Экономико-математическая библиотека

В. С. МИХАЛЕВИЧ
А. И. КУКСА

МЕТОДЫ ПОСЛЕДОВАТЕЛЬНОЙ ОПТИМИЗАЦИИ

Экономико-
математическая
библиотека

В. С. МИХАЛЕВИЧ,
А. И. КУКСА

МЕТОДЫ ПОСЛЕДОВАТЕЛЬНОЙ ОПТИМИЗАЦИИ В ДИСКРЕТНЫХ СЕТЕВЫХ ЗАДАЧАХ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ

МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1983

22.18
М 69
УДК 519.6

Методы последовательной оптимизации в дискретных сетевых задачах оптимального распределения ресурсов. Михалевич В. С., Кукса А. И.— М.: Наука. Главная редакция физико-математической литературы, 1983.— 208 с.

Книга посвящена актуальным приложениям численных методов решения дискретных задач оптимизации. Последовательно изложены современные методы решения задач оптимизации распределения ресурсов на сетях, основанные на идее последовательной оптимизации, и проанализирована эффективность таких методов.

Для специалистов в области прикладной математики и кибернетики, а также специалистов в области экономики, применяющих в своей работе математические методы.

Рис. 16. Табл. 16. Библ. 119 назв.

$М\frac{1502000000 - 027}{053(02)-83}62\text{-}82$

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Дискретные сетевые задачи оптимального распределения ресурсов являются обобщением соответствующих задач теории расписаний. Как известно, теория расписаний имеет дело с совокупностями действий (операций, работ и т. п.) и ресурсами (станками, процессорами и т. п.), необходимыми для осуществления этих действий. Экстремальные задачи распределения ресурсов возникают в связи с ограничениями на общие количества ресурсов или на сроки достижения конечных результатов совокупности действий. В публикациях по теории расписаний традиционно уделяется большое внимание задачам типа «станки — детали». В этих задачах технологический маршрут совокупности деталей представляется ориентированным графом операций, состоящим из нескольких несвязанных между собой цепей. Отношение «операции — ресурсы» описывается указанием имени станка, выполняющего данную операцию. Каждый станок одновременно выполняет не более одной операции. Однако многие важные практические задачи не вписываются в указанную схему. Представляет интерес изучение более общих операционных моделей сложных составных действий, в которых:

— каждая операция описывается действительным вектором, компонентами которого являются длительность операции и уровни потребления ресурсов;

— технологические ограничения описываются произвольным ориентированным графом без петель и контуров;

— общие уровни наличия нескольких видов ресурсов заданы как функции времени.

Задачи с указанными особенностями условимся называть сетевыми задачами распределения ресурсов и составления расписаний. В сущности, все известные задачи теории расписаний могут быть рассмотрены как сетевые (с пустым, в частности, отношением предшествования операций). Не претендуя на полноту, упомянем лишь наиболее обширные области применения задач рассматриваемого класса.

Традиционной областью применения сетевых задач распределения ресурсов и составления расписаний является сетевое планирование с ограниченными ресурсами. Экстремальные задачи распределения ресурсов в сетевом планировании были осознаны и сформулированы еще в конце 50-х годов. В теории сетевого планирования они занимают одно из центральных мест. Однако практическое решение этих задач в автоматизированных системах СПУ связано с рядом трудностей. Они вызваны в основном недостаточной разработкой вопросов создания и автоматизированного ведения баз нормативных данных, типовых технологических схем планируемых процессов, недостаточной степенью автоматизации процессов подготовки данных, оперативного и наглядного отображения результатов счета. Требования оперативности ослабляются в тех случаях, когда речь идет о перспективном крупномасштабном планировании, разовом решении задач с целью выбора планов развития на большой отрезок времени, когда цена оптимального решения сравнительно велика. Одно из важных применений такого рода — использование сетевых задач оптимального распределения ресурсов в математическом обеспечении программно-целевого планирования народнохозяйственных программ.

В середине 60-х годов в СССР начали широко обсуждаться методы математического обеспечения процедур программно-целевого планирования. В частности, была предложена модель оценки выполнимости сформулированной программы, исследование которой сводится к сетевой задаче распределения ресурсов.

Новые области использования сетевых задач распределения ресурсов стимулируются развитием вычислительной техники и расширением сфер ее применения. В конце 60-х годов — начале 70-х годов в связи с появлением и технической реализацией идеи мультипроцессирования были сформулированы задачи построения оптимальных мультипроцессорных расписаний, направленные на максимальное использование ресурсов развитой ЭВМ. В последнее десятилетие в этой интенсивно расширяющейся области исследований были получены интересные теоретические результаты. В связи с интенсивной разработкой сложных АСУ самого различного назначения поставлены задачи оптимальной организации внутренних процессов систем обработки данных, оперирующих большими массивами информации. Речь идет о распределении ресурсов вычислительной системы (физических устройств ЭВМ, реентерабельных программ и т. д.) при обработке информационно-связанной совокупности программ. В отличие от мультипроцессорных расписаний, задачи планирования вычислений при решении задач АСУ имеют ряд особенностей. А именно, объекты планирования (прикладные программы АСУ) характеризуются большей детерминированностью поведения, сравнительно низким уровнем требований к быстродействию процессоров и высоким уровнем требований к каналам обмена, часто должны решаться в заранее заданные сроки.

Настоящая монография посвящена численным методам решения дискретных сетевых задач оптимального распределения ограниченных ресурсов. Структура ее следующая. Вводная глава I «Дискретные сетевые задачи оптимального распределения ресурсов» посвящена содержательному описанию указанных задач. Она предназначена как пользователям, желающим распознать и сформулировать «свои» задачи на языке теории оптимизации, так и специалистам по методам дискретной оптимизации, для которых важно понять особенности возникающих задач, с тем чтобы использовать их для повышения эффективности соответствующих численных методов.

В главе II «Теория сложности экстремальных комбинаторных задач и сетевые задачи распределения ресурсов и составления расписаний» после краткого экскурса в современную прикладную теорию сложности вычислений приводятся результаты, показывающие, что даже простейшие сетевые задачи теории расписаний по крайней мере столь же сложны, как и ряд других широко известных экстремальных комбинаторных задач (например, задача минимизации дизъюнктивных нормальных форм функций алгебры логики, задача определения максимального внутренне устойчивого множества вершин графа). Как известно, большинство встречающихся на практике оптимизационных задач входит в класс так называемых *NP*-полных задач. Теория *NP*-полноты основана на понятии полиномиальной сводимости задач. Основной практический вывод этой теории состоит в том, что если хотя бы одна из *NP*-полных задач имеет эффективный полиномиальный алгоритм, то остальные тоже могут быть решены посредством алгоритма с полиномиальной оценкой трудоемкости. Однако, имея очевидное теоретическое значение, сам факт *NP*-полноты некоторой новой задачи оптимизации практически мало что дает исследователю-разработчику эффективных алгоритмов, а тем более пользователю. «Хорошие», эффективные алгоритмы в новых сложных задачах большой размерности являются не столько следствием хороших формальных схем вычислений, сколько результатом искусства разработчика, математика-вычислителя и системного программиста, их умения использовать особенности задач в целях ускорения вычислений и просто их опыта. В еще большей степени это справедливо, когда речь идет о задачах дискретного программирования, каковыми являются сетевые задачи распределения нескладируемых ресурсов. Существующий опыт решения этих задач говорит о том, что, несмотря на все ухищрения, практические алгоритмы оптимизации, например алгоритмы ветвей и границ, имеют экспоненциальную трудоемкость. В связи с высокой трудоемкостью оптимизации в строгом смысле опре-

деленный интерес представляют методы приближенной оптимизации и, в частности, методы эвристического типа. Последние, как правило, имеют полиномиальную оценку трудоемкости, и основной интерес в их анализе представляет отыскание гарантированной величины погрешности по функционалу. В главе III «Вариации длительности кратчайших расписаний. Приближенные алгоритмы с оценкой погрешности» представлен ряд наиболее общих результатов такого рода.

Собственно методы и алгоритмы последовательной оптимизации представлены в главах IV—VI. В главе IV «Динамическое программирование» алгоритм динамического программирования иллюстрируется на примере сетевой задачи определения кратчайшего расписания, хотя та же общая схема пригодна и для функционалов более общего вида. Приводятся асимптотические оценки трудоемкости алгоритмов динамического программирования в экстремальных задачах на ациклических графах. В главе V «Последовательный анализ вариантов» алгоритм последовательного анализа вариантов иллюстрируется на примере задачи объемного календарного планирования с аддитивно-сепарабельной целевой функцией. Приводятся оценки трудоемкости в среднем алгоритмов последовательного анализа вариантов, основанные на аналогии результатов соответствующих процедур выбора со статистиками крайних значений. И наконец, наиболее обширная глава VI «Метод ветвей и границ» посвящена, повидимому, самому перспективному в практическом отношении методу решения задач рассматриваемого класса — методу ветвей и границ. Алгоритмы ветвей и границ обладают рядом структурных особенностей, которые делают их пригодными для решения задач большой размерности. Они позволяют строить приближенные эффективные алгоритмы с апостериорной оценкой погрешности. Как известно, качество алгоритмов этого класса существенно зависит от способов вычисления оценок (границ). Сетевые задачи оптимального распределения ресурсов обладают рядом особенностей, которые могут быть эффектив-

но использованы для вычисления оценок. Систематически изучены два направления, позволяющие эффективно вычислять соответствующие оценки. Одно из них использует специальные функции, определенные на сети операций. Второе основано на использовании современных идей теории двойственности и негладкой оптимизации в дискретном программировании. Здесь же приводятся некоторые результаты численных экспериментов.

Каждая глава завершается комментариями, в которых делаются ссылки на использованные источники и родственные результаты.

В выборе содержания монографии и ее разделов мы руководствовались главным образом двумя критериями. В выборе моделей задач мы стремились к наиболее общим формальным моделям оптимизационных задач, нашедшим отражение в специальной литературе, учитывая, что даже такие модели все еще существенно упрощают реальные проблемы. В выборе методов решения задач мы ориентировались на общие схемы упорядоченного перебора вариантов, названные нами методами последовательной оптимизации. Даже с такими ограничениями круг исследователей, в той или иной степени занимавшихся проблематикой монографии, все еще очень широк. И мы заранее благодарны тем из них, кто любезно укажет нам на наши упущения.

В. С. Михалевич, А. И. Кукса

ГЛАВА I

ДИСКРЕТНЫЕ СЕТЕВЫЕ ЗАДАЧИ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ

1.1. Элементы языка сетевых моделей

В очень общих терминах задачи, о которых будет идти речь в этой книге, заключаются в следующем. Задана некоторая совокупность действий, подлежащих выполнению. В ходе выполнения каждого действия загружаются или расходуются определенные средства. Как правило, оказывается, что общего уровня наличных средств недостаточно для одновременного выполнения всех действий. Требуется упорядочить процесс выполнения действий во времени таким образом, чтобы при соблюдении ряда дополнительных условий минимизировать заданную функцию потерь. Выполнение совокупности действий рассматривается как процесс, состояние которого изменяется во времени и может варьироваться в определенных пределах посредством манипулирования заданными средствами. Иными словами, порядок использования средств является управляющей переменной процесса, и речь идет об оптимальном использовании заданных средств во времени с учетом эффективности процесса в целом. Частные задачи указанного типа традиционно рассматривались в детерминированной теории расписаний. Для более точного описания общих задач уточним содержание используемых в дальнейшем терминов.

Время. Время является независимой переменной в описании процесса выполнения действий и их совокупностей. В дальнейшем главным образом будет рассматриваться случай, когда время принимает только целочисленные неотрицательные значения. Отметим, что абсолютное значение единицы измерения времени зависит от объекта планирования и может изменяться в очень широких пределах. Так, планирование внутренних процессов вычислительных систем может осуществляться с точностью до наносекунд, планирование промышленного производства — с точностью до суток, а планирование

национальных программ развития — с точностью до нескольких лет.

Ресурсы. Термин «ресурсы» обозначает средства для достижения результатов отдельных действий и их совокупностей. По ряду причин удобно раличать два типа ресурсов: складируемые ресурсы, называемые иногда ресурсами типа «материалы», и нескладируемые ресурсы, или ресурсы типа «мощность». Примерами ресурсов первого типа являются: топливо, сырье, полуфабрикаты, детали и т. п. В экономических приложениях среди ресурсов этого типа иногда выделяют стоимость как интегральный показатель, характеризующий затраты всех остальных видов ресурсов (складируемых и нескладируемых). Складируемые ресурсы непосредственно расходуются в процессе выполнения того или иного действия. Как правило, не будучи использованными в данный момент времени, они могут быть использованы в более поздние моменты времени. Обычно складируемые ресурсы задаются общим объемом или функцией поставок во времени.

Примерами нескладируемых ресурсов являются: трудовые ресурсы, приборы, машины и комплексы технических средств, рабочие помещения и т. п. В отличие от складируемых, нескладируемые ресурсы в процессе выполнения соответствующих действий сами не расходуются, а производят сами или в сочетании с другими ресурсами некоторый расходуемый фактор (машино-смены, например), не допускающий складирования. Таким образом, простои нескладируемого ресурса в определенном смысле связаны с потерями. Обычно нескладируемые ресурсы заданы как функции времени. По существу, отнесение конкретного вида ресурса к тому или иному типу во многих случаях определяется не физической природой этого ресурса, а способом его использования при реализации того или иного действия. Существенно отличными, однако, являются математические методы для решения экстремальных задач распределения складируемых и нескладируемых ресурсов.

