X = \{b_1, \ldots, b_n\} \subset \mathbb{R}^{r-1}$ в семействе полупространств $\mathcal{P} = \{P(e_1 - e_2, \beta_1), \ldots, P(e_{r-2} - e_{r-1}, \beta_{r-2}), P(e_{r-1}, \beta_{r-1})\}$ с минимальной суммой $\sum_{i=1}^{r-1} \beta_i$. Если эта задача решена для некоторого набора $(\beta_1, \ldots, \beta_{r-1})$ с оценкой $\sum \beta_i \le \theta^*$, то из (41.21) будем иметь оценку

$$C_{\max}(S) \le D(r) + 2(r-1) + \theta^* - \sum_{q=1}^{r-1} \max \left\{ \frac{1}{m_q}, \frac{1}{m_{q+1}} \right\}. \quad (41.22)$$

При $r \ge 3$ для решения задачи СВ2$(r-1)$ мы можем воспользоваться ее сведением к задаче СВ1$(r-2)$ о суммировании $(r-2)$-мерных проекций $\{b'_j\}$ векторов $\{b_j\}$ на гиперплоскость $H = \{x \in \mathbb{R}^{r-1} \mid x(1) = 0\}$ (см. лемму 14.6 на стр. 114). Поскольку все векторы семейства $X_H = \{b'_1, \ldots, b'_n\}$ содержатся в пересечении $\hat{P}$ полупространств

$$\left\{ P\left(-e_2, \frac{1}{m_r}\right), P\left(e_2, \frac{1}{m_2}\right), P\left(e_2 - e_3, \frac{1}{m_2}\right), P\left(e_3 - e_2, \frac{1}{m_3}\right), \ldots, \right.$$

$$\left. P\left(e_{r-2} - e_{r-1}, \frac{1}{m_{r-2}}\right), P\left(e_{r-1} - e_{r-2}, \frac{1}{m_{r-1}}\right), P\left(e_{r-1}, \frac{1}{m_{r-1}}\right), P\left(-e_{r-1}, \frac{1}{m_r}\right) \right\},$$

то согласно теореме 11.5 (стр. 74) мы можем найти перестановку $\pi = (\pi_1, \ldots, \pi_n)$, обеспечивающую суммирование векторов $\{b'_j\}$ во множестве $(r-3)\hat{P} + P_a$, где $P_a =$

$\operatorname{conv}\left\{0, a - \frac{1}{r-2}\hat{P}\right\}$ для произвольного вектора $a \in \mathbb{R}^{r-2}$. Выбирая вектор $a = 0$, получаем множество P_a, совпадающее с пересечением полупространств

$$\left\{P\left(-e_2, \frac{1}{m_2}\right), P\left(e_2, \frac{1}{m_r}\right), P\left(e_2 - e_3, \frac{1}{m_3}\right), P\left(e_3 - e_2, \frac{1}{m_2}\right), \ldots,\right.$$

$$\left. P\left(e_{r-2} - e_{r-1}, \frac{1}{m_{r-1}}\right), P\left(e_{r-1} - e_{r-2}, \frac{1}{m_{r-2}}\right), P\left(e_{r-1}, \frac{1}{m_r}\right), P\left(-e_{r-1}, \frac{1}{m_{r-1}}\right)\right\},$$

умноженным на $\frac{1}{r-2}$. Таким образом, все частичные суммы $\sum_{j=1}^{k} b'_{\pi_j}$ содержатся во множестве $(r-3)\hat{P} + P_a \subseteq P\left(-e_2, \frac{r-3}{m_r} + \frac{1}{(r-2)m_2}\right) \cap P\left(e_2 - e_3, \frac{r-3}{m_2} + \frac{1}{(r-2)m_3}\right) \cap \cdots \cap P\left(e_{r-2} - e_{r-1}, \frac{r-3}{m_{r-2}} + \frac{1}{(r-2)m_{r-1}}\right) \cap P\left(e_{r-1}, \frac{r-3}{m_{r-1}} + \frac{1}{(r-2)m_r}\right)$, что обеспечивает решение задачи СВ1$(r-2)$ с оценкой

$$\sum_{q=1}^{r-1} \beta_q \le \left(r - 3 + \frac{1}{r-2}\right) \sum_{q=2}^{r} \frac{1}{m_q} \doteq \theta^*.$$

Согласно лемме 14.6, такая же оценка $\sum_{q=1}^{r-1} \beta_q \le \theta^*$ гарантируется при решении задачи СВ2$(r-1)$, что с учетом (41.22) позволяет строить расписание S_1 с оценкой

$$C_{\max}(S_1) \le L_{\max} + \left(2r - 2 + \left(r - 3 + \frac{1}{r-2}\right) \sum_{q=2}^{r} \frac{1}{m_q} - \sum_{q=1}^{r-1} \max\left\{\frac{1}{m_q}, \frac{1}{m_{q+1}}\right\}\right) p_{\max}.$$

Замечаем, что если для решения многопроцессорной задачи flow shop воспользоваться инверсным алгоритмом (в котором цеха проходятся работами в обратном порядке, а затем найденное расписание переворачивается во времени), то с той же трудоемкостью мы получим расписание S_2 с аналогичной оценкой, в которой вместо $\sum_{q=2}^{r} \frac{1}{m_q}$ фигурирует сумма $\sum_{q=1}^{r-1} \frac{1}{m_q}$. Таким образом, для лучшего из двух расписаний $S \doteq S_1 \wedge S_2$ справедлива оценка

$$C_{\max}(S) \le L_{\max} + \left(2r - 2 + \left(r - 3 + \frac{1}{r-2}\right)\left(\sum_{q=2}^{r-1} \frac{1}{m_q} + \min\left\{\frac{1}{m_1}, \frac{1}{m_r}\right\}\right)\right.$$

$$\left. - \sum_{q=1}^{r-1} \max\left\{\frac{1}{m_q}, \frac{1}{m_{q+1}}\right\}\right) p_{\max}. \tag{41.23}$$

При $r \ge 4$ величина в правой части (41.23) достигает своего максимума при $m_q \equiv 1$, $\forall q$. Таким образом, в худшем случае гарантируется оценка

$$C_{\max}(S) \le L_{\max} + 2r - 2 + \left(r - 4 + \frac{1}{r-2}\right)(r-1) = L_{\max} + \left(r^2 - 3r + 3 + \frac{1}{r-2}\right) p_{\max}.$$

При $r = 3$, замечая, что

$$\frac{1}{m_2} + \min\left\{\frac{1}{m_1}, \frac{1}{m_3}\right\} - \max\left\{\frac{1}{m_1}, \frac{1}{m_2}\right\} - \max\left\{\frac{1}{m_2}, \frac{1}{m_3}\right\} = \min\frac{1}{m_q} - \max\frac{1}{m_q} \le 0,$$

приходим к оценке

$$C_{\max}(S) \le L_{\max} + 4p_{\max}.$$

Наконец, при $r = 2$, замечая, что одномерная задача суммирования чисел $\{b_j = d_j(1) - d_j(2) \,|\, j \in \mathbb{N}_n\}$ решается линейным от n алгоритмом с оценкой

$$\sum_{j=1}^{k} b_j \le \theta^* \doteq (D(1) - D(2))^+, \ \forall\ k \in \mathbb{N}_n$$

(шаг 0 в этом случае не требуется), из (41.22) получаем оценку длины расписания:

$$C_{\max}(S) \le L_{\max} + \left(2 - \frac{1}{m_{\min}}\right) p_{\max},$$

что дает оценку

$$C_{\max}(S) \le L_{\max} + p_{\max},$$

если $m_1 = 1$ либо $m_2 = 1$.

Что касается трудоемкости алгоритмов решения нашей задачи при различных значениях r, то нетрудно видеть, что она определяется трудоемкостью алгоритма, применяемого на шаге 1.

При $r \ge 4$, используя на шаге 1 алгоритм из теоремы 11.5, получаем общую оценку трудоемкости $O(n^2r^2)$.

При $r = 3$ одномерная задача СВ1(1) решается с линейной от n трудоемкостью, что с той же трудоемкостью дает решение задачи СВ2(2) и определяет трудоемкость шага 1. Т.о., основная трудоемкость алгоритма сосредоточена в этом случае на шагах 2 и 3 и составляет $O(n \log m)$.

Наконец, при $r = 2$ мы также приходим к оценке трудоемкости $O(n \log m)$.

Теорема 41.1 доказана.

Анализируя оценку (41.23), полученную нами для $r \ge 3$, замечаем, что ее поведение довольно сильно различается в случаях $r = 3$ и $r \ge 4$.

При $r \ge 4$ наихудшим случаем является $m_q \equiv 1$; при этом достигаемая оценка (41.1) совпадает с оценкой (39.2) полученной нами в разделе 39 для обычной задачи flow shop. При возрастании значений $\{m_q\}$ величина в правой части (41.23) имеет тенденцию к уменьшению. В частности, при $m_q \ge r - 3,\ \forall q$, приходим к линейной оценке:

$$C_{\max}(S) < L_{\max} + (3r - 3)p_{\max}.$$

При $r = 3$ нет тенденции уменьшения оценки (41.23) при возрастании величин $\{m_q\}$. Наихудшая оценка $L_{\max} + 4p_{\max}$ достигается, когда все величины m_q равны $m_{\max}$. (Не имеет значения, сколь велико при этом $m_{\max}$.)

Наконец, при $r = 2$ наилучшим для оценки (41.3) является случай, когда одна из величин m_q равна 1. При этом значение второй величины не играет роли. Достигаемая в этом случае оценка $L_{\max} + p_{\max}$ является наилучшей возможной.

Глава 3

Задача о сборочной линии

Формулировка задачи приведена в п.2 раздела 22. Легко показывается, что любое расписание можно преобразовать без увеличения его длины таким образом, чтобы порядок работ на каждой из машин $M_1, \ldots, M_{m-1}$ совпадал с порядком, установленным на машине M_m. Таким образом, хотя бы одно оптимальное расписание содержится в классе перестановочных, что позволяет обойтись лишь рассмотрением последних без потери оптимальности. В выражении (22.17) приведена формула длины перестановочного расписания задачи СЛ для заданной перестановки работ π. Ниже будет доказана теорема, предоставляющая полиномиальное сведение проблемы приближенного решения задачи СЛ к приближенному решению задачи о нахождении нестрогого суммирования векторов в семействе областей.

Теорема 41.2 *Пусть имеется алгоритм $\mathcal{A}$ трудоемкости $\mathcal{T}_{\mathcal{A}}(m-1,n)$, который для некоторого $m \geq 2$ и любого $\hat{s}$-семейства векторов $X \subset \mathbb{R}^{m-1}$ решает задачу* НСВ5$(m-1)$ *с гарантированной оценкой*

$$\theta_5(b_{m-1}) \leq \theta_5^*(m-1).$$

Тогда существует алгоритм $\mathcal{A}'$ трудоемкости $O(\mathcal{T}_{\mathcal{A}}(m-1,n)+mn)$, который для любого примера задачи СЛ *с m машинами и n работами строит перестановочное расписание S_π с оценкой длины:*

$$C_{\max}(S_\pi) \leq L_{\max} + (1+\theta_5^*(m-1))p_{\max}. \tag{41.24}$$

Доказательство. Искомый алгоритм $\mathcal{A}'$ состоит из двух шагов.

На первом шаге с помощью алгоритма $\mathcal{A}$ решаем задачу НСВ5$(m-1)$ для векторов семейства $D = \{d_1, \ldots, d_n\} \subset \mathbb{R}^{m-1}$, определенных согласно (22.11), т.е. находим перестановку $\pi = (\pi_1, \ldots, \pi_n)$, задающую нестрогое суммирование векторов из D в семействе полупространств

$$\{P(e_1, \beta_1), \ldots, P(e_{m-1}, \beta_{m-1})\}$$

с величинами, $\{\beta_i\}$, удовлетворяющими соотношениям

$$\beta_i \leq \theta_5^*(m-1)p_{\max},\ i \in \mathbb{N}_{m-1}.$$

На втором шаге строим перестановочное расписание S_π.

Докажем, что расписание S_π удовлетворяет оценке (41.24).

Считая, что $\pi = (1, 2, \ldots, n)$, преобразуем (22.17) к виду

$$C_{\max}(S_\pi) = \max_{i \in \mathbb{N}_{m-1}} \max_k \left(\sum_{j=1}^{k} p_{ji} - \sum_{j=1}^{k-1} p_{jm} \right) + L_{\max}$$

(с учетом замечания 22.1, стр. 163, и обозначений (22.13), (22.12))

$$= \max_{i \in \mathbb{N}_{m-1}} \max_k P_\pi^k(i, m) + L_{\max} \leq L_{\max} + p_{\max} + \max_i \beta_i \leq L_{\max} + (1 + \theta_5^*(m-1))p_{\max}.$$

Теорема 41.2 доказана. □

В случае трех машин с учетом следствия 14.11 (стр. 116) получаем следующий результат.

Теорема 41.3 *Существует алгоритм трудоемкости $O(n \log n)$, который для всякого примера задачи* СЛ *с тремя машинами и n работами строит перестановочное расписание S_π с неулучшаемой оценкой длины*

$$C_{\max}(S_\pi) \leq L_{\max} + 1.25 p_{\max}.$$

Неулучшаемость оценки показывается на последовательности входов задачи СЛ, в которых n работ имеют вектор длительностей $(1, 1 - \frac{1}{n}, 1 - \frac{1}{2n})$ и еще одна работа — вектор $(0, 1, \frac{1}{2})$.