Операции, работы и системы операций. Операция — это физическое действие или процесс, осуществляемый в заданное время посредством заданного набора ресурсов. Каждая операция рассматривается как элементарное, атомарное действие. Она описывается лишь в ее отношении к внешнему окружению, т. е. в форме указания требуемых ресурсов и ее результатов, в то время как

внутренние процессы операции не являются предметом анализа. Осуществление операции состоит в том, что на время ее выполнения операции предоставляются необходимые ресурсы. Характеризуя процесс обслуживания, в дальнейшем будем полагать, что выражения «ресурсы обслуживают операции», «ресурсы используются на операциях», «операции потребляют ресурсы» и т. п. имеют один и тот же смысл. Ресурсы, обслуживающие данную операцию, не используются в то же самое время на других операциях. Формально каждая операция может быть описана своим именем или номером, списком имен используемых ею ресурсов и их количеством, а также длительностью. Длительность операции может зависеть от абсолютного (календарного) времени ее начала.

Как правило, операции являются элементами более общих целенаправленных действий или их совокупностей. В зависимости от содержания такие целенаправленные совокупности операций называются работами или системами (комплексами) операций. Обычно операции представляют собой определенные звенья, этапы некоторых физических, технологических, производственных процессов. Часто оказывается, что совокупности операций, образующих работы или системы операций, зависимы в том смысле, что в соответствии с требованиями реального процесса, отображаемого данной системой операций, одни операции должны следовать во времени за другими. Иначе говоря, на множестве операций, образующих систему операций, определено бинарное отношение предшествования. В свою очередь, работы или системы операций могут объединяться в более общие, содержащие их системы или комплексы операций и т. д. Таким образом, объединение операций в некоторую систему связано с определенной интерпретацией их как целенаправленных совокупностей действий, имеющих общие результаты. Иногда для обозначения общности результатов отдельных групп операций используют термин «событие».

Расписания. Термин «расписание» используется для обозначения порядка выполнения операций или работ во времени. В случае непрерываемых операций расписанием операции является пара чисел (календарное время начала операции, длительность операции). Соответственно, расписание работы или системы операций достаточно описывать списком времен начала входящих в них операций и их длительностями. Расписание называется допустимым, если:

1. В каждый момент времени суммарные уровни использования ресурсов каждого вида по всем операциям, выполняющимся в данное время, не превосходят общих уровней наличия соответствующих ресурсов.

2. Выполнены технологические ограничения, т. е. для каждой пары «связанных» операций последующая начинается не ранее окончания предыдущей.

3. Выполнены специальные условия. Специальные условия чаще всего регламентируют сроки начала или завершения отдельных операций, работ, сроки достижения определенных событий, взаимное расположение во времени отдельных групп операций и т. п.

Цели и задачи планирования. Наиболее известные цели календарного планирования сводятся к следующим:

А. Простейшая цель состоит в минимизации общего времени выполнения всех операций заданной системы операций при заданных общих уровнях наличия ресурсов каждого типа и вида. Родственная экстремальная задача состоит в минимизации максимального общего уровня потребления ресурсов при условии, что сроки завершения всех операций не превосходят заданной величины.

Б. Во многих практических задачах требуется минимизировать некоторую неубывающую по срокам завершения отдельных операций аддитивно-сепарабельную функцию потерь.

В. Наиболее общий тип целей в задачах календарного планирования отражает два ряда факторов. Первый фактор учитывает условные потери вследствие того, что операции или события осуществляются в те или иные сроки; второй фактор учитывает потери вследствие того, что ресурсы в некоторые моменты времени используются слишком интенсивно или недостаточно интенсивно. Последний фактор особенно важен в случае трудовых ресурсов.

Таким образом, в наиболее общем виде сетевые задачи оптимального распределения (использования) ресурсов определяются указанием:

— системы взаимосвязанных операций или работ;

— номенклатуры и уровней наличия ресурсов во времени;

— перечня специальных ограничений;

— цели планирования

и состоят в построении некоторого допустимого расписания, на котором заданная функция потерь достигает минимального значения.

В следующих разделах данной главы будут охарактеризованы наиболее известные области применения задач указаного типа и конкретизированы соответствующие задачи.

1.2. Сетевое планирование с ограниченными ресурсами

В практике народного хозяйства системы сетевого планирования и управления (СПУ) используются как инструмент для решения организационных задач планирования и управления комплексами работ на основе построения, анализа и оптимизации сетевых моделей работ. В настоящее время имеется значительный опыт разработки и использования систем СПУ в различных областях народного хозяйства. Наиболее целесообразными областями использования СПУ являются:

а) Целевые разработки сложных систем, в которых принимают участие организации и предприятия различных ведомств, включая научно-исследовательские и опытно-конструкторские работы, проектирование, опытное производство, испытания и т. д.

б) Государственные, межведомственные и региональные программы, например, охраны окружающей среды, развития региона и т. д.

в) Основная деятельность таких организаций и предприятий, как ОКБ, НИИ, проектные институты, предприятия опытного и мелкосерийного производства.

г) Строительство и монтаж промышленных и гражданских объектов (заводов, электростанций, шахт, мостов, дорог, жилых и гражданских зданий и т. п.).

д) Подготовка и освоение производства новых видов продукции.

е) Реконструкция и ремонт промышленных и гражданских объектов.

ж) Подготовка и проведение крупных организационных мероприятий (съездов, конференций, кампаний по предупреждению стихийных бедствий).

з) Разведка и освоение новых месторождений полезных ископаемых.

и) Ремонт промышленного оборудования и средств транспорта.

к) Материально-техническое снабжение крупных промышленных и гражданских объектов.

Как отмечалось, общей основой всех систем СПУ является использование сетевых моделей сложных состав-

ных работ — комплексов работ. Работа (операция) — основной элемент комплекса. Наиболее важная характеристика работы связана с понятием ее объема. Реальный смысл понятия объема работы может быть весьма различным: трудоемкость в человеко-днях или машино-сменах, тонно-километры, продолжительность в единицах времени, стоимость и т. д. В выбранных единицах объем представляется чаще всего скалярной величиной W. Объем работы во многих случаях точно известен до начала ее выполнения; в других случаях объем зависит от условий реализации (например, трудоемкость земляных работ может зависеть от применяемых средств механизации, а также от природных условий, времени года и т. п.); наконец, этот объем может быть случайной величиной из-за присущей самой работе неопределенности (трудоемкость поискового научного исследования и т. п.). Так как в общем случае работа есть процесс, происходящий во времени, можно говорить об объеме работы $W(t)$, выполненном к моменту времени t. Величина $W(t)$ является характеристикой состояния работы в момент времени t и не обязательно совпадает с какой-либо мерой труда, затраченного на достижение этого состояния. Формально выполнение работы состоит в изменении $W(t)$ от 0 до W. Скорость изменения функции $W(t)$ есть скорость ведения данной работы. Можно говорить о скорости ведения работы в данный момент времени t или о средней скорости за данный промежуток времени. В общем случае скорость выполнения работы является функцией времени и зависит от ряда других факторов (интенсивность потребления ресурсов, метеорологические условия и т. д.). В наиболее простых и распространенных случаях скорость выполнения работы принимают постоянной; при этом основной характеристикой работы становится ее продолжительность, играющая в этом частном случае роль объема.

Потребность отдельной работы в складируемом ресурсе l-го вида может быть задана числом r^l, равным общему количеству этого вида ресурса (в натуральных единицах), необходимому для выполнения работы в полном объеме. Более полная информация может быть дана с помощью функции, выражающей потребное количество ресурса l-го вида в зависимости от состояния работы. Такая функция, называемая графиком потребления ресурса данной работой, не убывает с возрастанием объема и равна r^l, когда объем достигает W. Потребность отдель-

ной работы в нескладируемом ресурсе k-го вида обычно описывают функцией, выражающей интенсивность потребления этого вида ресурса в зависимости от состояния работы (так называемый график интенсивности потребления ресурса данной работой). В простых моделях интенсивности потребления всех видов нескладируемых ресурсов в процессе выполнения работы принимают постоянными.

Потребность в данном виде складируемого ресурса для выполнения комплекса работ в целом на весь период реализации комплекса или некоторой его части определяется как сумма потребностей отдельных работ в этом виде ресурса. Потребность в данном виде нескладируемого ресурса для выполнения комплекса работ в каждый момент времени равна сумме интенсивностей потребления ресурсов всеми работами, выполняемыми в этот момент времени. Потребность в этом виде ресурса на весь период выполнения работ комплекса описывается функцией, выражающей зависимость общей интенсивности потребления ресурса от времени (график общей интенсивности потребления ресурса). Планируемое наличие каждого вида ресурса обычно описывается некоторой функцией времени (график поставок для складируемых ресурсов, график наличия ресурса для нескладируемых ресурсов).

Центральное место в системах СПУ с ограниченными ресурсами занимают задачи оптимального распределения ресурсов и составления календарных планов выполнения работ и использования соответствующих ресурсов. В качестве примера рассмотрим одну из типичных задач этого класса.

Задано множество работ; τ_i — длительность работы i, $i = 1, \ldots, n$, и набор ресурсов; индексы $k = 1, \ldots, p$ используются для обозначения нескладируемых ресурсов, а $k = p + 1, \ldots, (p + q)$ — для обозначения складируемых ресурсов. Требуется, чтобы работы выполнялись без прерываний. В этом случае порядок выполнения работ во времени можно описать набором $\{t_i\}_{i=1}^{n}$, где t_i — календарная дата начала работы i.

Потребности работы i, $i = 1, \ldots, n$, в ресурсе k, $k = 1, \ldots, (p + q)$, в общем случае могут быть заданы как функции двух переменных:

$$r_i^k(t, t_i), \quad t, t_i \in [0, T].$$

С достаточной для практических целей общностью мож-

но полагать, что при фиксированном значении t_i функция $r_i^k(t, t_i)$ является неотрицательной, кусочно-постоянной и непрерывной слева для всех t внутри интервала $[t_i, t_i + \tau_i]$, а вне его $r_i^k(t, t_i)$ тождественно равна 0.

Описание комплекса работ помимо описания самих работ включает в себя информацию об отношении непосредственного предшествования, определенном на множестве работ. В наиболее распространенных случаях соответствующее бинарное отношение является антирефлексивным и транзитивным. В этом случае говорят, что заданная совокупность работ частично-упорядочена.

Цель оптимального планирования может состоять в минимизации функции условных потерь:

$$f(t_1, \ldots, t_n) = \sum_{i=1}^{n} f_i(t_i)$$

при заданных ограничениях на область допустимых значений вектора $(t_1, \ldots, t_n)$. В большинстве случаев достаточно полагать, что $f_i(t_i)$, $i = 1, \ldots, n$,— неотрицательные, монотонно неубывающие функции времени. Ограничения на общие уровни потребления ресурсов имеют вид

$$\sum_{i=1}^{n} r_i^k(t, t_i) \leqslant r_0^k(t), \quad k = 1, \ldots, p; \quad t = 1, \ldots, T, \tag{1.1}$$

для нескладируемых ресурсов и

$$\sum_{\tau=1}^{t} \sum_{i=1}^{n} r_i^k(\tau, t_i) \leqslant \sum_{\tau=1}^{t} r_0^k(\tau), \tag{1.2}$$

$$k = p + 1, \ldots, q; \quad t = 1, \ldots, T,$$

для складируемых ресурсов.

В ограничениях на общие уровни потребления нескладируемых ресурсов требуется, чтобы в каждый момент времени суммарные уровни потребления ресурсов по всем работам, выполняющимся в данный момент, не превосходили заданных верхних границ. С другой стороны, требуется, чтобы для каждого t общий расход складируемых ресурсов на выполнение всех работ, начатых и, возможно, завершенных в интервале $[0, t]$, не превышал общего количества этих ресурсов, накопленного к моменту времени t, начиная с 0.

Ограничения на порядок выполнения работ во времени имеют вид

$$t_i + \tau_i \leqslant t_j, \quad (i, j) \in E, \tag{1.3}$$

где E — множество пар работ, для которых выполнено отношение предшествования.

Примеры целевой функции:

$$f(t_1, \ldots, t_n) = \max_{1 \leqslant i \leqslant n} (t_i + \tau_i),$$
$$f(t_1, \ldots, t_n) = \sum_{i=1}^{n} \max(0, t_i + \tau_i - D_i), \tag{1.4}$$

где D_i — так называемый директивный срок завершения работы i, $i = 1, \ldots, n$. Таким образом, экстремальная задача сетевого планирования с ограниченными ресурсами состоит в минимизации (1.4) при условиях (1.1)—(1.3).

1.3. Программно-целевой метод управления и сетевые задачи распределения ресурсов

Программно-целевое управление — это метод целенаправленного долгосрочного планирования социально-экономических, народнохозяйственных, научно-технических и других систем высокого уровня и принятия научно обоснованных управляющих решений по использованию имеющихся ресурсов для достижения поставленных целей. В программно-целевом управлении используются элементы и средства системного анализа и исследования операций, сетевого планирования и управления, прогнозирования и экспертного оценивания, имитационного моделирования и ряда других научных дисциплин. В общем случае речь идет об итеративном процессе уточнения плановых решений с учетом взаимного влияния управляемой системы и ее внешней среды. Вначале осуществляется прогноз внешней обстановки и состояния системы в интервале планирования. Орган принятия решений на основе информации о внешней среде и внутреннем состоянии системы формулирует генеральную цель развития системы. Обсуждая способы достижения поставленной цели, соответствующие коллективы специалистов естественным образом формулируют подцели и промежуточные цели, от достижения которых зависит достижение общей цели. Таким образом, формируется иерархическая структура целей (граф целей).

Цели и задачи $(r+1)$-го уровня графа целей рассматриваются как средства для достижения целей и задач r-го и других верхних уровней иерархии. На одном из низших уровней иерархии подцели отождествляются с некоторыми действиями, выполнение которых эквивален-

тно достижению этих подцелей. Система действий, направленных на достижение некоторой цели, образует программу. Таким образом, каждой вершине графа целей соответствует программа. Программа-цель i r-го уровня иерархии включает в себя программы всех целей, расположенных на уровнях $(r+1)$, $(r+2)$ и т. д. и связанных путем с вершиной i.

Вершинам и дугам графа целей могут быть сопоставлены определенным образом веса, интерпретируемые как вероятности успеха в соответствующих задачах и направлениях. Предполагается, что указанные вероятности могут быть указаны экспертами. Веса, сопоставленные вершинам-целям, используются для обоснования размеров финансирования соответствующих направлений действий.

Пример иерархии целей. Для общегосударственных программ глобальная цель формулируется в достаточно общих выражениях, отражающих политические установки государства. Цели второго уровня могут отражать отдельные стороны деятельности государства: максимальное удовлетворение потребностей граждан, обеспечение безопасности, поддержание внешнеполитических связей. На следующем уровне возникают задачи развития промышленности, сельского хозяйства, науки и культуры. На уровне отраслей возникают задачи разработки долгосрочных программ их развития с точки зрения научно-технических, финансовых и ресурсных возможностей выполнения соответствующих программ. На одном из нижних уровней речь может идти о плане реконструкции отдельного предприятия или о выполнении научно-исследовательской работы.

После того как сформирована иерархическая система целей, могут быть рассмотрены способы достижения каждой цели. Одну и ту же цель можно достичь разными способами. Соответственно этому может быть предложено несколько альтернативных программ для достижения данной цели. В общественной системе программы формируются коллективами специалистов различного профиля — социологами, военными, экономистами, медиками, педагогами, инженерами, учеными и т. д.— на языке соответствующей отрасли знаний. В соответствующих терминах формулируются и работы, составляющие программы. В понятии работы естественно выделяются объект, над которым совершается некоторое преобразование, и условия, которые надо создать, чтобы совершить это преобразование. Программа является средством, с по-

мощью которого соответствующая система изменяет свое состояние, а часто и состояние внешней среды, в соответствии со своими целями в изменяющихся условиях.

Далее производится расчет планов выполнения отдельных программ. При этом определяются сроки выполнения работ, составляющих программу, и затраты ресурсов. Совокупность приемлемых планов выполнения программы образует альтернативы плановых решений. Одно из альтернативных решений утверждается как официальная программа. Оценка выполнимости (реализуемости) программы при заданных внешних условиях состоит в разработке плана функционирования и развития производства работ, составляющих данную программу. В условиях ограниченных ресурсов, выделенных для осуществления целей программы, возникает необходимость в рассмотрении специальных программ, целью которых является развитие самих производительных сил (производства), обеспечивающих необходимыми материальными ресурсами все программы. По терминологии Г. С. Поспелова, такие «обеспечивающие» программы называются программами второго рода, а основные «целевые» программы — программами первого рода. Программа развития производства также формулируется как система работ. Однако, в отличие от программ первого рода, в которых объемы работ заранее определены, общий объем работ в программах второго рода заранее неизвестен и должен определяться, исходя из целей основных программ. Совместной или обобщенной программой назовем программу, состоящую из пары программ, одна из которых является программой первого рода, а вторая — программой второго рода. По определению, система работ обобщенной программы включает в себя все работы основной и вспомогательной программ вместе с отношением предшествования, определенным на множестве всех работ. Все работы основной программы подлежат обязательному включению в план, в то время как работы вспомогательной программы подлежат включению в план лишь в той степени, в какой это необходимо для достижения целей обобщенной программы. Как и в теории сетевого планирования, предполагается, что задан интервал времени $[0, T]$, $T < \infty$,— интервал планирования. Задана система работ обобщенной программы, включающая множество работ вместе с отношением предшествования работ. Формально систему работ можно представить в виде ациклического графа (V, E) с множеством вершин-работ V и отношением предшествования

$E \subset V \times V$. Множество V конечно, $V = \{1, \ldots, i, \ldots, n\}$; каждой работе взаимно однозначно сопоставлено неотрицательное число τ_i, $0 < \tau_i < \infty$, называемое длительностью работы. Условимся для простоты, что начатые работы не должны прерываться. Множество E имеет следующую интерпретацию: если $(i, j) \in E$, то по технологическим соображениям работа j должна быть начата после завершения работы i или ее части. Формально каждой паре (i, j) из E взаимно однозначно сопоставлено неотрицательное число τ_{ij}, равное длине минимального отрезка времени между временами начала работ j и i. Выполнение работ или программ связано с потреблением ресурсов. Различаются складируемые и нескладируемые ресурсы; $k = 1, \ldots, K$ — номера нескладируемых видов ресурсов, $l = 1, \ldots, L$ — номера складируемых видов ресурсов. Отметим, что термин «ресурсы» в данном случае интерпретируется весьма широко: ресурсами являются сырье, комплектующие изделия (входные продукты), агрегаты, машины (конечные продукты), станки, оборудование, производственные единицы (бригады, цехи, заводы), деньги, помещения и т. д. Формальное отличие ресурсов разных типов определяется видом ограничений на порядок выполнения работ или групп работ. Интенсивность потребления ресурсов в процессе выполнения работы i определена вектор-функцией

$$r_i(\tau) = \{r_i^k(\tau),\ k = 1, \ldots, (K + L)\},$$

где $r_i^k(\tau)$, $0 \leqslant \tau \leqslant \tau_i$, $i = 1, \ldots, n$,— неотрицательные, кусочно-постоянные, непрерывные слева функции времени. Кроме того, будем полагать, что работы обобщенной программы обладают способностью пополнять ресурсы. Интенсивность пополнения ресурсов в процессе выполнения работы i характеризуется вектор-функцией $\delta r_i(\tau)$, которая определена аналогично функции $r_i(\tau)$, $i = 1, \ldots, n$.

Если работа i включена в план и начата в момент времени t_i, $0 \leqslant t_i \leqslant T - \tau_i$, то ее графики потребления ресурсов в интервале времени $[0, T]$ определены функциями

$$r_i^k(t \mid t_i), \quad i = 1, \ldots, n; \quad k = 1, \ldots, (K + L);$$

$$r_i^k(t \mid t_i) = \begin{cases} r_i^k(t - t_i), & \text{если} \quad t \in [t_i, t_i + \tau_i], \\ 0 & \text{в противном случае.} \end{cases}$$

Функции пополнения ресурсов $\delta r_i^k(t \mid t_i)$ определяются аналогично.

Поскольку работы непрерываемы, расписание каждой из них определено указанием момента времени t_i, когда работа начата. Расписание совокупности работ обобщенной программы обозначим через S и определим как кортеж из n чисел t_i^S:

$$S = (t_1^S, \ldots, t_i^S, \ldots, t_n^S).$$

Условимся, что $t_i^S = \infty$, когда работа i не включена в план, $i = 1, \ldots, n$. Если S задано, уровни потребления и пополнения ресурсов каждого вида на отдельных работах определены соответственно функциями $r_i(t \mid t_i^S)$ и $\delta r_i(t \mid t_i^S)$. Уровни потребления ресурсов для совокупности всех работ определяются вектор-функцией

$$r(t \mid S) = \sum_{i=1}^{n} r_i(t \mid t_i^S).$$

Аналогично определена вектор-функция $\delta r(t|S)$ пополнения ресурсов. Исходные ресурсы, выделенные для выполнения основных работ и развития производства, предполагаются заданными вектор-функцией

$$r_0(t) = (r_0^k(t),\ k = 1, \ldots, (K + L)), \qquad t \in [0, T].$$

Условимся, что $r_0^k(t)$ также являются неотрицательными, кусочно-постоянными и непрерывными слева. Таким образом, общие уровни наличия ресурсов в каждый момент времени $t \in [0, T]$ определены вектор-функцией

$$r_\Sigma(t|S) = r_0(t) + \delta r(t|S).$$

Для описания состояния программы и ее результатов удобно пользоваться так называемыми интегральными графиками. Интегральный график наличия ресурсов для каждого t определяется как вектор-функция

$$R_\Sigma(t \mid S) = \int_0^t r_\Sigma(\tau \mid S)\, d\tau.$$

В свою очередь, интегральный график потребления ресурсов в процессе выполнения работ определен вектор-функцией

$$R(t \mid S) = \int_0^t r(\tau \mid S)\, d\tau.$$

В достаточно общем случае цель оптимального планирования может быть определена в терминах сроков начала или завершения работ и достигнутых уровней производства или объемов производства. Соответствующая целевая функция, определенная на множестве расписаний, может иметь вид $f(S)$ или $\varphi(S)$ или их комбинации:

$$f(S) = \sum_{i=1}^{n} f_i\left(t_i^S\right), \quad \varphi(S) = \sum_{t=1}^{T} \varphi_t\left(R_\Sigma(t \mid S)\right).$$

Здесь $f_i\left(t_i^S\right)$, $i = 1, \ldots, n$ (t_i^S — скалярные переменные),— функции условных потерь, зависящие от сроков начала (или завершения) отдельных работ обобщенной программы, а $\varphi_t(R_\Sigma)$, $t = 1, \ldots, T$ (R_Σ — векторная переменная размерности $(K+L)$),— функции условных потерь, зависящие от достигнутых в каждый период времени уровней производства конечной продукции каждого вида. Как указывалось, термин «условные потери» может иметь самое различное содержание: плата за несогласованность с ранее намеченными целями, потери продукции в натуральных единицах и т. п. С другой стороны, формальные свойства функций f_i и φ_t важны с точки зрения возможностей их использования в алгоритмах решения соответствующих задач оптимизации. Во многих случаях достаточно полагать, что $f_i\left(t_i^S\right)$, $i = 1, \ldots, n$, — монотонно неубывающие, а $\varphi_t(R_\Sigma)$, $t = 1, \ldots, T$,— линейные функции.

Расписание выполнения системы работ обобщенной программы называется допустимым, если оно удовлетворяет ряду условий. Условия предшествования работ

$$t_i^S + \tau_{ij} \leqslant t_j^S \tag{1.5}$$

определены для всех пар работ (i, j), связанных отношением предшествования E. Ресурсные ограничения поразному формулируются для складируемых и нескладируемых видов ресурсов. Для складируемых видов ресурсов в каждый момент времени должны выполняться условия

$$R(t|S) - R_\Sigma(t|S) \leqslant 0. \tag{1.6}$$

Соответственно, для нескладируемых видов ресурсов должны выполняться условия

$$r(t|S) - r_\Sigma(t|S) \leqslant 0 \tag{1.7}$$

для всех $t = 1, \ldots, T$.

С учетом сказанного сформулируем несколько задач оптимального планирования обобщенных программ.

Задача 1 заключается в минимизации целевой функции $f(S)$ при условиях (1.5)—(1.7).

Заметим, что она является естественным обобщением задач оптимального распределения ресурсов, возникающих в теории сетевого планирования и управления. По сравнению с задачами СПУ здесь допускается выбор работ, подлежащих выполнению, а также возможность пополнения ресурсов в процессе выполнения работ.

Задача 2 заключается в минимизации целевой функции $\varphi(S)$ при условиях (1.5)—(1.7), а также дополнительных условиях вида

$$t_i^S \leqslant \bar{t}_i, \quad i = 1, \ldots, n, \tag{1.8}$$

где $\bar{t}_i$ — заданные директивные поздние сроки начала работ или групп работ (событий). Ограничения (1.8) могут порождаться внешними по отношению к модели событиями типа «ввод в действие предприятия». Задача 2 покрывает такие области применения, где основной интерес представляют объемы выпуска продукции, в то время как сроки выпуска учитываются лишь как ограничения задачи.

Заметим, что задачи 1 и 2 далеко не покрывают обширные области применения задач такого рода. В определенных выше терминах может быть сформулирован и ряд других общих задач оптимального планирования. Например, определенный практический смысл имеет задача минимизации целевой функции $f(S)$ при условиях (1.5)—(1.7) и условиях

$$R_\Sigma(t|S) \geqslant \underline{R}_\Sigma(t), \quad r_\Sigma(t|S) \geqslant \underline{r}_\Sigma(t),$$

где $\underline{R}_\Sigma(t)$, $\underline{r}_\Sigma(t)$ — вектор-функции, определяющие соответственно необходимые уровни производства конечной продукции во времени и уровни развития производственных мощностей.