Теорема 41.3 позволяет найти точный интервал локализации оптимумов 3-машинной задачи СЛ: $[L_{\max}, L_{\max} + 1.25 p_{\max}]$.

Глава 4

Системы с различными маршрутами

Многомаршрутные модели теории расписаний являют собой качественно новую ступень сложности по сравнению с одномаршрутными, поскольку возникает новая проблема, связанная с увязкой различных потоков работ по машинам. Сравнительно просто она решается в простейших случаях — двухмаршрутных моделях для трех машин, рассматриваемых в разделе 43, и в задаче Акерса-Фридман для трех машин (см. раздел 42), решаемой с хорошей точностью жадным алгоритмом, использующим для построения приоритетных порядков работ на машинах алгоритм нестрогого суммирования векторов из раздела 14. Простой алгоритм увязки удается построить и в случае произвольного числа машин для двух встречных маршрутов (см. [40]. В [19] этот алгоритм обобщен на случай p произвольных маршрутов.)

Существенно сложнее проблема увязки маршрутов решается в общей модели job shop, что отражается как на трудоемкости алгоритма, так и на оценке его точности. И та и другая, однако, остаются полиномиальными в методе, описываемом в разделе 44. При этом оценка точности является полиномом, не зависящим от числа работ. Применение алгоритма $\mathcal{A}_{19.1}$ (равномерного распределения камней по кучам с запретами) из раздела 19 позволяет обобщить алгоритм решения задачи job shop на случай нетривиальных множеств допустимых исполнителей каждой операции.

Рассматриваемые в разделах 42, 43 системы характеризуются тем, что каждая работа $J_j \in \mathcal{J}$ имеет не более одной операции на каждой машине $M_i \in \mathcal{M}$. Такую операцию (если она имеется) мы обозначаем o_{ji}, в то время как o_q^j означает q-ю операцию в маршруте прохождения работы J_j по машинам. Вместо формулировки каждой задачи такого типа нам достаточно будет для каждой работы J_j (или для группы работ) указать маршрут ее прохождения по машинам: $(M(o_1^j), M(o_2^j), \ldots, M(o_{r_j}^j))$.

42 Задача Акерса-Фридман для трех машин

Задачей Акерса-Фридман называется частный случай задачи job shop, в котором $r_j \equiv m$ и $\cup_{q=1}^{m} M(o_q^j) = \mathcal{M},\ \forall J_j \in \mathcal{J}$. Таким образом, каждая работа должна обойти все машины согласно заданному для нее маршруту, побывав на каждой машине ровно один раз. В

отличие от системы flow shop, эти маршруты могут быть различны для разных работ. (Как легко понять, в системе Акерса-Фридман возможно $m!$ различных маршрутов.) В этом разделе мы рассмотрим лишь задачу трех машин, которая, как обобщение 3-машинной задачи flow shop, NP-полна в сильном смысле.

Теорема 42.1 *Пусть имеется алгоритм $\mathcal{A}$, который для любого $\hat{s}$-семейства векторов $\{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ с трудоемкостью $\mathcal{T}_{\mathcal{A}}(n)$ решает задачу* НСВ3 *с гарантированной оценкой*

$$\theta_3(b) \leq \theta_3^*.$$

Тогда существует алгоритм $\mathcal{A}_{42.1}$, который для любого примера задачи АФ *с тремя машинами и n работами с трудоемкостью $O(\mathcal{T}_{\mathcal{A}}(n) + n \log n)$ находит расписание S с оценкой длины*

$$C_{\max}(S) \leq L_{\max} + (3 + \theta_3^*) p_{\max}.$$

Доказательство. Алгоритм $\mathcal{A}_{42.1}$ состоит из двух шагов.

Алгоритм $\mathcal{A}_{42.1}$

Шаг 1. Применив к семейству векторов $D \subset \mathbb{R}^2$, определенных согласно (22.11), алгоритм $\mathcal{A}$ решения задачи НСВ3, получим перестановку $\pi = (\pi_1, \ldots, \pi_n)$, задающую нестрогое суммирование векторов D в семействе полуплоскостей

$$\{P(e_1, \beta_1),\ P(e_2, \beta_2),\ P(e_2 - e_1, \beta_3),\ P(-e_1, \beta_4),\ P(-e_2, \beta_5),\ P(e_1 - e_2, \beta_6)\}$$

с оценкой $\theta_3(b) \leq \theta_3^*$. Для удобства выкладок далее считаем, что работы занумерованы согласно перестановке π, иначе говоря, считаем, что $\pi = (1, 2, \ldots, n)$.

Шаг 2. Пусть o_i^j обозначает i-ю операцию работы J_j. Определим приоритет $\succ_p$ на множестве операций:

1. $o_1^j \succ_p o_3^k,\ o_2^j \succ_p o_3^k,\ \forall J_j, J_k \in \mathcal{J}$, т.е. любая первая или вторая операция какой-либо работы приоритетнее любой третьей;

2. $o_i^j \succ_p o_\nu^k,\ \forall j < k,\ i, \nu \in \{1, 2\}$, т.е. на множестве первых и вторых операций приоритет определяется перестановкой π номеров работ;

3. $o_3^j \succ_p o_3^k,\ \forall j < k$.

Расписание S строим жадным алгоритмом, который

- не позволяет машине простаивать, если есть операции, готовые на ней выполняться;

- выбирает очередную операцию на освободившуюся машину из очереди ждущих операций согласно определенному выше приоритетному порядку $\succ_p$.

Алгоритм $\mathcal{A}_{42.1}$ описан.

Для каждой машины $M_i,\ i = 1, 2, 3$ очередь ждущих операций организуем в виде "кучи" [132], т.е. такого двоичного дерева T^i с вершинами-операциями, в котором приоритет операций убывает по каждой из цепочек, идущих из корня в листья дерева T^i, и длина каждой такой цепочки не превосходит $O(\log n)$. Тогда постановка каждой операции в очередь ждущих и удаление наиболее приоритетной операции из очереди выполняются за $O(\log n)$ элементарных действий. Это обеспечивает реализацию жадного алгоритма с трудоемкостью $O(n \log n)$, и тем самым позволяет получить требуемую в теореме оценку трудоемкости алгоритма. Докажем оценку точности.

Расписание работы произвольной машины M_i можно представить как чередование интервалов работы $[s^i_\nu, f^i_\nu)$ и интервалов простоя $[f^i_\nu, s^i_{\nu+1}),\ \nu = 1, 2, \ldots$

Пусть $o_{ji} = o^j_1$ — первая операция работы J_j. Так как она стоит в очереди на машину M_i с момента $t = 0$, она не может пропустить вперед какую-либо менее приоритетную операцию. Поэтому для момента ее окончания (c_{ji}) мы можем выписать оценку:

$$c_{ji} \le \sum_{q=1}^{j} p_{qi} \doteq P^j_\pi(i). \tag{42.1}$$

Пусть $o_{ji} = o^j_2$ — вторая операция работы J_j, и выполняется на машине M_i в интервале $[s_{ji}, c_{ji}) \subseteq [s^i_\nu, f^i_\nu)$. Пусть $o_{j_1 i}, o_{j_2 i}, \ldots, o_{j_\nu i}$ — такая максимальная цепочка операций, последовательно выполняемых на машине M_i в интервале $[s^i_\nu, c_{ji})$, что $o_{j_\nu i} = o_{ji}$ и $o_{j_1 i} \succ_p o_{j_2 i} \succ_p \cdots \succ_p o_{ji}$. Операция $o_{j_1 i}$ либо выполняется первой в интервале $[s^i_\nu, c_{ji})$, т.е. $s_{j_1 i} = s^i_\nu$, либо ей предшествует менее приоритетная операция. В обоих случаях между моментом постановки операции $o_{j_1 i}$ в очередь на машину M_i и моментом $s_{j_1 i}$ прошло менее $p_{\max} = 1$ единиц времени.

Из определения приоритета $\succ_p$ заключаем, что $o_{j_1 i} \in \{o^{j_1}_1, o^{j_1}_2\}$ и

$$j_1 < j_2 < \cdots < j_\nu = j. \tag{42.2}$$

Пусть $o_{j_1 i} = o^{j_1}_2$, т.е. $o_{j_1 i}$ — вторая операция работы J_{j_1}, первая операция которой выполняется на машине $M_{i_1},\ i_1 \ne i$. Тогда из (42.1) имеем оценку

$$s_{j_1 i} < c_{j_1 i_1} + 1 \le P^{j_1}_\pi(i_1) + 1 = P^{j_1 - 1}_\pi(i) + 1 + (P^{j_1}_\pi(i_1) - P^{j_1 - 1}_\pi(i)).$$

Отсюда с учетом (42.2) и (22.13) получаем верхнюю оценку момента окончания операции o_{ji}:

$$c_{ji} = s_{j_1 i} + \sum_{k=1}^{\nu} p_{j_k i} < P^j_\pi(i) + 1 + \max_k P^k_\pi(i_1, i). \tag{42.3}$$

В этой оценке мы отошли от ее происхождения от работы J_{j_1}, перейдя к $\max_k$. Поэтому и о машине M_{i_1} мы можем сказать лишь то, что $i_1 \ne i$.

Поскольку величина $\max_k$ в правой части (42.3) неотрицательна, то оценка (42.3) справедлива и в случае $o_{j_1 i} = o^{j_1}_1$. Таким образом, (42.3) является оценкой момента

окончания второй операции любой работы. Это позволяет ввести верхнюю оценку на время $\hat{t}_{ji_3}$ постановки работы J_j в очередь на машину M_{i_3} для выполнения третьей операции работы J_j. Предполагая, что вторая операция этой работы выполняется на машине M_{i_2}, получаем:

$$\hat{t}_{ji_3} = c_{ji_2} < P_\pi^j(i_2) + 1 + \max_k P_\pi^k(i_1, i_2)$$

$$= P_\pi^{j-1}(i_3) + 1 + (P_\pi^j(i_2) - P_\pi^{j-1}(i_3)) + \max_k P_\pi^k(i_1, i_2) \le P_\pi^{j-1}(i_3)) + 1 + \beta, \qquad (42.4)$$

где

$$\beta = \max_{\{i_1, i_2, i_3 \,|\, i_2 \neq i_1,\, i_2 \neq i_3\}} \{\max_k P_\pi^k(i_1, i_2) + \max_k P_\pi^k(i_2, i_3)\}.$$

Сравнивая (42.4) с (42.3), нетрудно убедиться, что момент постановки второй операции любой работы в очередь на свою машину также удовлетворяет оценке (42.4), не говоря уже о первых операциях, для которых $\hat{t}_{ji} = 0$. Таким образом, момент постановки в очередь на машину M_i любой операции o_{ji} наступает не позже

$$\hat{t}_{ji} < P_\pi^{j-1}(i) + 1 + \beta. \qquad (42.5)$$

Из определения векторов $\{a_i \,|\, i \in N_6\}$ в задаче НСВ3, для любых двух индексов $i_1, i_2 \in \{1, 2, 3\}$ имеем $e_{i_1} - e_{i_2} = a_i$, где соответствие между индексами i_1, i_2 и i устанавливается таблицей 42.1 (в предположении $e_3 = 0$).

i	1	2	3	4	5	6	1
i_1	1	2	2	3	3	1	1
i_2	3	3	1	1	2	2	3

Таблица 42.1:

Из таблицы видно, что если $i_1, i_2, i_3 \in \{1, 2, 3\}$, $i_1 \neq i_2$, $i_2 \neq i_3$, и $e_{i_1} - e_{i_2} = a_{i'}$, $e_{i_2} - e_{i_3} = a_{i''}$, то $i' \neq i''$, $i' \neq s(i'')$, $i'' \neq s(i')$ имея в виду циклический порядок на множестве N_6, определенный в разделе 14. (В противном случае во второй и третьей строках таблицы в одном столбце или в соседних столбцах по диагонали оказались бы два совпадающих элемента i_2. Однако легко заметить, что это невозможно, т.к. каждая пара элементов (i', i') во второй строке таблицы граничит по диагонали с парами $(i'', i''), (i''', i''')$ из третьей строки, где i', i'', i''' — попарно различны.) Таким образом, с учетом замечания 22.1 (стр. 163) получаем оценку:

$$\beta \le 2 + \max(\beta_{i(i_1,i_2)} + \beta_{i(i_2,i_3)}) \le 2 + \max_{(i',i'') \in I} (\beta_{i'} + \beta_{i''}) = 2 + \theta_3(b) \le 2 + \theta_3^*. \qquad (42.6)$$

Пусть M_i — произвольная машина. Покажем, что величина ее простоя (D_i) не превосходит $1 + \beta$. Если момент t' начала ее последнего интервала работы наступает раньше $1 + \beta$, то очевидно, $D_i < 1 + \beta$. Пусть $t' \ge 1 + \beta$ и пусть j^* — максимальное такое j, что

$$P_\pi^{j-1}(i) + 1 + \beta \le t'. \qquad (42.7)$$

Тогда

$$P_{\pi}^{j^*}(i) + 1 + \beta > t'. \tag{42.8}$$

Из (42.5) и (42.7) заключаем, что все операции $\{o_{1i}, \ldots, o_{j^*i}\}$ становятся в очередь на машину M_i раньше момента t'. Так как моменту t' непосредственно предшествует простой на машине M_i, то все операции, вставшие в очередь на машину M_i строго до момента t', успели выполниться до момента t', согласно первому правилу жадного алгоритма. Отсюда $j^* < n$, и простой машины M_i до момента t' (а значит, и до момента C_i окончания ее работы) с учетом (42.8) оценивается:

$$D_i \leq t' - P_{\pi}^{j^*}(i) < 1 + \beta.$$

С учетом (42.6) получаем оценку

$$C_i = L_i + D_i \leq L_{\max} + 1 + \beta \leq L_{\max} + 3 + \theta_3^*,$$

что дает требуемую оценку длины расписания.

Теорема 42.1 доказана.

Из теоремы 42.1 и следствия 14.8 вытекают следующие результаты.

Теорема 42.2 *Существует алгоритм трудоемкости $O(n \log n)$, который для любого примера задачи Акерса-Фридман с тремя машинами и n работами строит расписание S с оценкой длины*

$$C_{\max}(S) < L_{\max} + 5.5 p_{\max}.$$

Следствие 42.3 *Для любого примера задачи Акерса-Фридман с m машинами его оптимум находится в интервале $[L_{\max}, L_{\max} + \mu_{AF}(m) p_{\max}]$, где значение функции $\mu_{AF}(m)$ при $m = 3$ удовлетворяет оценкам*

$$3 \leq \mu_{AF}(3) \leq 5.5.$$

43 Двухмаршрутные задачи трех машин

1. Занумеровав машины согласно одному из маршрутов, мы всегда можем считать, что первым маршрутом является $(1, 2, 3)$. Для второго маршрута имеется три принципиально различных варианта: $(3, 2, 1), (2, 1, 3)$ и $(2, 3, 1)$. В соответствии с выбором второго маршрута, пару маршрутов будем называть, соответственно:

- встречными;
- перекрещивающимися;
- циклически замкнутыми,

а соответствующие задачи обозначать R321, R213 и R231. Остальные два варианта второго маршрута ($(1, 3, 2)$ и $(3, 1, 2)$) сводятся к одному из трех предыдущих либо перенумерацией машин по второму маршруту, либо запуском обоих маршрутов "наоборот", с обращением вспять оси времени (стандартный прием в теории расписаний).

В каждом из трех перечисленных выше случаев будет построен свой алгоритм увязки маршрутов, учитывающий их специфику. При этом получаемые оценки точности будут различны, что говорит либо о недостатках предложенных алгоритмов, либо о "врожденной" сложности самих моделей, различающейся для различных пар маршрутов. Лучше всего поддаются увязке встречные маршруты, для которых построен алгоритм с наилучшей возможной оценкой точности вида (22.4). Вопрос о точности оценок для двух других двухмаршрутных задач остается открытым.

В каждой из трех задач множество работ $\mathcal{J}$ разбивается на два подмножества: $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2$, где $\mathcal{J}_i = \{J_j \,|\, j \in \mathcal{N}_i\},\ i = 1, 2;\ \mathcal{N}_1 = \{1, \ldots, n'\}$ — множество номеров работ с маршрутом $(1, 2, 3)$ (будем называть его "прямым"), а $\mathcal{N}_2 = \{n' + 1, \ldots, n\}$ — множество номеров работ второго маршрута. Операции работы $J_j \in \mathcal{J}_1$ на машинах M_1, M_2 и M_3 будем обозначать o_{j1}, o_{j2}, o_{j3}, а их длительности — a_j, b_j, c_j, соответственно. Для работы $J_j \in \mathcal{J}_2$ соответствующие обозначения: $o'_{j1}, o'_{j2}, o'_{j3}$ и a'_j, b'_j, c'_j. $[X]$ будет обозначать суммарную длительность операций из множества X. Для загрузки машин работами каждого из маршрутов будем использовать обозначения:

$$A = \sum_j a_j,\ B = \sum_j b_j,\ C = \sum_j c_j, A' = \sum_j a'_j,\ B' = \sum_j b'_j,\ C' = \sum_j c'_j.$$

Будем считать выполненными соотношения $p_{\max} = 1$ и (22.9), или

$$A + A' = B + B' = C + C' = L_{\max}. \tag{43.1}$$

2. Задача R321.

Вспомогательную роль при доказательстве главного результата будет иметь *двухстадийная задача встречных маршрутов* (ДЗВМ). От задачи R321 она отличается тем, что третья операция каждой работы (на машине M_3— для прямого маршрута, и на M_1 — для встречного) имеет нулевую длительность. Несмотря на сравнительную простоту ДЗВМ оказывается верной следующая

Теорема 43.1 *Задача* ДЗВМ NP*-трудна в сильном смысле.*

Доказательство. Пусть $l' = \max\{A, C'\}$ и задано расписание S, в котором работы на машине M_1 выполняются непрерывно в интервале $[0, A]$ в порядке $1, 2, \ldots, n'$, а на M_3 — непрерывно в интервале $[0, C']$ в порядке $n'+1, \ldots, n$. Расписание второй машины представляет собой чередование интервалов простоя и интервалов работы.

Для $t \in [0, l']$ определим функцию

$$f(t) = t + \sum_{j=k(t)+1}^{n'} b_j + \sum_{j=k'(t)+1}^{n} b'_j, \tag{43.2}$$

где $k(t) \in \mathcal{N}_1$ и $k'(t) \in \mathcal{N}_2$ — максимальные такие k и k', что

$$\sum_{j=1}^{k} a_j < t,\ \sum_{j=n'+1}^{k'} c'_j < t.$$

Поскольку работы $J_j,\ j \in \{k(t)+1,\dots,n'\} \cup \{k'(t)+1,\dots,n\}$, приходят на машину M_2 не раньше t, то для любого $t \in [0, l']$ выполняется

$$C_{\max}(S) \geq f(t). \tag{43.3}$$

В то же время, взяв в качестве t начало последнего интервала работы машины M_2, убеждаемся, что в (43.3) выполняется равенство. Таким образом, нами доказано соотношение

$$C_{\max}(S) = \max_{t\in[0,l']} f(t). \tag{43.4}$$

Докажем NP-полноту следующей задачи распознавания: для заданного примера задачи ДЗВМ и для заданного x определить, существует ли в задаче ДЗВМ расписание S длины не более x. К задаче ДЗВМ на распознавание сведем задачу 3-РАЗБИЕНИЕ ([17, стр. 283]). Пусть заданы множество $N = \{1, 2, \dots, 3n_0\}$, натуральное число E и целые числа $\{e_i \,|\, i \in N\}$ такие, что $E/4 < e_i < E/2,\ \sum e_i = n_0 E$. Рассмотрим задачу ДЗВМ, в которой для $n' = n_0$ работ прямого маршрута и $n'' = 4n_0 + 1$ работ встречного маршрута $(n' + n'' = n)$ заданы следующие величины длительностей операций:

$$a_i = 6E,\ b_i = a_i/2,\ i = 1,\dots,n';\ c'_i = e_i,\ i = n_0+1,\dots,4n_0;$$

$$c'_i = 5E,\ i = 4n_0+1,\dots,5n_0-1;\ c'_{5n_0} = 3E;\ c'_{5n_0+1} = 2E;$$

$$b'_i = c'_i/2,\ i = n_0+1,\dots,n.$$

Для ДЗВМ выполнено:

$$A = C' = B + B' = L_{\max}. \tag{43.5}$$

Пусть $S_{\pi,\pi'}$ — расписание в котором работы прямого маршрута выполняются согласно перестановке $\pi = (\pi_1,\dots,\pi_{n'})$ чисел из $\mathcal{N}_1$, а встречного — согласно перестановке $\pi' = (\pi'_{n'+1},\dots,\pi'_n)$ чисел из $\mathcal{N}_2$. Обозначим $x^i_\pi = \sum_{j=1}^{i} a_{\pi_j},\ z^i_{\pi'} = \sum_{j=n'+1}^{i} c'_{\pi'_j}$. Тогда из (43.2), (43.4) и (43.5) для определенного выше примера задачи ДЗВМ получаем:

$$C_{\max}(S_{\pi,\pi'}) = \max_t \left\{ (t - x^{k(t)}_\pi + t - z^{k'(t)}_{\pi'})/2 + x^{k(t)}_\pi/2 + z^{k'(t)}_{\pi'}/2 + \sum_{j=k(t)+1}^{n'} b_{\pi_j} \right.$$

$$\left. + \sum_{j=k'(t)+1}^{n} b'_{\pi'_j} \right\} = \max_t \left\{ (\rho_\pi(t) + \rho'_{\pi'}(t))/2 + \sum_{j=1}^{k(t)} a_{\pi_j}/2 + \sum_{j=n'+1}^{k'(t)} c'_{\pi'_j}/2 \right.$$

$$\left. + \sum_{j=k(t)+1}^{n'} a_{\pi_j}/2 + \sum_{j=k'(t)+1}^{n} c'_{\pi'_j}/2 \right\} = L_{\max} + \max_{t\leq L_{\max}} (\rho_\pi(t) + \rho'_{\pi'}(t))/2, \tag{43.6}$$

где $\rho_\pi(t) = t - x^{k(t)}_\pi$ — расстояние от t до ближайшей слева точки x^k_π, а $\rho'_{\pi'}(t) = t - z^{k'(t)}_{\pi'}$ — расстояние от t до ближайшей слева точки $z^k_{\pi'}$. Покажем, что

$$C_{\max}(S_{\pi,\pi'}) \geq L_{\max} + 4E,\ \forall \pi, \pi'. \tag{43.7}$$

Поскольку все операции $\{o_{i1}\}$ имеют одинаковую длину, то множество точек $\{x_\pi^i\}$ не зависит от перестановки π. Рассмотрим одну из операций o'_{i3} длины $5E$, и пусть она выполняется в интервале $[y,z],\ z=y+5E$. Если $[y,z]$ целиком содержится в одном из интервалов $[x_\pi^i, x_\pi^{i+1}]$, то $\rho_\pi(z)\ge 5E,\ \rho'_{\pi'}(z)=5E$, откуда $C_{\max}(S_{\pi,\pi'})\ge L_{\max}+5E$. Если же $x_\pi^{i+1}=y+\delta$, где $\delta\in(0,5E)$, то

$$\rho_\pi(x_\pi^{i+1})+\rho'_{\pi'}(x_\pi^{i+1})=6E+\delta,$$
$$\rho_\pi(z)+\rho'_{\pi'}(z)=5E-\delta+5E=10E-\delta,$$

и максимум из этих двух значений всегда не меньше $8E$, что с учетом (43.6) доказывает оценку (43.7).

Покажем, что исходная задача 3-РАЗБИЕНИЕ имеет ответ "да" если и только если в задаче ДЗВМ существует расписание длины $C_{\max}(S_{\pi,\pi'})=L_{\max}+4E$.

Пусть $S_{\pi,\pi'}$ — расписание длины $L_{\max}+4E$. Как мы только что убедились, в таком расписании каждая операция o'_{j3} длины $5E$ с необходимостью выполняется в интервале $[x_\pi^{i-1}+4E, x_\pi^i+3E],\ i\in\mathbb{N}_{n_0-1}$. Поскольку промежутки между этими интервалами равны E, то в них не могут быть выполнены операции $o'_{5n_0,3}$ и $o'_{5n_0+1,3}$. Кроме того, операция $o'_{5n_0,3}$ не может выполняться в интервале $[x_\pi^{n_0-1}+3E, x_\pi^{n_0}]$, так как в этом случае $\rho_\pi(x_\pi^{n_0})+\rho'_{\pi'}(x_\pi^{n_0})=9E$. Таким образом, операция $o'_{5n_0,3}$ выполняется внутри интервала $[0,4E]$, а операция $o'_{5n_0+1,3}$ — внутри $[x_\pi^{n_0-1}+3E, x_\pi^{n_0}]$. Поскольку в интервале $[0,L_{\max}]$ машина M_3 работает без простоев, это означает, что для множества $X\doteq\{o'_{i3}\,|\,i=n_0+1,\dots,4n_0\}\doteq\bar{C}$ остальных операций машины M_3 существует такое его разбиение на n_0 подмножеств $X=X_1\cup\dots\cup X_{n_0}$ (где X_i — множество операций из X, выполняемых в интервале $[x_\pi^{i-1}, x_\pi^i]$), что сумма длительностей операций каждого из множеств X_i равна E. Это дает ответ "да" в исходной задаче 3-РАЗБИЕНИЕ.

Для доказательства обратного утверждения достаточно показать, как по заданному 3-разбиению строится расписание длины $L_{\max}+4E$ в задаче ДЗВМ, — а это мы делать уже умеем.

Поскольку очевидна принадлежность задачи ДЗВМ на распознавание классу NP, то тем самым доказана NP-полнота этой задачи в сильном смысле.

Теорема 43.1 доказана.