1.4. Детерминированные задачи оптимального планирования в параллельных вычислительных системах

В последние годы в научной литературе по вычислительной технике и математической кибернетике большое внимание уделяется вычислительным системам высокой производительности, способным выполнять параллельные

вычисления. Условимся называть их параллельными вычислительными системами (ПВС). Производительность вычислительных систем такого класса в значительной степени определяется эффективностью использования их внутренних ресурсов. Возникающие при этом задачи оптимизации использования ресурсов исключительно сложны, не формализованы еще в виде известных моделей математического программирования и решаются разработчиками соответствующих вычислительных систем на инженерном, эвристическом уровне. Тем не менее некоторые частные задачи оптимизации использования ресурсов развитых ЭВМ в настоящее время уже сформулированы и могут быть удовлетворительно решены современными методами оптимизации. Как правило, эти задачи охватывают или отдельные внутренние процессы ЭВМ без их взаимной связи, или весьма упрощенно описывают вычислительный процесс в целом. Ниже будут рассмотрены некоторые из таких задач. Прежде чем формулировать их, отметим, что в современных исследованиях по теории операционных систем, реализующих функции планирования и управления ресурсами ЭВМ, обозначились два принципиальных подхода, определяющие как математические средства анализа возникающих ситуаций, так и применимость соответствующих результатов анализа. Речь идет о стохастическом и детерминированном подходах. В первом случае внутренние процессы ЭВМ описываются посредством известных стохастических моделей: марковских и полумарковских управляемых случайных процессов, моделей теории массового обслуживания и т. п. Во втором случае делается попытка детерминированного описания соответствующих процессов и решений в форме известных моделей теории расписаний. Принципиальное отличие указанных подходов заключается в следующем.

Во втором случае делаются предположения о том, что времена решения заданий ЭВМ известны заранее и что все задания одновременно готовы к выполнению. В первом же случае о тех же параметрах могут быть сделаны лишь вероятностные предположения. Далее речь будет идти о некоторых задачах второго направления.

Рассмотрим упрощенную модель составления многопроцессорных расписаний. Она состоит из трех основных компонент: процессоры, ресурсы, задачи. Все процессоры являются идентичными, каждый может выполнять самое большее одну задачу в каждый момент времени, имеется

$r_0^0 < \infty$ процессоров. Множество

$$P = \{1, \ldots, k, \ldots, K\}$$

ресурсов также конечно, и для каждого ресурса k задана граница r_0^k, равная общему количеству этого ресурса, имеющемуся в каждый момент времени. Задачи образуют третье конечное множество

$$V = \{1, \ldots, i, \ldots, n\},$$

все элементы которого должны быть выполнены процессорами при следующих ограничениях.

Во-первых, с каждой задачей связано положительное число τ_i — время, требуемое для выполнения i на любом из процессоров. Если некоторый процессор начинает выполнение задачи i в момент времени t_i, он завершает этот процесс в момент времени $(t_i + \tau_i)$, а затем может выполнять другие задачи.

Во-вторых, на множестве V определено бинарное отношение $\prec$, которое должно быть выполнено во время выполнения операций из V следующим образом: если $i \prec j$, то процесс выполнения задачи i должен быть завершен до того, как начнется процесс выполнения задачи j. Частичный порядок удобно описывать ориентированным графом без петель и контуров с множеством вершин V, в котором имеются дуги (i, j) тогда и только тогда, когда $i \prec j$. И наконец, для каждого ресурса k и задачи i задано неотрицательное число r_i^k, равное количеству ресурса k, требуемому задачей i в процессе ее выполнения. Общее ограничение на процесс выполнения совокупности всех задач состоит в том, что ни одно подмножество задач не может выполняться одновременно, если сумма их потребностей в каком-либо ресурсе k превосходит r_0^k, общее количество этого ресурса. Будем предполагать, что все τ_i, r_0^0 и r_i^k являются целыми числами.

Пусть задано целое положительное число D. Функция

$$S\colon V \to \{0, 1, \ldots, (D-1)\},$$

сопоставляющая каждой задаче i время t_i^S ее начала, определяет допустимое расписание, если она удовлетворяет следующим условиям:

1) для каждого $i \in V$ $t_i^S + \tau_i \leqslant D$;
2) для всех $i, j \in V$, если $i \prec j$, то

$$t_i^S + \tau_i \leqslant t_j^S;$$

3) для каждого целого числа t, $0 \leqslant t < D$, множество

$$V_S(t) = \{i \in V \mid t_i^S \leqslant t \leqslant t_i^S + \tau_i\}$$

удовлетворяет условию $|V_S(t)| \leqslant r_0^0$;

4) для каждого целого числа t, $0 \leqslant t < D$, и каждого k, $1 \leqslant k \leqslant K$,

$$\sum_{i \in V_S(t)} r_i^k \leqslant r_0^k.$$

Множество $V_s(t)$, определенное в условии 3), называется множеством задач, выполняемых в момент времени t в соответствии с расписанием S.

В теоретических работах наиболее часто встречаются следующие функции цели:

— минимизировать длительность расписания, т. е. найти минимальное D, при котором выполняются условия 1) — 4);

— минимизировать среднее время выполнения всех задач

$$f(S) = \sum_{i=1}^{n} c_i t_i^S,$$

где весовые коэффициенты c_i отражают важность задачи в том или ином практическом смысле;

— минимизировать максимальную или суммарную задержку времен завершения задач по сравнению с заданными директивными сроками:

$$f(S) = \sum_{i=1}^{n} \max(0, D_i - t_i^S - \tau_i),$$

где D_i, $i = 1, \ldots, n$,— директивные сроки завершения задач.

Таким образом, дискретные сетевые задачи оптимального распределения ограниченных ресурсов возникают на различных уровнях планирования и имеют однотипные постановки. В каждой задаче необходимо определенным образом сопоставить элементам одного заданного базового множества (работ) элементы другого базового множества (периодов времени), соблюдая при этом ряд условий. Будучи рассмотренными как задачи математического программирования, они имеют ряд характерных особенностей. Во-первых, это ограничения целочисленности переменных. Во-вторых, это присутствие в ограничениях задачи ресурсных (общих) и сетевых (специаль-

ных) ограничений. Специфика сетевых ограничений может быть эффективно использована с целью совершенствования алгоритмов решения задач. Примеры такого рода можно найти в главе VI.

Библиографический комментарий

В данной главе указаны лишь три наиболее важные области применения дискретных сетевых задач оптимального распределения ограниченных ресурсов: сетевое планирование и управление (СПУ), программно-целевое планирование, планирование вычислительного процесса параллельных вычислительных систем. Каждая из указанных областей весьма обширна и является предметом исследования в многочисленных публикациях.

В своей трактовке содержания задач сетевого планирования с ограниченными ресурсами мы следовали нормативной работе [1]. Проблематика СПУ начала интенсивно развиваться в СССР в 60-е годы. В 1967 году состоялась 1-я Всесоюзная конференция по математическим методам СПУ [2]. В дальнейшем развитие получили два теоретических направления исследований. Первое из них связано с использованием в задачах СПУ статистических методов [3]. Второе направление связано с трактовкой сетевых задач оптимального распределения ресурсов как задач теории оптимального управления [4 — 8]. Типичные модели задач этого типа — это задачи со складируемыми ресурсами, делимыми (прерываемыми) операциями, выпуклыми или линейными функциями потребностей операций в ресурсах и другими упрощающими предположениями [9, 10]. У потребителей сетевых задач оптимального распределения нескладируемых ресурсов широко используются различного рода эвристические процедуры решения задач [11, 12]. Собственно численным методам в задачах оптимального распределения нескладируемых ресурсов СПУ посвящено сравнительно небольшое число публикаций. Некоторые возможности использования математических методов оптимизации, а также результаты численных экспериментов приведены в работах [13 — 16].

В своем толковании проблематики программно-целевого планирования мы следовали основополагающим работам [17, 18]. Более расширенное освещение вопроса можно найти в [19]. Новые результаты в этой области содержатся в [20]. Отметим, что в общем комплексе идей программно-целевого планирования задачи оптимального распределения ресурсов играют вспомогательную роль, и как адекватные постановки задач, так и методы их решения находятся в стадии исследований.

Использованию математических моделей и методов в задачах планирования вычислительного процесса посвящено большое число публикаций. Отметим ряд отечественных монографий [21 — 25]. Новый стимул исследованиям в этом направлении дало появление вычислительных машин и систем, способных осуществлять параллельные вычисления [26]. Формулируя детерминированную задачу составления оптимального мультипроцессорного расписания, мы следовали [27 — 29].

В заключение отметим, что все рассмотренные в данной главе задачи являются обобщением задач теории расписаний [30].

ГЛАВА II

ТЕОРИЯ СЛОЖНОСТИ ЭКСТРЕМАЛЬНЫХ КОМБИНАТОРНЫХ ЗАДАЧ И СЕТЕВЫЕ ЗАДАЧИ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ И СОСТАВЛЕНИЯ РАСПИСАНИЙ

2.1. Элементы прикладной теории сложности

О сложности некоторой задачи имеет смысл говорить как о сложности алгоритмов, предназначенных для ее решения. Понятие сложности алгоритма чаще всего ассоциируется с его трудоемкостью, т. е. с количеством условных операций, необходимых для решения задачи. Традиционный способ анализа трудоемкости алгоритмов — эмпирический, т. е. многократное решение задачи на выборке ее входных данных и эмпирическая оценка интересующих исследователя величин и зависимостей. Заметим, что, как правило, практические задачи оптимизации являются существенно многопараметрическими и для них характерны большие объемы входных данных. Основной недостаток экспериментального подхода в изучении трудоемкости алгоритмов — это невоспроизводимость результатов эксперимента: нет достаточной уверенности в том, что результаты эксперимента достигаются на множестве входных данных, отличных от изученных. Для более обоснованного сравнения и выбора алгоритмов решения задачи необходима более развитая система понятий, характеризующих сложность алгоритмов и задач. Чтобы быть практически полезной, информация о сложности алгоритма должна быть машинно-независимой: практически хорошие (нетрудоемкие) алгоритмы являются таковыми независимо от того, в каких языках программирования они записаны или на каких машинах идет счет. С другой стороны, термины для описания сложности должны открывать определенные возможности для теоретического анализа сложности и получения соответствующих оценок.

Различаются меры сложности алгоритмов двух видов: статическая сложность, не зависящая от входных данных задачи, и динамическая сложность, зависящая от входных данных задачи. Типичной мерой статической

сложности является длина программы (определяемая, например, количеством операторов входного языка программирования). С другой стороны, длина программы в указанном смысле является и мерой простоты. Эта мера имеет наибольший практический смысл в тех случаях, когда время программирования значительно или когда программа используется (считается) нечасто. Динамическая мера сложности алгоритма дает информацию о потребностях алгоритма в ресурсах ЭВМ как функции входных данных задачи. Типичной динамической мерой сложности являются время вычислений и объем памяти, необходимой для реализации вычислений. Динамические меры сложности подходят в тех случаях, когда программа часто используется или когда потребности в ресурсах ЭВМ высоки.

Следует заметить, что различные меры сложности по-разному полезны в различных условиях применения. К тому же в реальных алгоритмах оценки времени вычислений и объема памяти алгоритма, как правило, не независимы: сокращение одной из них достигается нередко за счет возрастания другой. Временна́я сложность обычно является наиболее важным фактором, ограничивающим размеры задач, которые могут быть решены на современных вычислительных машинах. Далее будет идти речь главным образом об этой мере сложности.

Для данной задачи временна́я сложность — это функция, отображающая размерность задачи во время, требуемое для решения задачи. Применительно к этой простой мере сложности имеет смысл рассматривать нижние и верхние границы сложности, оценки сложности в среднем, асимптотические оценки сложности, модель вычислений и способы представления задачи. Способ представления задачи определяет размерность пространства, элементами которого являются входные данные задачи (алгоритма).

Обычно мы интересуемся внутренней сложностью вычислений в задаче определенного класса. Например, мы хотим знать, как быстро можно упорядочить список из n объектов, вначале расположенных в произвольном порядке — независимо от используемого алгоритма. В этом случае мы ищем нижнюю границу $L(n)$, являющуюся характеристикой задачи, а не алгоритма сортировки. Функция $L(n)$ означает, что ни один алгоритм не может решить задачу сортировки менее чем за $L(n)$ единиц времени для произвольного списка объектов, т. е. каждый

алгоритм сортировки должен работать по меньшей мере $L(n)$ единиц времени в худшем случае. С другой стороны, мы можем пожелать узнать, как долго известные алгоритмы осуществляют сортировку в худшем случае. Получим верхнюю границу $U(n)$, которая говорит о том, что мы можем упорядочить произвольный входной набор данных (произвольный исходный список из n объектов) не более чем за $U(n)$ единиц времени. При нашем уровне знаний нас не устроит алгоритм сортировки, работающий более чем $U(n)$ единиц времени, поскольку нам уже известен алгоритм, дающий результат за время $U(n)$. Поэтому алгоритмы чаще всего анализируются на предмет их поведения в худшем случае: мы надеемся уменьшить $U(n)$, показав, что новый алгоритм имеет «характеристику в худшем» лучше, чем каждый из известных ранее алгоритмов.