Теорема 43.2 *Для двухстадийной задачи встречных маршрутов с трудоемкостью $O(n\log n)$ может быть построено расписание S с оценкой длины*

$$C_{\max}(S)\le L_{\max}+p_{\max}. \tag{43.8}$$

Доказательство. При доказательстве мы будем использовать следующее известное

Утверждение 43.3 *Если $x_i\ge 0,\ y_i>0,\ i\in\mathbb{N}_n$, и*

$$\frac{x_1}{y_1}\le\frac{x_2}{y_2}\le\dots\le\frac{x_n}{y_n},\ \textit{то}\ \frac{x_1}{y_1}\le\frac{x_1+\dots+x_n}{y_1+\dots+y_n}\le\frac{x_n}{y_n}. \qquad \square$$

Искомое расписание S строится жадным алгоритмом, использующим на каждой машине приоритетный порядок работ $(1,2,\dots,n)$. При этом работы $J_j,\ j\in\mathcal{N}_1=\{1,\dots,n'\}$

занумерованы по неубывания отношения a_j/b_j (работы с $b_j = 0$ идут последними), а работы $J_j,\ j \in \mathcal{N}_2 = \{n'+1, \ldots, n\}$ — по неубывания отношения c'_j/b'_j (работы с $b'_j = 0$ идут последними).

Очевидно, искомая трудоемкость $O(n \log n)$ алгоритма складывается из трудоемкостей нумерации работ прямого и встречного маршрутов. Докажем для полученного расписания оценку (43.8).

Ясно, что машины M_1 и M_3 закончат свою работу в моменты L_1 и L_3, а машина M_2 — в момент $T = C_{\max}(S)$. Для определенности будем считать, что $L_3 \le L_1$. Через $D_i(a, b)$ обозначим величину простоя машины M_i в интервале $[a, b]$.

Если $T \le L_1 + 1$, то соотношение (43.8) выполнено. Пусть

$$T - L_1 > 1. \tag{43.9}$$

Пусть t — начало последнего интервала работы машины M_2. Ясно, что $t \le L_1$. При $t \le 1$ свойство (43.8) выполнено. Далее считаем, что $t > 1$. Рассмотрим два случая:

$$\text{A) } t \le L_3;\ \text{B) } t > L_3.$$

Случай A. Так как моменту t предшествует простой на машине M_2, то это означает, что все работы, пришедшие на машину M_2 строго до момента t, успели выполниться на ней также до момента t. Пусть для прямого маршрута это работы с номерами $1, \ldots, k$, а для встречного — с номерами $n'+1, \ldots, k'$. Обозначим

$$A_1 = \sum_{j=1}^{k} a_j,\ A_2 = A - A_1,\ B_1 = \sum_{j=1}^{k} b_j,\ B_2 = B - B_1,$$

$$C_1 = \sum_{j=n'+1}^{k'} c'_j,\ C_2 = C' - C_1,\ B'_1 = \sum_{j=n'+1}^{k'} b'_j,\ B'_2 = B' - B'_1.$$

Так как мы рассматриваем случай A, то $A_2 > 0,\ C_2 > 0,\ A_1 \ge t-1,\ C_1 \ge t-1$. Согласно выбранному нами правилу нумерации работ и по утверждению 43.3 имеем $B_1/A_1 \ge B_2/A_2,\ B'_1/C_1 \ge B'_2/C_2$, откуда, используя (43.9), получаем

$$\begin{aligned} D_2(0,T) &= D_2(0,t) = t - B_1 - B'_1 \le t - A_1B_2/A_2 - C_1B'_2/C_2 \\ &\le t - (t-1)B_2/(L_1 - t + 1) - (t-1)B'_2/(L_3 - t + 1) \\ &\le t - (t-1)(B_2 + B'_2)/(L_1 - t + 1) \\ &= t - (t-1)(T-t)/(L_1 - t + 1) < t - (t-1) = 1. \end{aligned}$$

Случай B. В этом случае $C_2 = B'_2 = 0$. Аналогично получаем:

$$\begin{aligned} D_2(0,T) &= t - B_1 - B'_1 \le t - B_1 \le t - A_1B_2/A_2 \\ &\le t - (t-1)(T-t)/(L_1 - t + 1) < 1. \end{aligned}$$

Теорема 43.2 доказана.

Эта теорема дает верхнюю оценку значения $\tilde{\mu}_{\text{ДЗВМ}}(3)$, которая достигается на примере с одной деталью прямого маршрута и одной — встречного при единичных длительностях операций. Таким образом, имеем

Следствие 43.4 $\tilde{\mu}_{\text{ДЗВМ}}(3) = 1$ □

Теорема 43.5 *Для трехмашинной задачи встречных маршрутов с трудоемкостью $O(n \log n)$ может быть построено расписание S с оценкой длины*

$$C_{\max}(S) \leq L_{\max} + 3p_{\max}. \tag{43.10}$$

Доказательство. Алгоритм $\mathcal{A}_{43.1}$ построения искомого расписания состоит из четырех этапов. В результате работы алгоритма все множество операций $\mathcal{O}$ будет разбито на три непересекающихся подмножества $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$. Для них будут построены индивидуальные расписания S_1, S_2, S_3, стыковка которых даст необходимый результат.

Обозначим: F_i^k — множество операций из $\mathcal{O}_k$, выполняемых на машине M_i; B_i^k и E_i^k — моменты начала и окончания выполнения операций из F_i^k в расписании S_k. Без ограничения общности можем считать выполненными соотношения (43.1) и $C' \leq A$.

Алгоритм $\mathcal{A}_{43.1}$

Этап 1 (выделение множества $\mathcal{O}_1$ и построение расписания S_1).

Рассмотрим множество всех первых и вторых операций обоих маршрутов. Для полученной ДЗВМ построим расписание $\hat{S}_1$ согласно алгоритму теоремы 43.2. При этом машина M_3, работая без простоев, закончит работу к моменту $E_3^1 = C'$. В это время машина M_1 еще работает.

Включаем в F_1^1 и F_3^1 все операции машин M_1 и M_3, успевающие выполниться в расписании $\hat{S}_1$ до момента E_3^1. Положив $E_1^1 = [F_1^1]$, будем иметь $E_1^1 \leq E_3^1$.

Включаем в F_2^1 вторые операции работ J_j, удовлетворяющих двум требованиям:

- первая операция работы J_j (o_{j1} либо o'_{j3}) включена во множество $\mathcal{O}_1$;
- вторая операция заканчивается в расписании $\hat{S}_1$ не позднее момента $E_1^1 + 2$.

Все операции из $\mathcal{O}_1$ выполняем согласно расписанию $\hat{S}_1$. (Полученное расписание обозначаем через S_1.)

Очевидно, что выполняются соотношения

$$E_1^1 \leq E_3^1 \leq E_2^1 \leq E_1^1 + 2. \tag{43.11}$$

Кроме того, имеем

$$[F_2^1] \geq E_1^1. \tag{43.12}$$

Действительно, в случае $E_2^1 \geq E_1^1 + 1$ неравенство (43.12) следует из соотношений $D_2(0, E_2^1) \leq D_2(0, T) = C_{\max}(\hat{S}_1) - B - B'$, равенства $L_{\max} = B + B'$ и теоремы 43.2.

Если же $E_2^1 < E_1^1 + 1$, то это возможно лишь в том случае, если в F_2^1 вошли вторые операции *всех* работ, первые операции которых также входят в $\mathcal{O}_1$. Но тогда по правилу упорядочения работ в расписании $\hat{S}_1$ имеем

$$[F_2^1] \geq E_1^1 B/A + B' = E_1^1 \left(\frac{B}{A} + \frac{B'}{E_1^1} \right) \geq E_1^1 \left(\frac{B}{B+B'} + \frac{B'}{B+B'} \right) = E_1^1.$$

Из (43.11) и (43.12) получаем

$$[F_i^1] \geq E_1^1, \ \ i = 1, 2, 3. \tag{43.13}$$

Этап 2 (выделение множества $\mathcal{O}_2$ и построение расписания S_2).

На множестве операций $\mathcal{O} \setminus (\mathcal{O}_1 \cup \{o'_{j1} \mid j = n'+1, \ldots, n\})$ рассмотрим одномаршрутную задачу Беллмана-Джонсона с тремя машинами и n работами. (Будем называть ее задачей БД.) Это возможно, так как каждая работа встречного маршрута представлена не более чем одной ненулевой операцией (на машине M_2). Недостающие операции работ в задаче БД будем называть фиктивными операциями нулевой длины. Обозначим через $L_{\text{БД}}$ максимальную из загрузок трех машин в задаче БД и выровняем загрузку остальных машин до уровня $L_{\text{БД}}$, удлиннив какие-то из их операций не более чем до 1.

Согласно лемме 39.7 (при $b = 2$), для полученной одномаршрутной задачи за $O(n \log n)$ операций построим перестановочное расписание S_π ($\pi = (\pi_1, \ldots, \pi_n)$), в котором машина M_1 работает непрерывно в интервале $[0, L_{\text{БД}}]$, машина M_2 — в интервале $[2, 2 + L_{\text{БД}}]$, машина M_3 — в интервале $[3, 3 + L_{\text{БД}}]$, и при этом выполняется

$$\sum_{j=1}^{k} p_{\pi_j 2} \leq \sum_{j=1}^{k} p_{\pi_j 3}, \ \ k \in \mathbb{N}_n. \tag{43.14}$$

Удалим из расписания S_π дополнительные длительности операций. При этом для каждой удлиненной операции машины M_2 считаем дополнительным ее начальный отрезок (чтобы после его удаления момент окончания операции остался прежним). Удалим также все фиктивные (вновь нулевые) операции. На машинах M_1 и M_3 устраним образовавшиеся в результате удаления простои, сдвинув оставшиеся операции влево — на машине M_1 и вправо — на машине M_3 при сохранении длины расписания ($L_{\text{БД}} + 3$). Полученное расписание обозначим $\hat{S}_2$.

Включаем в F_1^2 и F_2^2 все операции машин M_1 и M_3, соответственно, начинающие выполняться в расписании $\hat{S}_2$ до момента $E_1^2 = A - [F_1^1]$. Эти же операции успевают и закончиться до момента E_1^2 (кроме, быть может, последней из F_2^2 — операции работы J_{π_k}). Таким образом,

$$E_2^2 < E_1^2 + 1. \tag{43.15}$$

Включаем в F_3^2 операции $\{o_{j3}\}$ работ прямого маршрута с номерами $j \in \{\pi_1, \ldots, \pi_k\}$.

Все операции из $\mathcal{O}_2$ выполняем согласно расписанию $\hat{S}_2$, обозначая полученное расписание S_2. По построению расписания S_2,

$$B_2^2 \geq 2. \tag{43.16}$$

Этап 3 (построение расписания S_3 для множества операций $\mathcal{O}_3$).

В F_1^3, F_2^3 и F_3^3 включаем оставшиеся операции машин M_1, M_2 и M_3. Поскольку $\mathcal{O}_3$ не содержит первых операций исходных работ, то мы решаем на $\mathcal{O}_3$ задачу, обратную к ДЗВМ. Для этого, назвав третьи операции "первыми", решаем задачу ДЗВМ, а затем прогоняем полученное расписание наоборот. Для построенного расписания S_3 справедливы соотношения:

$$E_2^3 \le E_1^3 = E_3^3 = C_{\max}(S_3),\ B_2^3 = 0,$$

$$B_1^3 = C_{\max}(S_3) - [F_1^3],\ B_3^3 = C_{\max}(S_3) - [F_3^3],$$

причем машины M_1 и M_3 работают без простоев в интервалах $[B_1^3, E_1^3]$ и $[B_3^3, E_3^3]$.

Этап 4 (построение расписания S).

Пристыкуем расписание S_2 к S_1 справа. Из свойств (43.11) и (43.16) следует, что стыковка произойдет на машине M_1. Таким образом, моменты начала всех операций из $\mathcal{O}_2$ увеличатся на E_1^1. К полученному расписанию (обозначим его $S_1 \oplus S_2$) пристыкуем справа расписание S_3, увеличив моменты начала всех операций из $\mathcal{O}_3$ на величину

$$\Delta \doteq E_1^1 + \max\{E_2^2, E_3^2 - B_3^3\}. \tag{43.17}$$

При этом возможны два варианта стыковки: либо на машине M_2, либо на машине M_3 (см. рис. 43.1),— в зависимости от того, на какой из двух величин достигается максимум в правой части (43.17). Эти варианты будем обозначать, соответственно, B и C.

Алгоритм $\mathcal{A}_{43.1}$ описан.

В допустимости полученного расписания $S = S_1 \oplus S_2 \oplus S_3$ нетрудно убедиться по его построению. Столь же очевидна декларируемая в теореме оценка трудоемкости алгоритма. Остается привести доказательство оценки (43.10).

Вариант C (стыковка S_2 и S_3 на машине M_3).

Так как в этом случае имеем $C_{\max}(S) = E_1^1 + C_{\max}(\hat{S}_2) = E_1^1 + L_{\text{БД}} + 3$, то из (43.13) получаем оценку (43.10).

Вариант B (стыковка S_2 и S_3 на машине M_2).

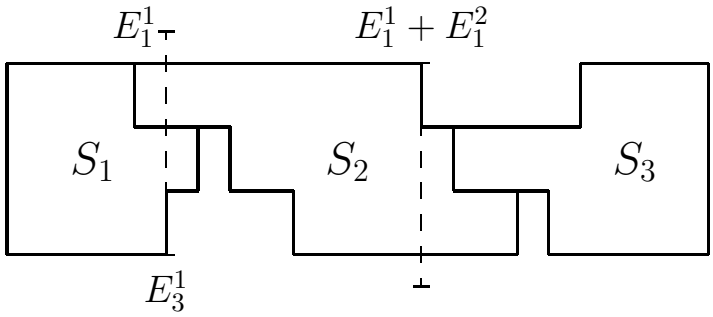
Рассмотрим три случая в зависимости от того, нагрузка какой из трех машин $M_i,\ i = 1, 2, 3$, в расписании S_3 является максимальной.