Пытаясь улучшить нижнюю границу, мы концентрируем свое внимание на методах, позволяющих повысить точность, с которой трудоемкость всех возможных алгоритмов может быть ограничена снизу. Улучшение верхней границы означает нахождение алгоритма с лучшей «характеристикой в худшем». Конечная цель исследования алгоритмов в некоторой задаче состоит в построении оптимального алгоритма, для которого имеет место $L(n) = U(n)$. В настоящее время для большинства задач, в том числе для задач, рассматриваемых в данной монографии, эта цель еще не достигнута.

Динамические меры сложности требуют точного определения входных данных задач (алгоритмов). Предположение о том, что входные данные являются худшими из возможных, является одной из исходных посылок в оценивании $U(n)$. Однако ориентация на худший случай может приводить к очень пессимистическим оценкам сложности. Например, симплекс-метод линейного программирования имеет экспоненциальную верхнюю оценку сложности, в то же время на практических задачах он почти всегда работает хорошо, т. е. во многих случаях время счета для него много меньше экспоненциальных оценок. Аналогичная ситуация типична в других задачах оптимизации. В таких случаях кажется практически более правдоподобным судить об алгоритмах (оценивать алгоритмы) по оценкам сложности в среднем.

В некоторых задачах анализ сложности в среднем почти всегда более реалистичен, чем анализ поведения алгоритмов в худшем случае. Однако и у этого подхода

имеются свои недостатки. Часто очень трудно выбрать подходящую вероятностную меру на множестве входных данных. Например, предположение о том, что различные части входных данных распределены независимо, делает анализ проще. Однако уже очень простые алгоритмы преобразования данных «разрушают» это предположение, усложняя дальнейший анализ. Во всех случаях, даже если удается оценить сложность алгоритма в среднем, остается риск появления входного набора данных, на котором сложность алгоритма намного превзойдет эту оценку.

В сложных задачах получение точных оценок, как верхних и нижних, так и в среднем, затруднительно. Иногда оказывается удобным при анализе алгоритмов подчитывать только количество «элементарных» операций, а о сложности алгоритмов судить по асимптотическому росту числа элементарных операций. Асимптотические оценки сложности полезны и потому, что многие практические задачи действительно имеют большую размерность. Асимптотические оценки сложности зависят от удачного выбора доминирующей в исследуемом алгоритме операции и с точностью до константы-сомножителя и членов более низкого порядка совпадают с точными оценками трудоемкости. Отметим, что асимптотические оценки сложности алгоритмов имеют особое значение в связи с ускоренным ростом быстродействия современных ЭВМ и возможностью решения задач большой размерности, характерных для экономики, управления в технических системах и т. д.

Как отмечалось, время вычислений является основным фактором, определяющим максимальные размеры практических задач, успешно решаемых посредством ЭВМ. Многие задачи могут быть решены сравнительно легко с незначительными затратами машинного времени (например, упомянутые выше задачи сортировки). В практике вычислений встречается, однако, и большое число важных задач, точное решение которых невозможно даже на самых мощных современных ЭВМ, т. е. требует практически недостижимых затрат машинного времени. Примерами являются многие экстремальные комбинаторные задачи. Соответственно, в теории вычислений различают две группы задач: это задачи, сложность которых оценивается полиномом от размерности задачи, и задачи с экспоненциальной оценкой сложности. Качественное отличие указанных групп задач иллюстрируется

следующей таблицей, в которой приводится условное время вычислений (1 шаг = 1 мкс) как функция n — размерности задачи (см. табл. 2.1).

Каждая мера сложности алгоритмов неявно предполагает существование вычислителя и некоторую модель

Таблица 2.1

Полиномиальный и экспоненциальный порядок сложности

Оценка сложности	Размерность n					
	20	50	100	200	500	1000
$100n^2$	0,04 с	0,25 с	1 с	4 с	25 с	2 мин
2^n	1 с	35 лет	$3 \cdot 10$ веков			

вычислений. Например, модель машины с произвольным доступом к памяти является абстракцией многих существующих ЭВМ общего назначения и часто используется как средство анализа алгоритмов. Другой широко используемой моделью вычислений является машина Тьюринга.

2.2. Простейшая модель вычислений: машины Тьюринга

Многоленточная машина Тьюринга (МТ) состоит из нескольких, например k, лент, бесконечно простирающихся вправо. Ленты разбиты на клетки, каждая из которых содержит один символ; число символов на каждой ленте конечно. Одна клетка на каждой ленте обозревается головкой этой ленты; головка может считывать с ленты и записывать на нее. Работа МТ определяется простой программой, называемой управляющим устройством. Оно всегда находится в одном из конечного числа состояний, которое можно рассматривать как номер текущей команды в программе. Один шаг вычисления на МТ состоит в следующем. В соответствии с текущим состоянием управляющего устройства и символами на лентах, обозреваемыми каждой из головок, МТ выполняет одну или несколько из следующих операций:

1) изменить состояние управляющего устройства;

2) записать на лентах новые символы вместо старых в каких-нибудь (или во всех) клетках под головками;

3) сдвинуть какие-нибудь (или все) головки независимо друг от друга на одну клетку влево (L), или вправо (R), или оставить на месте (S).

Формально k-ленточная детерминированная машина Тьюринга (ДМТ) задается семеркой

$$(Q, T, I, \delta, b, q_0, q_f),$$

где Q — множество состояний; T — множество символов на ленте; I — множество входных символов, $I \subseteq T$; b — пустой символ, $b \in T \backslash I$; q_0 — начальное состояние, $q_0 \in Q$; q_f — заключительное (допускающее) состояние, $q_f \in Q$; δ — функция переходов:

$$\delta\colon Q \times T^k \to Q \times (T \times \{L, R, S\})^k,$$

т. е. по произвольному набору из состояния и k символов на лентах МТ выдает новое состояние и k пар, каждая из которых состоит из нового символа на ленте и направления сдвига головки.

Пусть, например, машина находится в состоянии $q \in Q$, а ее головка на i-й ленте обозревает символ $a_i \in T$, $i = 1, 2, \ldots, k$, и $\delta(q, a_1, \ldots, a_k) = (q', (a_1', d_1), (a_2', d_2), \ldots, (a_k', d_k'))$, $q' \in Q$, $a_i' \in T$, $d_i \in \{L, R, S\}$, $i = 1, \ldots, k$. Тогда за один шаг эта машина Тьюринга переходит в состояние q', заменяет a_i на a_i' и сдвигает головку на i-й ленте в направлении d_i.

Машину Тьюринга можно приспособить для распознавания языков. Символы на лентах такой машины включают алфавит рассматриваемого языка (входные символы), пустой символ b и, возможно, другие символы. Вначале на первой ленте, начиная с самой левой клетки, записано слово (цепочка) из входных символов. Все клетки справа от клеток, содержащих входное слово, пусты. Все остальные ленты целиком пусты. Слово из входных символов допускается (воспринимается) тогда и только тогда, когда машина Тьюринга, начав работу в выделенном начальном состоянии (головки при этом находятся на левых концах своих лент), сделает последовательность шагов, которые в конце концов приведут ее в допускающее состояние. Языком, допускаемым данной машиной Тьюринга M, называется множество $L(M)$ всех слов из входных символов, допускаемых в описанном только что смысле.

Работу МТ формально можно описать с помощью мгновенных описаний. Мгновенным описанием (МО) k-ленточной машины Тьюринга M называется набор $D = (\alpha_1, \ldots, \alpha_k)$, где α_i для каждого i представляет собой слово вида xqy, причем xy — слово на i-й ленте машины

M (пустые символы, стоящие справа от его правого конца, опускаются), а q — текущее состояние машины. Головка на i-й ленте обозревает символ, стоящий справа от q. Если мгновенное описание D_1 переходит в мгновенное описание D_2 за один шаг машины Тьюринга M, то пишут $D_1 \underset{M}{\vdash} D_2$. Если $D_1 \underset{M}{\vdash} D_2 \underset{M}{\vdash} \ldots \underset{M}{\vdash} D_n$ для некоторого $n \geqslant 2$, то пишут $D_1 \underset{M}{\vdash}{}^{+} D_n$. Если либо $D' = D''$, либо $D' \underset{M}{\vdash}{}^{+} D''$, то пишут $D' \underset{M}{\vdash}{}^{*} D''$. Данная k-ленточная машина Тьюринга $M = (Q, T, I, \delta, b, q_0, q_f)$ допускает слово $a_1 \ldots a_n$, где $a_i \in I$, если $(q_0 a_1 a_2 \ldots a_n, q_0, q_0 \ldots q_0) \underset{M}{\vdash}{}^{*} (\alpha_1, \alpha_2, \ldots, \alpha_k)$, для некоторых α_i, содержащих q_f.

В дополнение к естественной интерпретации машины Тьюринга как устройства, допускающего какой-то язык, ее можно рассматривать как устройство, вычисляющее некоторую функцию φ. Аргументы этой функции кодируются на входной ленте в виде слов x со специальным маркером #, отделяющим их друг от друга. Если машина Тьюринга останавливается, имея на ленте, выделенной в качестве выходной, целое число y, то полагают $\varphi(x) = y$. Таким образом, процесс вычисления мало отличается от процесса допускания языка. Временна́я сложность $T(n)$ машины Тьюринга M равна наибольшему числу шагов, сделанных ею при обработке входа длины n (для всех входов длины n). Если на каком-нибудь входе длины n машина Тьюринга не останавливается, то для этого n значение $T(n)$ не определено.

k-ленточной недетерминированной машиной Тьюринга (НМТ) M называется семерка

$$(Q, T, I, \delta, b, q_0, q_f),$$

где смысл всех компонент тот же, что и для детерминированной машины Тьюринга, за исключением функции переходов δ, которая представляет собой отображение множества $Q \times T^k$ в множество подмножеств множества $Q \times (T \times \{L, R, S\})^k$. Иными словами, по данному состоянию и списку из k ленточных символов функция δ выдает конечное множество вариантов следующего шага; каждый вариант состоит из нового состояния, k новых ленточных символов и k сдвигов головок. Заметим, что НМТ M может выбрать любой из этих вариантов шага, но не может выбрать следующее состояние из одного шага, а новые ленточные символы — из другого, или сделать еще какую-нибудь комбинацию вариантов шага. Мгновенные описа-

ния НМТ определяются точно так же, как и для детерминированной машины Тьюринга. Данная НМТ $M = (Q, T, I, \delta, b, q_0, q_f)$ делает шаг, исходя из своего текущего состояния q и символов $x_1, \ldots, x_k$, обозреваемых каждой из k головок. Из множества $\delta(q, x_1, \ldots, x_k)$ она выбирает некоторый элемент $(r, (y_1, d_1), \ldots, (y_k, d_k))$. Этот элемент указывает, что новым состоянием должно стать r, на i-й ленте следует записать y_i вместо x_i и головку сдвинуть в направлении d_i, $i = 1, \ldots, k$. Если при некотором выборе следующего шага мгновенное описание C переходит в мгновенное описание D, пишут $C \underset{M}{\vdash} D$ (или $C \vdash D$, если ясно, о какой машине идет речь). Заметим, что для данной НМТ M может быть несколько таких D, что $C \vdash D$, но если машина M детерминирована, то для каждого C существует не более одного D. Запись $C \vdash^* D$ означает, что для некоторого n выполняется $C = C_1 \vdash C_2 \vdash \ldots \vdash C_n = D$ или $C_1 = D$. Говорят, что НМТ M допускает цепочку w, если $(q_0 w, q_0, q_0, \ldots, q_0) \vdash^* \vdash^* (\alpha_1, \ldots, \alpha_k)$, где все α_i содержат символ заключительного состояния q_f. Языком, допускаемым машиной M, называется множество цепочек, допускаемых машиной M. Говорят, что НМТ M имеет временну́ю сложность $T(n)$, если для всякой допустимой входной цепочки длины n найдется последовательность не более чем из $T(n)$ шагов, которая приводит в допускающее состояние.

При данной входной цепочке x можно считать, что недетермипированная машина Тьюринга M параллельно выполняет все возможные последовательности шагов, пока не достигнет допускающего МО или пока не окажется, что дальнейшие шаги невозможны. Иначе говоря, после i шагов можно считать, что существует много экземпляров M. Каждый экземпляр представляет МО, в котором M может оказаться после i шагов. На $(i+1)$-м шаге экземпляр C порождает j своих копий, если НМТ с соответствующим МО может выбрать следующий шаг j способами. Таким образом, возможные последовательности шагов НМТ M с входом x можно представить в виде дерева мгновенных описаний. Каждый путь из корня в лист этого дерева представляет некоторую последовательность шагов. Если σ — кратчайшая последовательность шагов, оканчивающаяся допускающим МО, то, как только M сделает $|\sigma|$ шагов, она остановится и допустит вход x. Время, затрачиваемое на обработку входа x, равно длине последовательности σ, т. е. $|\sigma|$. Если на входе x никакая

последовательность шагов не приводит к допускающему МО, то M отвергает x, а время обработки x не определено. Часто удобно представлять себе, что НМТ M угадывает только шаги из последовательности σ и проверяет, что σ на самом деле оканчивается допускающим МО. Но поскольку детерминированная машина обычно не может заранее угадать допускающую последовательность шагов, то детерминированное моделирование НМТ M потребовало бы просмотра в некотором порядке дерева всех возможных последовательностей шагов на входе x и этот просмотр продолжался бы до обнаружения кратчайшей последовательности, оканчивающейся допускающим МО. Если никакая последовательность шагов не приводит к допусканию входа, то детерминированное моделирование НМТ M могло бы продолжиться бесконечно долго, если нет какой-нибудь априорной границы на длину кратчайшей допускающей последовательности. Поэтому естественно ожидать, что НМТ способны выполнять задания, которые никакие ДМТ не могут выполнить за то же время. Тем не менее все еще открыт важный вопрос о существовании языков, допускаемых НМТ с данной временной сложностью, но не допускаемых никакой ДМТ с той же сложностью.