Случай 1. По теореме 43.2 имеем $B_1^3 \le 1$. Отсюда с учетом (43.15) следует $D_1(0, T) \le 2$.

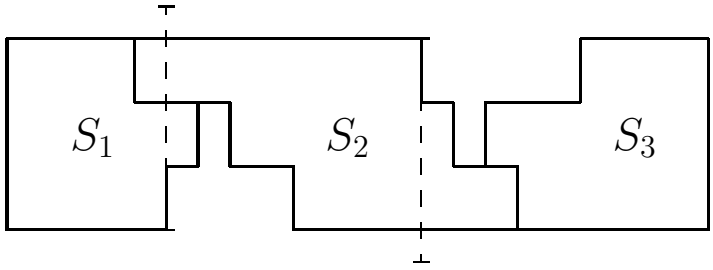
Случай 2. По теореме 43.2 имеем $D_2(E_1^1 + E_2^2, C_{\max}(S)) \le 1$, откуда по построению расписаний $\hat{S}_2$ и S_π с использованием (43.13) получаем:

$$C_{\max}(S) \le E_1^1 + E_2^2 + [F_2^3] + 1 \le E_1^1 + C_{\max}(S_\pi) = E_1^1 + L_{\text{БД}} + 3 \le L_{\max} + 3.$$

Случай 3. В силу (43.14) и по построению расписания S_2 имеем $E_3^2 - E_2^2 \ge 1$. По теореме 43.2, $B_3^3 \le 1$. Наконец, поскольку рассматривается вариант B, то $E_2^2 \ge E_3^2 - B_3^3$.



B) Стыковка на машине M_2.



C) Стыковка на машине M_3.

Рис. 43.1: Два варианта схемы расписания S.

Из этих трех неравенств получаем $E_3^2 - B_3^3 \le E_2^2 \le E_3^2 - 1 \le E_3^2 - B_3^3$, откуда $E_2^2 = E_3^2 - B_3^3$. Но этот случай входит в уже рассмотренный вариант C.

Теорема 43.5 доказана.

Из теоремы 43.5 и соотношения $\tilde{\mu}_{\mathrm{R321}}(3) \ge \tilde{\mu}_{\mathrm{FS}}(3) = 3$ вытекает

Следствие 43.6 $\tilde{\mu}_{\mathrm{R321}}(3) = 3$ □

Приведем без доказательства результат, полученный в [40] для задачи встречных маршрутов с произвольным числом машин.

Теорема 43.7 *Для задачи встречных маршрутов с m машинами и n работами с трудоемкостью $O(n^2m^2)$ может быть построено расписание S с оценкой длины*

$$C_{\max}(S) \le L_{\max} + (2m^2 - (m+3)/2)p_{\max}.$$ □

(В алгоритме построения расписания дважды используется алгоритм решения задачи КСВ из главы I: отдельно для множества работ прямого маршрута и отдельно — для встречного. После этого расписание для каждой машины получается регулярным чередованием ее операций из прямого и встречного маршрутов.)

3. Задача R213.

Теорема 43.8 *Для любого примера задачи* R213 *его оптимум лежит в интервале* $I_{\mathrm{R213}} = [L_{\max}, L_{\max}+4p_{\max}]$. *Существует алгоритм трудоемкости* $O(n\log n)$, *который для всякого примера с* n *работами строит его расписание* S *со значением* $C_{\max}(S) \in I_{\mathrm{R213}}$.

Доказательство.

Алгоритм $\mathcal{A}_{43.2}$ построения искомого расписания S состоит из двух этапов. На первом этапе для $\hat{s}$-семейства векторов $D = \{d_1, \ldots, d_n\} \subset \mathbb{R}^2$, определенного согласно (22.11), с помощью алгоритма $\mathcal{A}_{13.1}$ (стр. 92) решаем задачу НСВ6. В результате согласно следствию 14.12 с трудоемкостью $O(n\log n)$ находим перестановку $\pi = (\pi_1, \ldots, \pi_n)$, задающую нестрогое суммирование векторов из D в семействе полуплоскостей

$$\mathcal{P}_6(2,b) = \{P(e_1,\beta_1), P(e_2,\beta_2), P(e_2-e_1,\beta_3), P(e_1-e_2,\beta_4)\}$$

с оценкой

$$\theta_6(b) = \max\{\beta_1+\beta_3,\ \beta_2+\beta_4\} \le 1.$$

Полученную перестановку π используем на втором этапе алгоритма $\mathcal{A}_{43.2}$ для задания приоритетных порядков операций каждой машины в жадном алгоритме $\mathcal{A}_{42.1}$ построения расписания, применяемом при решении задачи Акерса-Фридман в разделе 42.

Замечаем, что на реализацию каждого этапа $\mathcal{A}_{43.2}$ алгоритма требуется не более $O(n\log n)$ вычислительных операций, что дает требуемую оценку трудоемкости всего алгоритма. Докажем, что получаемое им расписание удовлетворяет оценке

$$C_{\max}(S) \le L_{\max} + 4. \tag{43.18}$$

Вновь будем предполагать, что работы занумерованы согласно перестановке π, т.е. $\pi = (1,2,\ldots,n)$, и обозначать $x^k = \sum_{j=1}^{k} x_j$.

Отметим, что для рассматриваемой задачи R213 оба маршрута: $(1,2,3)$ и $(2,1,3)$,— заканчиваются на машине M_3. В точности повторив анализ работы алгоритма $\mathcal{A}_{42.1}$, проведенный для задачи Акерса-Фридман, приходим к оценке момента ($\hat{t}_{j3}$) прихода работы J_j на машину M_3, аналогичной оценке (42.4):

$$\hat{t}_{j3} < P_\pi^{j-1}(3) + 1 + \beta,$$

где

$$\beta = \begin{cases} \beta' \doteq \max_k(P_\pi^k(1) - P_\pi^{k-1}(2)) + \max_k(P_\pi^k(2) - P_\pi^{k-1}(3)), & \text{для } j \in \mathcal{N}_1, \\ \beta'' \doteq \max_k(P_\pi^k(2) - P_\pi^{k-1}(1)) + \max_k(P_\pi^k(1) - P_\pi^{k-1}(3)), & \text{для } j \in \mathcal{N}_2. \end{cases}$$

Отсюда получаем оценку величины простоя (D_3) машины M_3, аналогичную оценке (42.6):

$$D_3 < 1 + \beta.$$

Из определения векторов $\{d_j\}$ получаем:

$$\begin{aligned} \beta' &\le \max_k \min\{(e_1-e_2, d_\pi^k), (e_1-e_2, d_\pi^{k-1})\} + \max_k \min\{(e_2, d_\pi^k), (e_2, d_\pi^{k-1})\} + 2 \\ &\le 2 + \beta_4 + \beta_2, \\ \beta'' &\le 2 + \beta_3 + \beta_1, \end{aligned}$$

откуда

$$\begin{aligned} C_{\max}(S) &= L_3 + D_3 < L_{\max} + 1 + \beta \le L_{\max} + 3 + \max\{\beta_1 + \beta_3, \beta_2 + \beta_4\} \\ &= L_{\max} + 3 + \theta_6(b) \le L_{\max} + 4. \end{aligned}$$

Теорема 43.8 доказана.

С учетом (40.5) и соотношения $\tilde{\mu}_{\mathrm{R213}} \ge \tilde{\mu}_{\mathrm{FS}}(3)$ получаем

Следствие 43.9 $\tilde{\mu}_{\mathrm{R213}} \in [3, 4]$ □

4. Задача R231.

Теорема 43.10 *Пусть имеется алгоритм $\mathcal{A}$, который для любого $\hat{s}$-семейства векторов $\{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ с трудоемкостью $\mathcal{T}_{\mathcal{A}}(n)$ решает задачу* HCB4 *с гарантированной оценкой*

$$\theta_4(b) \le \theta_4^*.$$

Тогда существует алгоритм $\mathcal{A}_{43.3}$, который для любого примера задачи R231 *с n работами с трудоемкостью $O(\mathcal{T}_{\mathcal{A}}(n) + n \log n)$ находит расписание S с оценкой длины*

$$C_{\max}(S) \le L_{\max} + (3 + \theta_4^*) p_{\max}.$$

Доказательство. Для построения искомого расписания S применим алгоритм $\mathcal{A}_{43.3}$, который отличается от алгоритма $\mathcal{A}_{42.1}$ лишь тем, лишь тем, что на шаге 1 вместо задачи HCB3 решается задача HCB4.

Пусть в результате решения задачи HCB4 найдена перестановка векторов $\pi = (\pi_1, \ldots, \pi_n)$, задающая нестрогое суммирование векторов из $D = \{d_1, \ldots, d_n\} \subset \mathbb{R}^2$ в семействе полуплоскостей $\mathcal{P}_4(2, b) = \{P(e_1 - e_2, \beta_1), P(e_2, \beta_2), P(-e_1, \beta_3)\}$, где

$$\theta_4(b) = \max\{\beta_1 + \beta_2,\ \beta_2 + \beta_3\} \le \theta_4^*. \tag{43.19}$$

Вновь для удобства выкладок считаем, что $p_{\max} = 1$ и работы перенумерованы согласно перестановке π, иначе говоря, считаем, что $\pi = (1, 2, \ldots, n)$.

Аналогично тому, как это делалось в разделе 42 для задачи АФ(3), оценим сверху момент прихода каждой работы $J_j \in \mathcal{J}$ на место выполнения своей второй и своей третьей операции. Получаем, что

- всякая работа $J_j \in \mathcal{J}_1$ приходит на машину M_3 раньше момента $P_\pi^{j-1}(3) + 3 + \beta_1 + \beta_2$;
- всякая работа $J_j \in \mathcal{J}_2$ приходит на машину M_3 не позже момента $P_\pi^{j-1}(3) + 1 + \beta_2$ и на машину M_1 — раньше момента $P_\pi^{j-1}(1) + 3 + \beta_2 + \beta_3$.

Отсюда машина M_3 заканчивает работу не позднее момента $L_{\max}+3+\beta_1+\beta_2$, а машина M_1 — не позднее момента $L_{\max}+3+\beta_2+\beta_3$, и с учетом (43.19) получаем требуемую оценку длины расписания S.

Теорема 43.10 доказана.

Из теоремы 43.10 и следствия 14.9 вытекает следующий результат.

Теорема 43.11 *Для любого примера задачи* R231 *его оптимум лежит в интервале* $I_{\mathrm{R231}} = [L_{\max}, L_{\max}+5p_{\max}]$. *Существует алгоритм трудоемкости* $O(n\log n)$, *который для всякого примера с* n *работами строит его расписание* S *со значением* $C_{\max}(S)\in I_{\mathrm{R231}}$.

С учетом (40.5) и соотношения $\tilde{\mu}_{\mathrm{R231}} \geq \tilde{\mu}_{\mathrm{FS}}(3)$ получаем

Следствие 43.12 $\tilde{\mu}_{\mathrm{R231}} \in [3,5]$ □

44 Задача Job Shop и общая задача G

Теорема 44.1 *Пусть имеется алгоритм* $\mathcal{A}$ *трудоемкости* $\mathcal{T}_{\mathcal{A}}(m,n)$, *который для любого множества* $\Psi\subset\mathbb{R}^m$, *центрально симметричного относительно некоторой точки, и любого семейства векторов* $X=\{x_1,\ldots,x_n\}\subset\Psi,\ \Sigma(X)=0$, *находит такую перестановку* $\pi=(\pi_1,\ldots,\pi_n)$, *что*

$$\sum_{i=1}^{k} x_{\pi_i}\in\varphi(m)\Psi,\ k\in\mathbb{N}_n.$$

Тогда для задачи job shop с m *машинами,* n *работами и не более чем* r *операциями каждой работы можно предложить алгоритм трудоемкости* $O(\mathcal{T}_{\mathcal{A}}(mr,n)+mrn)$, *вычисляющий расписание* S *с оценкой длины*

$$C_{\max}(S)\leq L_{\max}+\psi p_{\max}, \tag{44.1}$$

где

$$\psi=(r-1)(r\varphi(mr)+2r-1). \tag{44.2}$$

Доказательство. Искомое расписание S строится алгоритмом $\mathcal{A}_{44.1}$, состоящим из 7 шагов.

Алгоритм $\mathcal{A}_{44.1}$

1. Каждое из множеств O^j дополним $r-r_j$ операциями нулевой длины, произвольно доопределив для них функцию $M(o_q^j)\in\mathcal{M}$.

2. Доупорядочим каждое множество O^j до линейного порядка и занумеруем операции $o^j_q \in O^j$ в соответствии с этим порядком: $o^j_1 \to o^j_2 \to \cdots \to o^j_r$.

3. Для каждой работы $J_j,\ j \in \mathbb{N}_n$, количество ее операций увеличим в m раз, рассматривая для каждого $q \in \mathbb{N}_r$ вместо одной операции o^j_q множество операций $\{o^q_{ji} \,|\, i \in \mathbb{N}_m\}$ "уровня q". При этом операция o^q_{ji} выполняется на машине $M_i,\ i \in \mathbb{N}_m$; операция, выполняемая на машине $M(o^j_q)$, совпадает с o^j_q, а остальные $m-1$ фиктивных операций имеют нулевую длительность.

4. Выровняем нагрузку на всех машинах путем увеличения длительностей некоторых операций без изменения величин $L_{\max}$ и $p_{\max}$. (Мы можем это сделать, поскольку все машины имеют теперь одинаковое количество операций, равное rn'.) Далее считаем, что p^q_{ji} обозначает новую длительность операции o^q_{ji}.

5. Для каждой работы $J_j,\ j \in \mathbb{N}_n$, определим вектор

$$x_j = (p^1_{j1}, \ldots, p^1_{jm}, p^2_{j1}, \ldots, p^2_{jm}, \ldots, p^r_{j,m}) \in [0,1]^{rm} \doteq \Psi.$$

Определим семейство векторов $X = \{x_1, \ldots, x_n\} \subset \Psi$, вектор $x = \Sigma(X)/n$ и семейство $X' = X - x = \{x'_j \doteq x_j - x \,|\, j \in \mathbb{N}_n\}$. Тогда $X' \subset \Psi - x$ и $\Sigma(X') = 0$. Используя алгоритм $\mathcal{A}$ из условия теоремы, найдем такую нумерацию векторов из X', что

$$\sum_{j=1}^{k} x'_j \in \varphi(mr)(\Psi - x),\ k \in \mathbb{N}_n,$$

или в терминах векторов $\{x_j\}$:

$$\sum_{j=1}^{k} x_j - (k - \varphi(mr))x \in \varphi(mr)\Psi,\ k \in \mathbb{N}_n. \tag{44.3}$$

Для каждого уровня $q \in \mathbb{N}_r$ и каждой машины $M_i,\ i \in \mathbb{N}_m$, определим множество операций $O^q_i = \{o^q_{ji} \,|\, j \in \mathbb{N}_n\}$, а сумму длительностей операций из множества O^q_i обозначим L^q_i. Согласно п.4, имеем

$$\sum_{q=1}^{r} L^q_i = L_i = L_{\max},\ i \in \mathbb{N}_m. \tag{44.4}$$

Из (44.3) вытекают соотношения

$$\sum_{j=1}^{k} p^q_{ji} = (k - \varphi(mr))L^q_i/n + \varphi(mr)\delta^q_{ki},\ q \in \mathbb{N}_r;\ k \in \mathbb{N}_n;\ i \in \mathbb{N}_m. \tag{44.5}$$

где $\delta^q_{ki} \in [0,1],\ q \in \mathbb{N}_r;\ k \in \mathbb{N}_n;\ i \in \mathbb{N}_m$.

6. Для некоторого целого числа Δ (значение которого определим позднее) для каждого $q \in \mathbb{N}_r$ и $i \in \mathbb{N}_m$, расширим множество O_i^q до множества $\tilde{O}_i^q$, добавив в него $\Delta(r-1)$ фиктивных операций длины L_i^q/n. Обозначим $\tilde{O} = \cup_{q,i}\tilde{O}_i^q$, $n' = n+\Delta(r-1)$, и для каждого $q \in \mathbb{N}_r$ определим интервал номеров $I^q \doteq \{(q-1)\Delta+1, \ldots, (q-1)\Delta+n\}$. Операции множества $\tilde{O}_i^q$ будем обозначать $\tilde{o}_{ji}^q$, причем занумеруем их таким образом, что $\tilde{o}_{ji}^q = o_{j-(q-1)\Delta,i}^q$ для $j \in I^q$. Для каждого $j \in \mathbb{N}_{n'}$ совокупность операций $\{\tilde{o}_{ji}^q \,|\, q \in \mathbb{N}_r;\ i \in \mathbb{N}_m\}$ для удобства будем называть "работой" $\tilde{J}_j$.

7. В качестве расписания на множестве операций $\tilde{\mathcal{O}}$ возьмем их приоритетную укладку $\tilde{S}$, в которой на каждой машине $M_i \in \mathcal{M}$ операции выполняются в последовательности $(\tilde{o}_{1i}^1, \ldots, \tilde{o}_{1i}^r, \tilde{o}_{2i}^1, \ldots, \tilde{o}_{2i}^r, \ldots, \tilde{o}_{n'i}^r)$, т.е. сначала выполняются все r операций "работы" $\tilde{J}_1$, затем — r операций "работы" $\tilde{J}_2$, и т.д. Тогда моменты начала $(\tilde{s}_{ki}^\nu)$ и окончания $(\tilde{c}_{ki}^\nu)$ операции $\tilde{o}_{ki}^\nu$ в расписании $\tilde{S}$ вычисляются по формулам:

$$\tilde{s}_{ki}^\nu = \sum_{j=1}^{k-1}\sum_{q=1}^{r}\tilde{p}_{ji}^q + \sum_{q=1}^{\nu-1}\tilde{p}_{ki}^q = \sum_{q=1}^{\nu-1}\sum_{j=1}^{k}\tilde{p}_{ji}^q + \sum_{q=\nu}^{r}\sum_{j=1}^{k-1}\tilde{p}_{ji}^q, \tag{44.6}$$

$$\tilde{c}_{ki}^\nu = \tilde{s}_{ki}^\nu + \tilde{p}_{ki}^\nu = \sum_{q=1}^{\nu}\sum_{j=1}^{k}\tilde{p}_{ji}^q + \sum_{q=\nu+1}^{r}\sum_{j=1}^{k-1}\tilde{p}_{ji}^q, \tag{44.7}$$

Моменты начала операций $\tilde{o}_{ji}^q$, соответствующих операциям из исходного множества $\mathcal{O}$, дадут искомое расписание S.

Алгоритм $\mathcal{A}_{44.1}$ описан.

Определим допустимые значения параметра Δ, при которых расписание S является допустимым. Достаточное условие допустимости расписания S предоставляют соотношения

$$o_{k'i_1}^\nu \to o_{k'i_2}^{\nu+1},\ \nu \in \mathbb{N}_{r-1};\ k' \in \mathbb{N}_n;\ i_1, i_2 \in \mathbb{N}_m,$$

или с учетом соответствия между o- и $\tilde{o}$-операциями, $\tilde{c}_{k'+(\nu-1)\Delta,i_1}^\nu \le \tilde{s}_{k'+\nu\Delta,i_2}^{\nu+1}$. Обозначив $k = k' + (\nu-1)\Delta$, для любого $k \in I^\nu$ из (44.6), (44.7) получим соотношения:

$$\begin{aligned}
\tilde{c}_{ki_1}^\nu - \tilde{s}_{k+\Delta,i_2}^{\nu+1} &= \sum_{q=1}^{\nu}\sum_{j=1}^{k}\tilde{p}_{ji_1}^q + \sum_{q=\nu+1}^{r}\sum_{j=1}^{k-1}\tilde{p}_{ji_1}^q - \sum_{q=1}^{\nu}\sum_{j=1}^{k+\Delta}\tilde{p}_{ji_2}^q - \sum_{q=\nu+1}^{r}\sum_{j=1}^{k-1+\Delta}\tilde{p}_{ji_2}^q \\
\text{(из (44.5))} &\le \sum_{q=1}^{\nu}\frac{k-\varphi(mr)}{n}L_{i_1}^q + \sum_{q=\nu+1}^{r}\frac{k-1-\varphi(mr)}{n}L_{i_1}^q \\
&- \sum_{q=1}^{\nu}\frac{k+\Delta-\varphi(mr)}{n}L_{i_2}^q - \sum_{q=\nu+1}^{r}\frac{k+\Delta-1-\varphi(mr)}{n}L_{i_2}^q + \varphi(mr)r \\
\text{(из (44.4))} &= \varphi(mr)r - \frac{1}{n}\sum_{q=\nu+1}^{r}L_{i_1}^q + \frac{1}{n}\sum_{q=\nu+1}^{r}L_{i_2}^q - \frac{\Delta L_{\max}}{n} \\
&\le \varphi(mr)r + (r-\nu) - \frac{\Delta L_{\max}}{n} \le 0.
\end{aligned}$$

Таким образом, для допустимости расписания S достаточно выполнения неравенства $\Delta \geq (\varphi(mr)r + r - 1)n/L_{\max}$. Положим

$$\Delta = \lceil (\varphi(mr)r + r - 1)n/L_{\max} \rceil. \tag{44.8}$$

Тогда длина расписания S оценивается сверху через длину приоритетной укладки $\tilde{S}$, которая совпадает с максимальной из нагрузок машин $M_1, \ldots, M_m$ операциями из множества $\tilde{O}$. В силу (44.4), эти нагрузки совпадают и равны $\tilde{l}_{\max} = L_{\max} + \Delta(r-1)L_{\max}/n$. С учетом (44.8) получаем оценку:

$$\begin{aligned} C_{\max}(S) &\leq \tilde{l}_{\max} < L_{\max} + \frac{(r-1)L_{\max}}{n} + (\varphi(mr)r + r - 1) \\ &\leq L_{\max} + (r-1)(\varphi(mr)r + 2r - 1). \end{aligned}$$

Теорема 44.1 доказана.

Из теоремы 44.1 и следствия 11.7 (стр. 75) получаем следующий результат.

Теорема 44.2 *Для любого примера задачи* JS *с m машинами, n работами и не более чем r операциями каждой работы его оптимум лежит в интервале $I_{JS} = [L_{\max}, L_{\max} + r(r-1)(mr+1)p_{\max}]$. Существует алгоритм трудоемкости $O(n^2m^2r^2)$, который для любого примера строит расписание S со значением длины из интервала I_{JS}.*

Замечаем, что когда количество операций каждой работы не превосходит константы, теорема 44.2 позволяет локализовать оптимум в интервале, длина которого не зависит от числа работ и линейна от числа машин.

Перейдем теперь к общей задаче G, т.е. к задаче нахождения оптимального расписания в модели G, сформулированной в разделе 22. От задачи JS она отличается двумя моментами:

- частичный порядок на каждом множестве O^j может быть произвольным;
- для каждой операции o_q^j задано множество допустимых исполнителей $\mathcal{M}_q^j \subseteq \mathcal{M}$.

Теорема 44.3 *Для любого примера задачи* G *его оптимум лежит в интервале $I_G = [L_{\max}^{\text{opt}}, L_{\max}^{\text{opt}} + (\psi+1)p_{\max}]$, где функция ψ определена согласно (44.2) и не зависит от числа работ, а величина $L_{\max}^{\text{opt}}$ эффективно находится из решения задачи* МПРС *(см. раздел 19) на сети с $N = \sum_{J_j \in \mathcal{J}} r_j \leq rn$ внутренними вершинами и m стоками.*

Существует полиномиальный алгоритм, который для всякого входа задачи G *строит приближенное расписание S со значением длины из интервала I_G.*

Доказательство. Алгоритм построения расписания S с оценкой $C_{\max}(S) \leq L_{\max}^{\text{opt}} + (\psi+1)p_{\max}$ состоит из трех этапов. **Этап 1.** Произвольно доупорядочиваем каждое из множеств O^j. **Этап 2.** Опираясь на теорему 19.2, с трудоемкостью $O(N^2m^2)$ находим функцию $M : \mathcal{O} \to \mathcal{M}$, решая задачу РКЗ о нахождении равномерного распределения E' камней-операций по кучам-машинам с учетом ограничений на множество допустимых исполнителей $\mathcal{M}_q^j$ каждой операции o_q^j. При этом гарантируется, что максимальная

из загрузок машин будет отличаться от своего оптимума не более чем на величину длительности одной операции: $L_{\max} = L_{\max}(E') \leq L^{\text{opt}}_{\max} + p_{\max}$. **Этап 3.** Опираясь на теорему 44.1, находим расписание $S = \{s^j_q \,|\, o^j_q \in \mathcal{O}\}$ с оценкой длины $C_{\max}(S) \leq L_{\max} + \psi p_{\max}$.

Теорема 44.3 доказана.

Из теоремы 44.1 и следствия 11.7 получаем следующий результат.

Следствие 44.4 *Существует алгоритм трудоемкости $O(n^2m^2r^2)$, находящий для любого примера задачи* G *с m машинами, n работами и не более чем r операциями каждой работы его расписание S с гарантированной абсолютной оценкой точности:*

$$C_{\max}(S) - C_{\max}(S_{\text{opt}}) \leq (r(r-1)(mr+1)+1)p_{\max}. \qquad \square$$

45 Открытые вопросы и проблемы

Естественно, что в той части теории расписаний, исследованию задач которой посвящена данная диссертация, остается еще масса нерешенных проблем. Практически не известны минимальные интервалы локализации оптимумов для всех исследуемых задач (кроме небольшого числа исключений). Однако мы не будем ставить здесь глобальных проблем, а сфокусируем внимание на более конкретных и интересных на наш взгляд открытых вопросах.

1. Каковы минимальные интервалы локализации оптимумов в задачах R213, R231, $F4||C_{\max}$?

2. Верно ли, что длину интервала локализации оптимумов в задаче Акерса-Фридман можно ограничить величиной $O(m^3)$?

3. Найти нелинейную от m нижнюю оценку длины интервала локализации оптимумов в задаче Джонсона.

4. Найти “плохой” пример для трехмашинной задачи Акерса-Фридман, который бы давал хоть немного лучшую нижнюю оценку длины интервала локализации оптимумов — по сравнению с известным “плохим” примером для задачи Джонсона.

Литература

[1] Адельсон-Вельский Г.М., Диниц Е.А., Карзанов А.В. Потоковые алгоритмы. М.: Наука, 1975.