2.3. Классы P и NP, языки и задачи

Рассмотрим два важных класса языков. Определим P как множество языков, допускаемых детерминированными машинами Тьюринга с полиномиальной временной сложностью, т. е. $P = \{L |$ существуют такие ДМТ M и полином $p(n)$, что временная сложность машины M равна $p(n)$ и $L(M) = L\}$. Аналогично, пусть NP — это множество языков, допускаемых недетерминированными машинами Тьюринга с полиномиальной временной сложностью.

Можно показать, что если язык L принадлежит NP, то он допускается некоторой детерминированной машиной Тьюринга с временной сложностью $c^{p(n)}$, где c — постоянная, а p — полином, зависящие от L. С другой стороны, неизвестно, существует ли язык из NP, не принадлежащий P, т. е. является ли P собственным подмножеством NP. Однако можно показать, что некоторые языки не менее трудно распознать, чем любой язык из NP, в том смысле, что существование детерминированного по-

линомиального алгоритма распознавания этих языков влечет существование таких алгоритмов для каждого языка из NP. Такие языки называются NP-полными.

Точнее, язык L_0 из NP называется NP-полным, если по данному детерминированному алгоритму распознавания L_0 с временно́й сложностью $T(n) \geqslant n$ и любому языку L из NP можно эффективно найти детерминированный алгоритм, распознающий L за время $T(p_L(n))$, где p_L — полином, зависящий от L. Говорят, что L сводится к L_0. NP-полноту языка L_0 можно доказать, показав, что L_0 принадлежит NP, и каждый язык из NP можно полиномиально трансформировать в L_0.

Язык L называется полиномиально трансформируемым в L_0, если некоторая детерминированная машина Тьюринга M с полиномиально ограниченным временем работы преобразует каждую цепочку w в алфавите языка L в такую цепочку w_0 в алфавите языка L_0, что $w \in L$ тогда и только тогда, когда $w_0 \in L_0$. Если L полиномиально трансформируем в L_0 и L_0 распознается некоторым детерминированным алгоритмом a с временно́й сложностью $T(n) \geqslant n$, то можно выяснить принадлежность цепочки w к языку L, преобразовав w в w_0 с помощью машины M и затем применив a для выяснения принадлежности w_0 к языку L_0. Если время работы машины M ограничено полиномом $p(n)$, то $|w_0| \leqslant p(|w|)$. Таким образом, существует алгоритм, выясняющий принадлежность w к языку L за время $p(|w|) + T(p(|w|)) \leqslant 2T(p(|w|))$. Если бы функция T была полиномом (т. е. язык L_0 принадлежал бы P), то этот алгоритм, распознающий L, работал бы полиномиально ограниченное время, и язык L также принадлежал бы P.

Некоторые авторы действительно определяют язык L_0 как NP-полный, если он принадлежит NP и каждый язык из NP полиномиально трансформируем в L_0. Определение, основанное на сводимости, означает, что если M_0 — детерминированная машина Тьюринга, распознающая NP-полный язык L_0 за время $T(n)$, то всякий язык из NP можно распознать за время $T(p(n))$, где p — некоторый полином, с помощью ДМТ, которая обращается к M_0 как к подпрограмме нуль и более раз. Определение, основанное на трансформируемости, означает, что к M_0 можно обратиться лишь один раз и затем использовать результат ее работы заранее фиксированным образом. Какое бы из этих определений ни взять, ясно, что если некоторый детерминированный алгоритм распознает L_0 за поли-

номиальное время, то все языки из NP можно распознать за полиномиальное время. Таким образом, либо все NP-полные языки принадлежат P, либо ни один из них не принадлежит P. Первое верно тогда и только тогда, когда $P = NP$. В настоящее время мы можем выдвинуть лишь гипотезу о том, что $P \subset NP$.

Обратимся теперь к связи языков и задач. Классы P и NP определены выше как классы языков. Многие задачи, в частности задачи оптимизации, можно сформулировать как задачи распознавания языков. Например, рассмотрим задачу, которая при каждом значении входных данных требует ответа «да» или «нет». Можно закодировать все возможные значения входных данных такой задачи в виде цепочек символов и переформулировать ее как задачу о распознавании языка, состоящего из всех цепочек, представляющих входные данные нашей задачи, которые приводят к ответу «да». Чтобы связь задачи — языки стала более явной, примем некоторые соглашения о способах кодирования задач.

1. Целые числа будем представлять в десятичной системе счисления.

2. Вершины графа будем представлять целыми числами 1, 2, ..., n, закодированными как десятичные числа. Ребра (или дуги) графа будем представлять парами (i_1, i_2), где i_1, i_2 — десятичные представления вершин, составляющих ребро (дугу).

3. Булевы выражения (формулы) с n пропозициональными переменными будем записывать в виде цепочек, в которых * представляет «и», + представляет «или», $\bar{x}$ представляет «не x», а целые числа 1, 2, ..., n представляют пропозициональные переменные. Знак *, когда можно, опускается. Если нужно, будем ставить скобки. Теперь мы можем сказать, что задача принадлежит P или NP, если ее стандартный код принадлежит P или NP соответственно.

Пример. Рассмотрим задачу о внутренне устойчивом (независимом) множестве вершин графа $G = (V, E)$; V — множество вершин графа, $E \subseteq V \times V$ — множество его ребер. Подмножество вершин графа называется независимым, если для каждой пары входящих в него вершин имеет место $(v_i, v_j) \notin E$. Задача о независимом k-множестве формулируется так: содержит ли данный граф независимое множество из k вершин, где k — данное целое число? Эту задачу можно закодировать цепочкой

$$k(i_1, j_1)(i_2, j_2) \ldots (i_p, j_p) \ldots (i_r, j_r), \qquad (2.1)$$

в которой за числом k записаны все ребра графа (V, E); r — общее количество ребер, причем ребро (v_{i_p}, v_{j_p}) представлено парой (i_p, j_p) и т. д. Язык L, представляющий задачу о независимом k-множестве, образует такие цепочки вида (2.1), что граф с ребрами (i_p, j_p), $1 \leqslant p \leqslant r$, содержит независимое множество из k вершин. Задачу о независимом множестве могут представлять и другие языки. Например, можно было бы использовать двоичные числа вместо десятичных. Необходимо лишь, чтобы существовал такой полином p, что цепочку w, представляющую вход задачи о независимом множестве в одном языке, можно было бы за время $p(|w|)$ преобразовать в цепочку, представляющую тот же вход в другом языке. При таком условии, когда речь идет о принадлежности к классу P или NP, точный выбор языка для представления задачи неважен. Задача о независимом множестве принадлежит классу NP. Недетерминированная машина Тьюринга сначала может «догадаться», какие k вершин образуют независимое множество, а затем проверить, что любая пара этих вершин не соединена ребром; при этом для проверки достаточно $O(n^3)$ шагов, где n — длина кода задачи о независимом множестве. Известно, что задача о независимом множестве NP-полна, и в настоящее время неизвестны способы ее решения за детерминированное полиномиальное время.

Особый интерес представляют экстремальные задачи, например задача нахождения наибольшего независимого множества вершин графа. Многие такие задачи можно преобразовать за полиномиальное время в задачи распознавания языка. Например, наибольшее независимое множество вершин графа можно найти так. Пусть n — число вершин графа. Для каждого k от 1 до n выясняем, содержит ли граф независимое k-множество. Можно убедиться в том, что время нахождения наибольшего независимого множества таким методом полиномиально зависит от длины представления графа G и времени выяснения того, содержит ли граф независимое k-множество. В общем случае, организуя надлежащим образом двоичный поиск, можно решить задачу оптимизации вида «найти наибольшее k такое, что $P(k)$ истинно», где P — некоторое условие, а число допустимых k оценивается экспонентой от длины n стандартного описания задачи. Если из истинности $P(k)$ следует истинность $P(i)$ для $i < k$, а k изменяется от 0 до c^n, где c — некоторая постоянная, то наибольшее k, для которого истинно $P(k)$, можно найти дво-

ичным поиском, проведя $\log c^n = n \log c$ проверок условия $P(k)$. Легко убедиться в том, что оптимальное значение k можно найти за время, полиномиально зависящее от n и наибольшего времени проверки условия $P(k)$. Поэтому в дальнейшем будем заниматься только задачами, требующими ответа «да» или «нет».

2.4. *NP*-полнота задачи выполнимости булевой формулы

Рассмотрим язык L, образованный всеми цепочками, представляющими выполнимые булевы формулы (т. е. формулы, принимающие значение 1 на некотором наборе значений переменных). Покажем, что L принадлежит NP. Недетерминированный алгоритм, распознающий L, начинает с того, что «угадывает» выполняющийся набор значений пропозициональных переменных во входной цепочке, если такой набор существует. Затем угаданное значение (0 или 1) каждой переменной подставляется вместо всех вхождений этой переменной во входную цепочку. Далее вычисляется значение полученного выражения, чтобы проверить его совпадение с 1. Нетрудно вычислить значение булевой формулы за $O(n^2)$ шагов. Следовательно, некоторая недетерминированная машина Тьюринга допускает выполнимые булевы формулы с полиномиальной временно́й сложностью, потому задача распознавания выполнимости булевых формул принадлежит NP. Отметим, что «угадывание» правильного решения в действительности означает параллельную проверку всех возможных решений.

Вообще, способ доказательства NP-полноты некоторой задачи состоит обычно из двух частей. Вначале показывают, что ее представление в некотором языке, например L_0, принадлежит NP, затем показывают, что всякий язык из NP полиномиально трансформируем (или сводим) в L_0. Первая часть доказательства, как это видно из задачи выполнимости булевой формулы, обычно оказывается простой. Трудности вызывает доказательство того, что всякая задача из NP полиномиально трансформируема в данную задачу. Однако раз уж установлена NP-полнота некоторой задачи L_0, можно доказать NP-полноту любой задачи, показав, что L принадлежит NP, а L_0 полиномиально трансформируема в L.

Теорема 2.1. *Задача выполнимости булевой формулы NP-полна.*

Доказательство. Поскольку задача выполнимости принадлежит NP, остается показать, что всякий язык L из NP полиномиально трансформируем в задачу выполнимости. Пусть M — недетерминированная машина Тьюринга, распознающая L за полиномиальное время, и пусть на ее вход подана цепочка w. По M и w можно построить булеву формулу w_0, выполнимую тогда и только тогда, когда M допускает w. Основная трудность доказательства — показать, что для каждой машины M найдется алгоритм, который построит булеву формулу w_0 из w за время, ограниченное полиномом. Этот полином зависит от M. Чтобы упростить доказательство, будем полагать, что M имеет одну ленту. Предположение правомерно, так как если некоторый язык допускается k-ленточной НМТ с временно́й сложностью $T(n)$, то он допускается и одноленточной НМТ с временно́й сложностью $O(T^2(n))$ ([6], стр. 413, 414).

Пусть состояниями M будут $q_1, q_2, \ldots, q_s$, а символами на ленте — $x_1, x_2, \ldots, x_m$. Пусть $p(n)$ — временна́я сложность машины M.

Предположим, что цепочка w, поданная на вход машины M, имеет длину n. Таким образом, если M допускает w, то делает это не более чем за $p(n)$ шагов. Если M допускает w, то найдется хотя бы одна такая последовательность МО $Q_0, Q_1, \ldots, Q_q$, что Q_0 — начальное МО, $Q_{i-1} \vdash Q_i$ для $1 \leqslant i \leqslant q$, Q_q — допускающее МО, $q \leqslant p(n)$ и ни одно МО не занимает более $p(n)$ клеток на ленте.

Построим булеву формулу w_0, которая «моделирует» последовательность МО, проходимых машиной M. Любое присваивание значений «истина» (число 1) и «ложь» (число 0) переменным формулы w_0 соответствует самое большее одной последовательности МО, возможно, незаконной (т. е. такой, которая не может на самом деле реализоваться). Булева формула w_0 примет значение 1 тогда и только тогда, когда это присваивание значений переменным соответствует последовательности МО $Q_0, Q_1, \ldots$ $\ldots, Q_q$, ведущей к допусканию входа. Иными словами, булева формула w_0 будет выполнима тогда и только тогда, когда M допускает w. Перечислим пропозициональные переменные в w_0 вместе с их подразумеваемой интерпретацией.