[2] Аксенов В.А. Полиномиальный алгоритм приближенного решения одной задачи теории расписаний // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1988. С. 8–11. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **28**.)

[3] Бабушкин А.И., Башта А.Л., Белов И.С. Построение календарного плана для многомаршрутной задачи трех станков // *Автомат. и Телемех.* 1976. Т. **7** С. 154–158.

[4] Бабушкин А.И., Башта А.Л., Белов И.С. Построение календарного плана для задачи встречных маршрутов // *Кибернетика*. 1977. N 4. С. 130–135.

[5] Бабушкин А.И., Башта А.Л., Белов И.С., Душин Б.И. Минимизация цикла работы поточной линии // *Автомат. и Телемех.* 1975. Т. **6**. С. 161–167.

[6] Белов И.С., Столин Я.Н. Алгоритм в одномаршрутной задаче календарного планирования. В кн.: Математическая экономика и функциональный анализ. М.: Наука, 1974. С. 248–257.

[7] Берж К. Теория графов и ее применения. М.: Издательство иностранной литературы, 1962.

[8] Боровков А.А. К вероятностной постановке двух экономических задач // *Докл. АН СССР*, 1962, Т. **146**, N 5, С. 983–986.

[9] Воегингер Г.Д., Севастьянов С.В. Линейная аппроксимационная схема для многопроцессорной задачи open shop // *Дискретный анализ и исследование операций*. Сер. 1. 1999. Т. **6**, N 2. С. 3–22.

[10] Гимади Э.Х., Глебов Н.И. Дискретные экстремальные задачи принятия решений. Учебное пособие // Новосибирск: НГУ, 1991.

[11] Гимади Э.Х., Глебов Н.И., Перепелица В.А. Исследования по теории расписаний // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1974. С. 3–10. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **12**.)

[12] Гимади Э.Х., Глебов Н.И., Перепелица В.А. Алгоритмы с оценками для задач дискретной оптимизации // *Проблемы кибернетики*. М.: Наука, 1975. Вып. **31**. С. 35–42.

[13] Гимади Э.Х., Залюбовский В.В., Севастьянов В.В. Полиномиальная разрешимость задач календарного планирования со складируемыми ресурсами и директивными сроками // *Дискретный анализ и исследование операций*. Сер. 2. 2000. Т. **7**, N 1. С. 9–34.

[14] Гимади Э.Х., Перепелица В.А. Статистически эффективный алгоритм выделения гамильтонова контура (цикла) // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1973. С. 15–28. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. 22.)

[15] Гимади Э.Х., Перепелица В.А. Асимптотически точный подход к решению задачи коммивояжера // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1974. С. 35–45. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **12**.)

[16] Гринберг В.С., Севастьянов С.В. О величине константы Штейница // *Функциональный анализ и его приложения*. 1980. Т. **14**. С. 56–57.

[17] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

[18] Душин Б.И. Алгоритм для 2-маршрутной задачи Джонсона // *Кибернетика*. 1988. N 3. С. 53–58.

[19] Душин Б.И. Алгоритм для p-маршрутной задачи Джонсона // *Кибернетика*. 1989. N 2. С. 119–122.

[20] Кадец М.И. Об одном свойстве векторных ломаных в n-мерном пространстве // *Успехи мат. наук*. 1953. Т. **8**. С. 139–143.

[21] Кадец В.М., Кадец М.И. Перестановки рядов в пространствах Банаха // Тарту: Тартуский государственный университет, 1988.

[22] Каширских К.Н., Поттс К.Н., Севастьянов С.В. Улучшенный алгоритм решения двухмашинной задачи flow shop с неодновременным поступлением работ // *Дискретный анализ и исследование операций*. 1997. Т. **4**, N 1. С. 13–32.

[23] Каширских К.Н., Поттс К.Н., Севастьянов С.В. Улучшенный алгоритм решения двухмашинной задачи flow shop с неодновременным поступлением работ // Международная Сибирская конференция по исследованию операций (SCOR-98). Новосибирск, 22–27 июня 1998. Материалы конференции. Новосибирск: Изд-во Ин-та математики, 1998. С. 87.

[24] Кононов А.В., Севастьянов С.В. О сложности нахождения связной предписанной раскраски вершин графа // *Дискретный анализ и исследование операций*. Сер. 1. 2000. Т. **7**, N 2. С. 21–46.

[25] Кононов А.В., Севастьянов С.В., Черных И.Д. Полиномиальная разрешимость задачи open shop при условиях доминирования // Второй сибирский конгресс по прикладной и индустриальной математике (ИНПРИМ-96). Новосибирск, 25–30 июня 1996. Материалы конференции. Новосибирск: Изд-во Ин-та математики, 1996. С. 137–138.

[26] Неумытов Ю.Д., Севастьянов С.В. Приближенный алгоритм с точной оценкой для трехмашинной задачи встречных маршрутов // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1993. С. 53–65. (Тр./ АН. Сиб. отд-ние. Ин-т математики. Вып. **31**.)

[27] Перепелица В.А., Гимади Э.Х. К задаче нахождения минимального гамильтонова контура на графе со взвешенными дугами // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1969. С. 57–65. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **15**.)

[28] Петухов С.П., Севастьянов С.В., Черных И.Д. Линейный алгоритм решения задачи open shop с тремя машинами с относительной погрешностью 11/8 // Второй сибирский конгресс по прикладной и индустриальной математике (ИНПРИМ-96). Новосибирск, 25–30 июня 1996. Материалы конференции. Новосибирск: Изд-во Ин-та математики, 1996. С. 140–141.

[29] Рокафеллар Р. Выпуклый анализ. М.: Мир, 1973.

[30] Севастьянов С.В. Об асимптотическом подходе к некоторым задачам теории расписаний // III Всесоюз. конф. по проблемам теор. кибернетики. Тез. докл. Новосибирск, 1974. С. 67–69.

[31] Севастьянов С.В. Об асимптотическом подходе к некоторым задачам теории расписаний // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1975. С. 40–51. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **14**.)

[32] Севастьянов С.В. О приближенном решении некоторых задач теории расписаний // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1978. С. 66–75. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **32**.)

[33] Севастьянов С.В. К задаче компактного суммирования векторов // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1979. С. 77–89. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **33**.)

[34] Севастьянов С.В. О приближенном решении задачи календарного распределения // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1980. С. 49–63. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **20**.)

[35] Севастьянов С.В. Приближенные алгоритмы в задачах Джонсона и суммирования векторов // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1980. С. 64–73. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **20**.)

[36] Севастьянов С.В. О связи задачи календарного распределения с одной задачей на единичном кубе // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1980. С. 93–103. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **35**.)

[37] Севастьянов С.В. О задаче календарного распределения // V Всесоюз. конф. по проблемам теор. кибернетики. Тез. докл. Новосибирск, 1980. С. 92–93.

[38] Севастьянов С.В. О приближенном решении задачи Джонсона // V Всесоюз. конф. по проблемам теор. кибернетики. Тез. докл. Новосибирск, 1980. С. 93–95.

[39] Севастьянов С.В. Оценки и свойства функций Штейница // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1981. С. 59–73. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **36**.)

[40] Севастьянов С.В. Некоторые обобщения задачи Джонсона // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1981. С. 45–61. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **21**.)

[41] Севастьянов С.В. Алгоритмы с оценками для задач теории расписаний. Автореф. дис. на соиск. учен. степ. канд. физ.-мат. наук. (01.01.09). Новосибирск 1981.

[42] Севастьянов С.В. Алгоритмы с оценками для задач Джонсона и Акерса–Фридмана в случае трех станков // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1982. С. 51–57. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **22**.)

[43] Севастьянов С.В. Эффективное построение расписаний, близких к оптимальным, для случаев произвольных и альтернативных маршрутов деталей // *Докл. АН СССР*. 1984. Т. **276**, N 1. С. 46–48.

[44] Севастьянов С.В. Алгоритм с оценкой для задачи с маршрутами деталей произвольного вида и альтернативными исполнителями // *Кибернетика*. 1986, N 6. С. 74–79.

[45] Севастьянов С.В. Геометрия в теории расписаний // Модели и методы оптимизации. Новосибирск: Наука, Сиб. Отдел., 1988. С. 226–261. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Т. **10**.)

[46] Севастьянов С.В. Теорема о компактном суммировании векторов в двумерном пространстве // *Методы дискретного анализа*. Новосибирск: Изд-во Ин-та математики, 1988. С. 61–65. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **47**.)

[47] Севастьянов С.В. О задаче сетевого календарного планирования с ограничениями на ресурсы при наличии в сети циклов // Методы и программы решения оптимизационных задач на графах и сетях. Часть П/ Тез. докл. IV Всес. совещания. Новосибирск, 1989. С. 96–99.

[48] Севастьянов С.В. О компактном суммировании векторов // *Дискретная математика*. 1991. Т. **3**, N 3. С. 66–72.

[49] Севастьянов С.В. Полиномиально разрешимый случай задачи open-shop с произвольным числом машин // *Кибернетика и системный анализ*. 1992. N 6. С. 135–154.

[50] Севастьянов С.В. Построение приближенного расписания для системы поточного типа // *Управляемые системы*. Новосибирск: Изд-во Ин-та математики, 1993. С. 66–71. (Тр./ АН СССР. Сиб. отд-ние. Ин-т математики. Вып. **31**.)

[51] Севастьянов С.В. Эффективное построение расписаний в системах открытого типа // *Сибирский журнал исследования операций*. 1994. Т. **1**, N 1. С. 20–42.

[52] Севастьянов С.В. Нестрогое суммирование векторов в задачах теории расписаний // *Сибирский журнал исследования операций*. 1994. Т. **1**, N 2. С. 67–99.

[53] Севастьянов С.В. Нестрогое суммирование векторов на плоскости и его применение в задачах теории расписаний // *Дискретный анализ и исследование операций*. 1995. Т. **2**, N 2. С. 69–100.

[54] Севастьянов С.В., Черных И.Д. Достаточное условие эффективной разрешимости задачи open shop // *Дискретный анализ и исследование операций*. 1996. Т. **3**, N 1. С. 57–74.

[55] Севастьянов С.В. Нестрогое суммирование векторов в многооперационных задачах теории расписаний // Второй сибирский конгресс по прикладной и индустриальной математике (ИНПРИМ-96). Новосибирск, 25–30 июня 1996. Материалы конференции. Новосибирск: Изд-во Ин-та математики, 1996. С. 142.

[56] Севастьянов С.В. Применение геометрических методов в многостадийных задачах теории расписаний // Тез.докл. на 11 Международной конф. по проблемам теоретической кибернетики. Ульяновск, 10–14 июня 1996. Ульяновск: Изд-во СВНЦ, 1996. С. 180–181.

[57] Севастьянов С.В. Геометрические методы в теории расписаний // Международная Сибирская конференция по исследованию операций (SCOR-98). Новосибирск, 22–27 июня 1998. Материалы конференции. Новосибирск: Изд-во Ин-та математики, 1998. С. 35–38.

[58] Севастьянов С.В., Черных И.Д. Доказательство теорем теории расписаний с помощью компьютера // Международная Сибирская конференция по исследованию операций (SCOR-98). Новосибирск, 22–27 июня 1998. Материалы конференции. Новосибирск: Изд-во Ин-та математики, 1998. С. 90.

[59] Севастьянов С.В. Многопараметрический анализ сложности дискретных задач // Дискретный анализ и исследование операций (DAOR'2000). Материалы конференции. Новосибирск, 26 июня – 1 июля 2000. С. 61–62.

[60] Сотсков Ю.Н., Струсевич В.А., Танаев В.С. Математические модели и методы календарного планирования. Учебное пособие // Минск: Universitetskoe, 1994.

[61] Танаев В.С., Сотсков Ю.Н., Струсевич В.А. Теория расписаний. Многостадийные системы // М.: Наука, 1989.

[62] Харари Ф. Теория графов, М.: Мир, 1973.

[63] Approximation Algorithms for NP-Hard Problems, Ed. by D.S. Hochbaum, PWS, 1997.

[64] Akers, S.B. and Friedman, J. A non-numerical approach to production scheduling problems // *Oper. Res.* 1955. V. **3**. P. 429–442.

[65] Alon, N., Spencer, J., and Erdös, P. The probabilistic method. A Willey-Interscience Publication, 1992.

[66] Alon, N., Csirik, J., Sevastianov, S.V., Vestjens, A.P.A., and Woeginger, G.J. On-line and Off-line Approximation Algorithms for Vector Covering Problems // SFB-Report 78, July 1996. TU Graz, Austria.

[67] Alon, N., Csirik, J., Sevastianov, S., Vestjens, A., and Woeginger, G. On-line and off-line algorithms for vector covering // in: Algorithms — ESA'96, 4th Annual European Symposium. Proceedings. Springer-Verlag, 1996. P. 406-418. (*Lecture Notes in Comp.Sci.* V. **1136**.)

[68] Alon, N., Azar, Y., Csirik, J., Epstein, L., Sevastianov, S.V., Vestjens, A.P.A., and Woeginger, G.J. On-line and off-line approximation algorithms for vector covering problems // Fourth European Symposium on Algorithms (Barcelona, 1996). *Algorithmica*. 1998. V. **21**, N. 1. P. 73–79.

[69] Avgustinovich, S.V. and Sevast'janov, S.V. Vector Summation within Minimal Angle // *Computational Geometry: Theory and Appl.* 1993. V. **2**. P. 235–239.

[70] Banaszczyk, W. The Steinitz Constant of the Plane // *J. Reine und Angew. Math.* 1987. V. **373**. P. 218–220.

[71] Banaszczyk, W. A note on the Steinitz Constant of the Euclidean Plane // *C. R. Math. Rep. Acad. Sci. Canada*. 1990. V. **12**, N. 4. P. 97–102.

[72] Banaszczyk, W. The Steinitz Lemma on Rearrangement of Series for Nuclear Spaces // *J. Reine und Angew. Math.* 1990. V. **403**. P. 187–200.

[73] Banaszczyk, W. The Steinitz lemma in l^2_∞ // *Period. Math. Hung.* 1991. V. **22**. P. 183–186.

[74] Bárány, I. and Fiala, T. Nearly optimum solution of multi-machine scheduling problems // *Szigma*. Mat.-Közgaz-dasági Folyóirat. 1982. V. **15**, N. 3. P. 177–191. (Hungarian)

[75] Bárány, I. and Grinberg, V.S. A vector-sum theorem in two-dimensional space // *Period. Math. Hung.* 1985. V. **16**. P. 135–138.

[76] Beck, J. and Fiala, T. "Integer-making" theorems // *Discrete Appl. Math.* 1981. V. **3**. P. 1–8.

[77] Bergström, V. Ein neuer Beweis eines Satzes von E.Steinitz // *Abhendlungen aus dem Mathematischen Seminar der Hamburgischen Universitat*. 1931. V. **8**. P. 148–152.

[78] Brucker, P. Scheduling Algorithms. Berlin: Springer, 1995.

[79] Carlier, J. and Pinson, E. An algorithm for solving the job-shop problem // *Management Science*. 1989. V. **35**. P. 164–176.

[80] Chen, B. Scheduling Multiprocessor Flow Shops // D.-Z. Du and J. Sun (Eds.). New Advances in Optimization and Approximation. Kluwer, 1994. P. 1–8.

[81] Chen, B., Potts, C.N., and Woeginger, G.J. A review of machine scheduling: complexity, algorithms and approximability // *Handbook of Combinatorial Optimization*. D.-Z. Du and P. M. Pardalos (Eds.). Kluwer Academic Publishers, 1998. P. 21–169.

[82] Chen, B. and Strusevich, V.A. Approximation algorithms for three-machine open shop scheduling // *ORSA Journal on Computing*. 1993. V. **5**. P. 321–326.

[83] Dvoretzky, A. Problem // Proceedings of Symposia in Pure Mathematics. V. **7**. Convexity. Providence, RI: Amer. Math. Soc., 1963.

[84] Fiala, T. An algorithm for the open-shop problem // *Math. Oper. Res.* 1983. V. **8**. P. 100–109.

[85] Gabow, H.N. and Kariv, O. Algorithms for Edge Coloring Bipartite Graphs and Multigraphs // *SIAM J.Comput.* 1982. V. **11**. P. 117–129.

[86] Garey, M.R. and Johnson, D.S. Complexity Results for Multiprocessor Scheduling under Resource Constraints // *SIAM J.Comput.* 1975. V. **4**. P. 397–411.

[87] Gonzalez, T. and Sahni, S. Open Shop Scheduling to Minimize Finish Time // *J. ACM*. 1976. V. **23**, N. 4. P. 665–679.

[88] Graham, R.L. Bounds for certain multiprocessing anomalies // *Bell System Technical Journal*. 1966. V. **45**. P. 1563–1581.

[89] Grinberg, V.S. and Sevast'janov, S.V. Value of the Steinitz constant // *Functional Anal. Appl.* 1980. V. **14**. P. 125–126.

[90] Hall, L.A. Approximability of flow shop scheduling // in: Proceedings of the 36th IEEE Symposium on Foundations of Computer Science. 1995. P. 82–91.

[91] Jansen, K., Solis-Oba, R., and Sviridenko, M. Makespan minimization in job shops: a polynomial time approximation scheme // Proceedings of the 31st Annual ACM Symp. on Theory of Computing. 1999. P. 394–399.

[92] John, F. Extremum problems with inequalities as subsidiary conditions // *Studies and essays, presented to R. Courant on his 60th Birthday*. N.Y., 1948. P. 187–204.

[93] Johnson, S.M. Optimal Two and Three-Stage Production Schedules with Set-up Times Included // *Nav. Res. Log. Quart.* 1954. V. **1**, N. 1. P. 61–68.

[94] Kashyrskikh, K.N., Kononov, A.V., Sevastianov, S.V., and Tchernykh, I.D. Polynomially solvable class of instances of the 2-stage 3-machine open shop problem // Дискретный анализ и исследование операций (DAOR'2000). Материалы конференции. Новосибирск, 26 июня – 1 июля 2000. С. 174.

[95] Kononov, A., Sevastianov, S., and Tchernykh, I. When difference in machine loads leads to efficient scheduling in open shops // *Annals of Oper. Res.* 1999. V. **92**. P. 211–239.

[96] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. Sequencing and scheduling: algorithms and complexity // *Handbooks in Operations Research and Management Science* V. **4**. Amsterdam: North Holland, 1993. P. 445–522.

[97] Lee, C.-Y. and Vairaktarakis, G.L. Minimizing Makespan in Hybrid Flow-shops // *Opns. Res. Letters*. 1994. V. **16**. P. 149–158.

[98] Leichtweiss, K. Zwei Extremalprobleme der Minkowski-Geometrie // *Math. Z.* 1955. V. **62**. P. 37–49.

[99] Mas-Colell, A. The theory of general economic equilibrium. A differentiable approach // Cambridge: University-press, 1985.

[100] Nilsson, Bengt J. (private communication).

[101] Olson, J. and Spencer, J. Balancing families of sets // *J. Comb. Theory*. (Ser. A). 1978. V. **25**. P. 29–37.

[102] Potts, C.N. Analysis of heuristics for two-machine flow-shop sequencing subject to release dates // *Math. Oper. Res.* 1985. V. **10**, N. 4. P. 576–584.

[103] Potts, C.N., Sevast'janov, S.V., Strusevich, V.A., Van Wassenhove, L.N., and Zvaneveld, C.M. The two-stage assembly scheduling problem: complexity and approximation // Report 9256/A. Rotterdam, The Netherlands: Erasmus University Rotterdam, 1992.

[104] Potts, C.N., Sevast'janov, S.V., Strusevich, V.A., Van Wassenhove, L.N., and Zvaneveld, C.M. The two-stage assembly scheduling problem: complexity and approximation // *Operations Research*. 1995. V. **43**, N. 2. P. 346–355.

[105] Schuurman, P. and Woeginger, G.J. Approximation Algorithms for the Multiprocessor Open Shop Problem // Report Woe-13, October 1997. TU Graz, Austria.

[106] Shmoys, D.B., Stein, C., and Wein, J. Improved Approximation Algorithms for Shop Scheduling Problems // *SIAM J. on Computing*. 1994. V. **23**. P. 617–632.

[107] Sevastianov, S.V. Efficient construction of schedules close to optimal for the class of arbitrary and alternative routes of parts // *Soviet Math. Dokl.* 1984. V. **29**. P. 447–450. (translated from Russian)

[108] Sevast'janov S.V. An algorithm with an estimate for a problem with routings of parts of arbitrary shape and alternative executors // *Cybernetics*. 1986. V. **22**. P. 773–781. (translated from Russian)

[109] Sevast'janov, S.V. On a geometric method in scheduling theory // 14th Int. symp. on math. programming (abstracts). Amsterdam, The Netherlands, August 5–9, 1991.

[110] Sevast'janov, S.V. Polynomially solvable case of the open-shop problem with arbitrary number of machines // *Cybernet. Systems Anal.* 1992. V. **28**, N. 6. P. 918–933 (1993). (translated from Russian)

[111] Sevast'janov, S.V. New polynomially solvable cases of the open shop problem // in: Operations Research 1994. Intern. Conf. on Oper. Res., Aug. 30 – Sept. 2, 1994. Program and Abstracts. Berlin: TU Berlin, 1994. P. 111.

[112] Sevast'janov, S.V. Nonstrict vector summation in job shop scheduling // in: Operations Research 1994. Intern. Conf. on Oper. Res., Aug. 30 – Sept. 2, 1994. Program and Abstracts. Berlin: TU Berlin, 1994. P. 112.

[113] Sevast'janov, S.V. On some geometric methods in scheduling theory: a survey // *Discrete Appl. Math.* 1994. V. **55**. P. 59–82.

[114] Sevast'janov, S.V. Vector summation in Banach space and polynomial algorithms for flow shops and open shops // *Math.Oper.Res.* 1995. V. **20**, N. 1. 90–103.

[115] Sevastianov, S.V. Nonstrict vector summation in multi-operation scheduling // Memorandum COSOR 95-37. Eindhoven University of Technology, 1995.

[116] Sevast'yanov, S.V. Efficient scheduling in open shops // A. D. Korshunov (ed.). Discrete analysis and oper. res. Dordrecht: Kluwer, 1996. P. 235–255.

[117] Sevast'yanov, S.V. Nonstrict vector summation in scheduling problems // A. D. Korshunov (ed.). Discrete analysis and oper. res. Dordrecht: Kluwer, 1996. P. 257–287.

[118] Sevast'yanov, S.V. and Woeginger, G.J. Makespan minimization in open shops: a polynomial time approximation // SFB-Report **68**, Mai 1996. TU Graz, Austria.

[119] Sevast'yanov, S.V. and Woeginger, G.J. Makespan Minimization in Preemptive Two Machine Job Shops // SFB-Report 81, July 1996. TU Graz, Austria.

[120] Sevast'yanov, S.V. Nonstrict vector summation in the plane and its application to scheduling problems // A. D. Korshunov (ed.), Operations research and discrete analysis. Dordrecht: Kluwer, 1997. P. 241–272.

[121] Sevastianov, S.V. Seven problems: so different yet close // in: R. Burkard and G. Woeginger (Eds.). Algorithms — ESA'97, 5th Annual European Symposium, Graz, Austria, September 1997. Proceedings. Springer-Verlag, 1997. P. 443–458. (*Lecture Notes in Comp.Sci.* V. **1284**.)

[122] Sevast'janov, S.V. and Banaszczyk, W. To the Steinitz lemma in coordinate form // *Discrete Math.* 1997. V. **169**. P. 145–152.

[123] Sevastianov, S.V. and Woeginger, G.J. Makespan minimization in open shops: A polynomial time approximation scheme // *Math. Programming*. 1998. V. **82**. P. 191–198.

[124] Sevastianov, S.V. and Tchernykh, I.D. Computer-Aided Way to Prove Theorems in Scheduling // G. Bilardi, a.o. (Eds.). Algorithms — ESA'98. 6th Annual European Symposium, Venice, Italy, August 1998. Proceedings. Springer-Verlag, 1998, 502–513. (Lecture Notes in Comp.Sci., V. **1461**)

[125] Sevastianov, S.V. and Woeginger, G.J. Makespan minimization in preemptive two machine job shops // *Computing*. 1998. V. **60**, N. 1. P. 73–79.

[126] Sevastianov, S.V. Compact vector summation: how it works in different fields of discrete mathematics // Y. Rabani, D. Shmoys, G. Woeginger (Eds.). Combinatorial Approximation Algorithms, Dagstuhl-Seminar-Report. V. **187**. 1998; 18.08.–22.08.97 (9734). P. 14–15.

[127] Sevastianov, S. Nonstrict vector summation in multi-operation scheduling // *Annals of Oper.Res.* 1998. V. **83**. P. 179–211.

[128] Sevastianov, S.V. Geometrical heuristics for multiprocessor flowshop scheduling with uniform machines at each stage // Symposium on operations research 1999, Magdeburg, September 1–3, 1999. Book of Abstracts. P. 101–102.

[129] Sevastianov, S.V. Multi-parameter complexity analysis of discrete problems // Proc. International Workshop *Discrete Optimization Methods in Scheduling and Computer-Aided Design*, Minsk, September 5–6, 2000. Minsk, 2000. P. 172–174.

[130] Shmoys, D.B., Stein, C., and Wein, J. Improved Approximation Algorithms for Shop Scheduling Problems // Proceedings of the 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA), 1991. P. 148–157.

[131] Steinitz, E. Bedingt Konvergente Reihen und Convexe Systeme // *J. Reine und Angew. Math.* 1913. V. **143**. P. 128–175.

[132] Tarjan, R.E. Data structures and network algorithms. Philadelphia, PA: SIAM, 1983.

[133] Tchernykh, I.D., Kashyrskikh, K.N., and Sevastianov, S.V. 4-parameter complexity analysis of the open shop problem // Дискретный анализ и исследование операций (DAOR'2000). Материалы конференции. Новосибирск, 26 июня – 1 июля 2000. С. 175.

[134] Wallis, W.D., a.o. Combinatorics: Room Squares, Sum-Free Sets, Hadamard Matrices // *Lecture Notes in Math.* 1972. V. **292**.

[135] Wallis, J.S. Williamson matrices of even order // *Lecture Notes in Math.* 1974. V. **403**. P. 132–142.

[136] Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevastianov, S.V., and Shmoys, D.B. Short shop schedules // *Oper.Res.* 1997. V. **45**, N. 2. P. 288–294.