1. $C\langle i, j, t\rangle = 1$ тогда и только тогда, когда i-я клетка на входной ленте машины M содержит символ x_j в момент времени t. Здесь $1 \leqslant i \leqslant p(n)$, $1 \leqslant j \leqslant m$ и $0 \leqslant t \leqslant p(n)$.

2. $S\langle k, t\rangle = 1$ тогда и только тогда, когда M в момент времени t находится в состоянии q_k. Здесь $1 \leqslant k \leqslant s$ и $0 \leqslant t \leqslant p(n)$.

3. $H\langle i, t\rangle = 1$ тогда и только тогда, когда головка в момент времени t обозревает i-ю клетку. Здесь $1 \leqslant i \leqslant p(n)$ и $0 \leqslant t \leqslant p(n)$.

Таким образом, пропозициональных переменных всего $O(p^2(n))$ и их можно представить двоичными числами, содержащими не более $c \log n$ разрядов, где c — постоянная, зависящая от p. В дальнейшем удобно будет представлять себе пропозициональную переменную как один символ, а не как $c \log n$ символов. Потеря множителя $c \log n$ не может отразиться на полиномиальной ограниченности времени, затрачиваемого на вычисление той или иной функции.

Из этих пропозициональных переменных построим булеву формулу w_0, следуя структуре последовательности МО $Q_0, Q_1, \ldots, Q_q$. В этом построении мы воспользуемся предикатом $U(x_1, x_2, \ldots, x_r)$, принимающим значение 1, когда в точности один из аргументов $x_1, x_2, \ldots, x_r$ принимает значение 1. Предикат U можно выразить булевой формулой вида

$$U(x_1, x_2, \ldots, x_r) = (x_1 + x_2 + \ldots + x_r) \prod_{i \neq j} (\neg x_i + \neg x_j). \tag{2.2}$$

Первый множитель в (2.2) утверждает, что по крайней мере одна из переменных x_i истинна. Остальные $r(r-1)/2$ множителей утверждают, что никакие две переменные не являются истинными одновременно. Заметим, что формальная запись формулы $U(x_1, x_2, \ldots, x_r)$ имеет длину $O(r^2)$. Если M допускает w, то найдется допускающая последовательность МО $Q_0, Q_1, \ldots, Q_q$, проходимая машиной M в процессе обработки цепочки w. Для упрощения рассмотрений мы, не умаляя общности, будем считать, что машина модифицирована так, что она делает ровно $p(n)$ шагов, а все ее МО содержат $p(n)$ клеток. Этого можно добиться, заставив M, если нужно, работать после перехода в допускающее состояние, но не изменяя его и не сдвигая головку, и дополнив все МО нужным числом пустых символов. Поэтому будем строить w_0 как произведение семи формул A, B, C, D, E, F, G, которые утверждают, что $Q_0, Q_1, \ldots, Q_q$ — допускающая последовательность МО, причем каждое МО Q_i имеет длину $p(n)$ и $q = p(n)$. Утверждение о том, что $Q_0, Q_1, \ldots, Q_{p(n)}$ — допускающая последовательность МО, равносильно совокупности утверждений:

1) в каждом МО головка обозревает ровно одну клетку;

2) в каждом МО в каждой клетке ленты стоит ровно один символ;

3) каждое МО содержит в точности одно состояние;

4) при переходе от одного МО к следующему изменяется содержимое разве что клетки, обозреваемой головкой;

5) изменение состояния, положения головки и содержимого клетки при переходе от МО к следующему происходит в соответствии с функцией переходов машины M;

6) первое МО является начальным;

7) состояние в последнем МО — заключительное.

Построим булевы формулы A, B, C, D, E, F, G, соответствующие утверждениям 1 — 7.

1. A утверждает, что в каждый момент времени машина M обозревает в точности одну клетку. Пусть A_t утверждает, что в момент t обозревается в точности одна клетка. Тогда $A = A_0 A_1 \ldots A_{p(n)}$, где

$$A_t = U(H\langle 1, t\rangle, H\langle 2, t\rangle, \ldots, H\langle p(n), t\rangle). \qquad (2.3)$$

Заметим, что если раскрыть выражение, обозначенное через U, то окажется, что формула A имеет длину $O(p^3(n))$ и ее можно записать за такое же время.

2. B утверждает, что каждая клетка ленты в каждый момент времени содержит в точности один символ. Пусть B_{it} утверждает, что i-я клетка ленты в момент времени t содержит ровно один символ.

Тогда

$$B = \prod_{i,t} B_{it},$$

где

$$B_{it} = U(C\langle i, 1, t\rangle, C\langle i, 2, t\rangle, \ldots, C\langle i, m, t\rangle). \qquad (2.4)$$

Длина каждой формулы B_{it} не зависит от n, поскольку размер m ленточного алфавита зависит только от машины Тьюринга M. Таким образом, B имеет длину $O(p^2(n))$.

3. C утверждает, что в каждый момент времени t машина M находится в одном и только одном состоянии:

$$C = \prod_{0 \leqslant t \leqslant p(n)} U(S\langle 1, t\rangle, S\langle 2, t\rangle, \ldots, S\langle s, t\rangle).$$

Так как число состояний s машины M постоянно, то C имеет длину $O(p(n))$.

4. D утверждает, что в любой момент времени t можно изменить содержимое не более чем одной клетки:

$$D = \prod_{i,j,t} [(C\langle i, j, t\rangle \equiv C\langle i, j, t+1\rangle) + H\langle i, t\rangle].$$

Формула $(C\langle i, j, t\rangle \equiv C\langle i, j, t+1\rangle) + H\langle i, t\rangle$ утверждает, что либо

а) в момент t головка обозревает клетку i, либо

б) j-й символ записан в клетке i в момент $t+1$ тогда и только тогда, когда он был записан там в момент t.

Поскольку A и B утверждают, что в момент t головка обозревает только одну клетку и клетка i содержит лишь один символ, то в момент времени t либо головка обозревает клетку i, либо содержимое клетки i не меняется. Учитывая определение знака $\equiv$, получаем, что длина D есть $O(p^2(n))$.

5. E утверждает, что очередное МО машины M получается из предыдущего одним переходом, разрешаемым функцией переходов δ машины M. Пусть E означает одно из следующих утверждений:

а) В момент t клетка i не содержит символа j.

б) В момент t головка не обозревает клетку i.

в) В момент t машина M не находится в состоянии k.

г) Очередное МО машины M получается из предыдущего переходом, разрешаемым функцией переходов машины M. Тогда

$$E = \prod_{i,j,k,t} E_{ijkt},$$

где

$$E_{ijkt} = \neg C\langle i, j, k\rangle + \neg H\langle i, t\rangle + \neg S\langle k, t\rangle + \sum_{l} C\langle i, j_l, t+1\rangle S\langle k_l, t+1\rangle H\langle i_l, t+1\rangle. \quad (2.5)$$

Здесь l пробегает по всем шагам машины M, возможным в случае, когда M обозревает символ x_j в состоянии q_k. Иными словами, для каждой тройки $\langle q, x, d\rangle$ из $\delta(q_k, x_j)$ существует одно значение l, для которого $x_{j_l} = x$, $q_{k_l} = q$ и число i_l равно $i-1$, i или $i+1$ в зависимости от d: сдвигается ли головка влево ($d = L$, $i_l = i-1$), вправо ($d = R$, $i_l = i+1$) или остается на месте ($d = S$, $i_l = i$). Здесь δ — функция переходов для M. Поскольку машина M недетерминирована, то таких троек $\langle q, x, d\rangle$ может быть несколько, но во всяком случае их число конечно. Таким образом, E_{ijkt} имеет ограниченную длину, не зависящую от n. Поэтому E имеет длину $O(p^2(n))$.

6. F утверждает, что выполнены начальные условия:

$$F = S\langle 1, 0\rangle H\langle 1, 0\rangle \prod_{1<i<n} C\langle i, j_i, 0\rangle \prod_{n<i\leqslant p(n)} C\langle i, 1, 0\rangle,$$

где $S\langle 1, 0\rangle$ утверждает, что M в момент $t=0$ находится в состоянии q_1, которое мы считаем начальным; $H\langle 1, 0\rangle$ утверждает, что M в момент $t=0$ обозревает самую левую клетку ленты; $\prod_{1\leqslant i\leqslant n} C\langle i, j_i, 0\rangle$ утверждает, что первые n клеток вначале содержат входную цепочку w; $\prod_{<i\leqslant p(n)} C\langle i, 1,0\rangle$ утверждает, что остальные клетки вначале пусты. Мы считаем, что пустым символом является x_1. Очевидно, что F имеет длину $O(p(n))$.

7. G утверждает, что M в конце концов приходит в заключительное состояние. Поскольку мы потребовали, чтобы машина M оставалась в заключительном состоянии после того, как оно достигнуто, то $G = S\langle s, p(n)\rangle$. Мы считаем, что заключительным состоянием является q_s.

Булева формула w_0 — это произведение $ABCDEFG$. Поскольку каждый из семи сомножителей состоит не более чем из $O(p^3(n))$ символов, сама формула w_0 также состоит не более чем из $O(p^3(n))$ символов. Так как мы считали каждую пропозициональную переменную за один символ, а в действительности переменные представляются цепочками длины $O(\log n)$, то на самом деле границей для $|w_0|$ будет $O(p^3(n)\log n)$, а это не превосходит $cnp^3(n)$, где c — некоторая постоянная. Таким образом, длина цепочки w_0 полиномиально зависит от длины цепочки w. Должно быть ясно, что если даны цепочка w и полином p, то формулу w_0 можно записать за время, пропорциональное ее длине.

По данной допускающей последовательности МО Q_0, Q_1, ..., Q_q можно, очевидно, найти такое присваивание значений 0 и 1 пропозициональным переменным $C\langle i, j, t\rangle$, $S\langle k, t\rangle$ и $H\langle i, t\rangle$, что w_0 примет значение «истина». Обратно, если дано присваивание значений переменным формулы w_0, при котором она становится истинной, то легко найти допускающую последовательность МО. Таким образом, формула w_0 выполнима тогда и только тогда, когда M допускает цепочку w.

Мы не наложили на язык L, допускаемый машиной M, никаких ограничений, кроме того, что он принадлежит классу NP. Поэтому мы показали, что любой язык из NP полиномиально трансформируем в задачу выполнимости

булевых формул. Отсюда заключаем, что задача выполнимости *NP*-полна.

Пусть теперь булева формула задана в конъюнктивной нормальной форме (КНФ), т. е. представляет собой произведение сумм литералов, каждый из которых — переменная или ее отрицание. Возникает задача: выполнима ли булева формула, заданная как КНФ? Аналогично, k-выполнимость определим как задачу, где речь идет о выполнимости КНФ, в которой каждый сомножитель состоит в точности из k литералов. Например, $(x+y+z)\times$ $\times(\bar{x}+w+\bar{z})$ есть 3-КНФ.

Теорема 2.2. *Задача выполнимости булевых формул, находящихся в КНФ, NP-полна.*

Доказательство. Достаточно показать, что каждая из формул A, ..., G, построенных в доказательстве теоремы 2.1, либо уже находится в КНФ, либо может быть преобразована в таковую с помощью правил булевой алгебры, причем ее длина увеличится не более чем в постоянное число раз. Формула U, определенная равенством (2.2), уже находится в КНФ. Отсюда следует, что A, B и C тоже находятся в КНФ. F и G тривиально находятся в КНФ, поскольку F и G — произведения литералов. D — формула вида $(x\equiv y)+z$. Если раскрыть знак $\equiv$, получится выражение

$$xy+\bar{x}\bar{y}+z, \tag{2.6}$$

эквивалентное

$$(x+\bar{y}+z)(\bar{x}+y+z). \tag{2.7}$$

Подставив (2.7) вместо всех вхождений (2.6) в D, получим формулу, эквивалентную исходной, но находящуюся в КНФ и превосходящую исходную формулу по длине не более чем в два раза.

Наконец, E — произведение формул E_{ijkt}. Поскольку длины всех E_{ijkt} ограничены независимо от n, каждую формулу E_{ijkt} можно преобразовать в формулу в КНФ, причем ее длина не будет зависеть от n. Поэтому преобразование формулы E в формулу в КНФ увеличивает ее длину не более чем в постоянное число раз.

Итак, формулу w_0 можно представить в КНФ, увеличив ее длину не более чем в постоянное число раз.

Теорема 2.3. *Задача 3-выполнимости NP-полна.*

Доказательство. Покажем, что выполнимость формул в КНФ полиномиально трансформируема в 3-выполнимость. В данном произведении сумм заменим

каждую сумму $(x_1 + x_2 + \ldots + x_k)$, $k \geqslant 4$, па

$$(x_1 + x_2 + y_1)(x_3 + \bar{y}_1 + y_2)(x_4 + \bar{y}_2 + y_3) \ldots$$
$$\ldots (x_{k-2} + \bar{y}_{k-4} + y_{k-3})(x_{k+1} + x_k + \bar{y}_{k-3}), \quad (2.8)$$

где $y_1, y_2, \ldots, y_{k-3}$ — новые переменные. Например, для $k = 4$ выражение (2.8) принимает вид $(x_1 + x_2 + y_1)(x_3 + x_4 + \bar{y}_1)$.

Присвоить значение новым переменным так, чтобы подставляемая формула приняла значение 1, можно тогда и только тогда, когда один из литералов $x_1, x_2, \ldots, x_k$ имеет значение 1, т. е. тогда и только тогда, когда исходная формула принимает значение 1. Допустим, что $x_i = 1$. Тогда положим $y_j = 1$ для $j \leqslant i - 2$ и $y_j = 0$ для $j > i - 2$. Подставляемая формула принимает значение 1. Обратно, допустим, что при некоторых значениях переменных y_i подставляемая формула принимает значение 1. Если $y_1 = 0$, то одна из переменных x_1 и x_2 должна иметь значение 1. Если $y_{k-3} = 1$, то одна из переменных x_{k-1} и x_k должна иметь значение 1. Если $y_1 = 1$ и $y_{k-3} = 0$, то для некоторого i, $1 \leqslant i \leqslant k - 4$, выполнены оба равенства $y_i = 1$ и $y_{i+1} = 0$, откуда следует, что переменная x_{i+2} должна иметь значение 1. В любом случае некоторая переменная x_i должна иметь значение 1.

Длина формулы, получаемой в результате описанной выше замены, превосходит длину исходной формулы не более чем в постоянное число раз. Таким образом, по данной формуле E в КНФ можно найти, применяя описанное выше преобразование к каждой сумме, формулу E' в 3-КНФ, выполнимую тогда и только тогда, когда выполнима исходная формула. Более того, легко показать, что E' можно найти за время, пропорциональное длине формулы E, выполняя преобразования очевидным образом.

2.5. *NP*-полнота простейших сетевых задач теории расписаний

2.5.1. Задачи с неравнодлительными операциями. Начнем с задачи, известной как задача составления кратчайшего мультипроцессорного расписания. Требуется построить расписание минимальной длительности для системы из n частично-упорядоченных неравнодлительных операций, выполняющихся на идентичных процессорах. Предполагается, что операции не прерываются. Решение этой

задачи в принципе можно найти, решив несколько задач, допускающих ответы «да» или «нет». Действительно, длительность кратчайшего расписания не превосходит суммы длительностей операций. В свою очередь, сумма длительностей операций ограничена экспоненциальной функцией от длины стандартного представления длительностей операций. А так как представление длительностей операций короче представления задачи в целом, то сумма длительностей операций, а следовательно и длительность кратчайшего расписания, ограничена экспоненциальной функцией от длины стандартного представления задачи. Пусть эта функция есть c^n, где c — некоторая константа. Поиск кратчайшего расписания можно осуществить по методу дихотомии. Если на некотором шаге процесса поиска найдено расписание длительностью ω_0, $0 < \omega_0 < c^n$, то на следующем шаге следует испытать значение ω_1, равное наименьшему целому числу, большему или равному $\omega_0/2$. При этом общее количество шагов не превышает $n \log c$.

Задача составления кратчайшего расписания в форме, допускающей ответы «да» или «нет», такова: «существует ли расписание без прерываний длительностью не более ω для выполнения системы из n операций $v_1, \ldots, v_i, \ldots, v_n$ с отношением непосредственного предшествования $\prec$, длительностями операций τ_i на m идентичных процессорах (единицах однородного ресурса), причем каждая из них может выполняться на любом из процессоров и каждый из процессоров может одновременно выполнять не более одной операции»? Входными данными задачи являются n, m, ω, $\prec$, $\tau_1, \ldots, \tau_n$. Каждый вход задачи можно описать с помощью пяти символов: 0, 1, левая и правая скобки, запятая. Числа n, m, ω могут быть представлены в двоичной форме. Разделяемые запятой, они начинают строку входных данных. Затем следуют пары (i, j), где i и j — числа в двоичной форме. Пара (i, j) включается в строку тогда и только тогда, когда операция v_i непосредственно предшествует v_j. Наконец, перечисляются времена τ_i, разделенные запятой. Например, пусть $n = 5$, $m = 2$, $\omega = 8$; $\prec$ задано как $1 \prec 4 \prec 5$ и $2 \prec 3$; последовательность длительностей такова: 4, 5, 4, 2, 1. Данный вход задачи описывается как последовательность

101,10,1000(1,100)(100,101)(10,11)100,101,10,1.

Ответом для этого входа является «нет», так как критический путь в данной сети операций равен 9.

Чтобы показать, что данная задача составления расписаний *NP*-полна, покажем, что к ней сводится (в нее полиномиально трансформируема) задача 3-выполнимости конъюнктивной нормальной формы.

Теорема 2.4. *Сетевая задача составления кратчайшего мультипроцессорного расписания NP-полна.*

Доказательство. Покажем, как получить за детерминированное полиномиальное время из булевой формулы E в форме 3-КНФ вход F задачи составления мультипроцессорного расписания такой, что для F существует расписание тогда и только тогда, когда E выполнима. Вход задачи F будет иметь пустое отношение предшествования $< = \varnothing$ и $m = 2$. Длительности операций в F будут довольно большими и фактически экспоненциально зависящими от длины E. Однако, поскольку длительности операций закодированы бинарно, длина представления F будет все еще полиномом от длины E.

Пусть $E = E_1 E_2 \ldots E_k$, и пусть $E_j = (x_j + y_j + z_j)$, где каждое из x_j, y_j, z_j — литерал (переменная или ее отрицание). В выражении E встречается всего p переменных, например, $e_1, \ldots, e_p$. Построим множество операций, длительность каждой из которых равна $(k+p+1)$-разрядному десятичному числу, причем разряды с меньшими номерами соответствуют суммам E_j, а разряды с большими номерами, за исключением $(k+p+1)$-го, соответствуют переменным (использование высшего разряда будет объяснено ниже).

1. Для каждой переменной e_i, $i = 1, \ldots, p$, построим операции v_i и v_i' с длительностями τ_i и τ_i'. Десятичные представления τ_i и τ_i' имеют 0 на всех местах (разрядах), за исключением следующих:

а) τ_i и τ_i' имеют 1 в $(i+1)$-м разряде слева, соответствующем переменной e_i;

б) τ_i имеет 1 на j-м месте справа, соответствующем E_j, если e_i есть x_j, либо y_j, либо z_j (т. е. если присваивание $e_i = 1$ делает E_j истинным);

в) τ_i' имеет 1 в j-м разряде справа, если $\bar{e}_i$ есть либо x_j, либо y_j, либо z_j (т. е. если присваивание $e_i = 0$ делает E_j истинным).

2. Для каждого j, $1 \leqslant j \leqslant k$, построим две операции u_j и u_j'. Пусть длительности u_j и u_j' равны $\sigma_j = \sigma_j' = 10^{j-1}$, т. е. σ_j и σ_j' имеют единственную единицу в разряде E_j и нули во всех остальных разрядах.

3. Пусть w_0 — операция с длительностью $\rho_0 = \sum_{i=k}^{p+k} 10^i$.

Таким образом, ρ_0 имеет 1 в самом левом разряде и в каждом разряде, соответствующем переменной, и 0 в разрядах, соответствующим всем E_j.

4. Пусть w_1 — операция с длительностью $\rho_1 = \sum_{i=1}^{p+k} 10^i$.

Таблица 2.2

Иллюстрация к доказательству теоремы 2.7

Операции	Длительности	Разряды								
		1	2	3	4	5	6	7	8	9
			e_1	e_2	e_3	e_4	e_5	E_1	E_2	E_3
v_1	τ_1		1							
v_2	τ_2			1				1	1	
v_3	τ_3				1			1		1
v_4	τ_4					1			1	1
v_5	τ_5						1			
v_1'	τ_1'		1					1		
v_2'	τ_2'			1						
v_3'	τ_3'				1				1	
v_4'	τ_4'					1				
v_5'	τ_5'						1			1
u_1	σ_1							1		
u_2	σ_2								1	
u_3	σ_3									1
u_1'	σ_1'							1		
u_2'	σ_2'								1	
u_3'	σ_3'									1
w_0	ρ_0	1	1	1	1	1	1			
w_1	ρ_1	1	1	1	1	1	1	1	1	1
	ω	1	2	2	2	2	2	3	3	3

В табл. 2.2 показаны 1-разряды длительностей соответствующих операций для следующей 3-КНФ:

$$E = (\bar{e}_1 + e_2 + e_3)(e_2 + \bar{e}_3 + e_4)(e_3 + e_4 + \bar{e}_5).$$

В данном случае имеем

$$p = 5,\ k = 3.$$

Наконец, положим

$$\omega = 10^{p+k} + 2\sum_{i=k}^{p+k-1} 10^i + 3\sum_{i=0}^{k-1} 10^i.$$

Заметим, что сумма длительностей всех операций равна 2ω. Действительно, в разряде 1 всего появляется 2 единицы: в строках ρ_0 и ρ_1; в разрядах со второго по $(p+1)$-й всего появляется 4 единицы: по одной в строках ρ_0, ρ_1 и по одной в строках τ_i и τ_i'; в разрядах с $(p+2)$-го по $(p+k+1)$-й всего появляется 6 единиц: по одной в строках σ_j, σ_j', ρ_1 и 3 единицы в строках τ_i и τ_i'. Последний факт следует из того, что каждая из сумм E_j выполнима, в каждой из них строго по 3 терма и каждый из термов e_i отмечается единицей либо в τ_i, либо в τ_i'. Следовательно, ответом в задаче F является «да» тогда и только тогда, когда множество длительностей операций может быть разбито на две части с равными суммами. Покажем, что ответом является «да» тогда и только тогда, когда E выполняется.

Необходимость. Предположим, что E выполняется, если переменной e_i присвоено значение a_i (0 или 1) для каждого i. Основываясь на этом присваивании, выбираем подмножество операций, сумма длительностей которых в точности равна ω. Они могут быть выполнены на одном процессоре в произвольном порядке, в то время как оставшиеся операции будут выполняться на другом процессоре также в пределах лимита времени ω. Начнем с того, что выберем v_i, если $a_i = 1$, и v_i', если $a_i = 0$. Поскольку E выполняется при данном назначении, в десятичном представлении суммы длительностей выбранных таким образом операций в каждом из k разрядов справа стоит либо 1, либо 2, либо 3. Следовательно, мы можем выбрать от 0 до 2 операций u_j и u_j' для каждого j, получив тем самым в сумме длительностей 3 в каждом из k разрядов справа. Заметим, что, поскольку мы выбрали одну из операций v_i или v_i', но не обе вместе, в суммах длительностей операций будут 0 в самом левом разряде и 1 в каждом из p следующих за ним разрядов. Следовательно, выбор w_0 даст нам подмножество операций с общей длительностью ω.

Достаточность. Заметим, что w_0 и w_1 должны выполняться на разных процессорах, поскольку сумма их длительностей превосходит ω. Рассмотрим другие операции процессора, выполняющего w_0 (назовем его P_1). Для того чтобы сумма длительностей всех операций процессора P_1 имела 2 в разрядах со второго по $(p+1)$-й, процессором P_1 должна выполняться в точности одна из операций v_i и v_i' для каждого i. Длительности выбранных v_i и v_i' должны иметь по меньшей мере одну единицу в каждом из k разрядов справа. Если бы на j-м месте справа не было единицы, длительностей операций u_j и u_j' процессора P_1 было бы недостаточно для получения значения 3 в этом разряде у общего времени выполнения всех операций процессором P_1. Теперь мы можем легко определить искомое назначение переменных выражения E. А именно, сопоставим e_i значение 1, если v_i выполняется процессором P_1, и сопоставим 0, если v_i выполняется процессором P_2. Приведенные выше соображения гарантируют, что каждое E_j равно 1 при данном назначении. Фактически мы доказали более сильное утверждение, чем теорема, поскольку в доказательстве речь шла о задаче с $\prec = \varnothing$ и $m = 2$.

Следствие 2.1. *Задача составления мультипроцессорного расписания с двумя процессорами и пустым отношением предшествования NP-полна.*

В более общих терминах следствие 2.1 эквивалентно следующему утверждению, в котором идет речь об известной задаче о камнях.

Следствие 2.2. *Задача определения того, можно ли разбить заданный набор целых чисел на две части таким образом, чтобы суммы чисел, попавших в каждую часть, были равны, является NP-полной.*

Заметим, что в приведенном выше доказательстве существенно используются большие значения длительностей операций. Можно возразить, что реальная «размерность» задач составления расписаний зависит не только от длины записи информации о длительностях операций, но и от суммы длительностей. Способность различных переменных, от которых зависит задача, делать ее трудной не всегда пропорциональна длине кода этих переменных. Кажется правдоподобным, что в задаче составления расписания именно длины чисел в десятичном представлении в большей мере усложняют задачу, чем другие ее параметры. Однако можно показать, что даже специальные

случаи задач, в которых длительности операций принимают значения 1 или 2, но с непустым отношением предшествования, также являются NP-полными. Таким образом, структура отношения предшествования также порождает комбинаторные трудности, присущие случаю с большими длительностями операций.

2.5.2. Задачи с m процессорами и единичными длительностями операций. Задача с m идентичными процессорами и n операциями единичной длительности (задача Ху), заданная в форме «да — нет», представляет собой вопрос: «существует ли расписание длины ω для выполнения на m процессорах n операций с длительностями $\tau_i = 1$, $i = 1, \ldots, n$, и отношением предшествования операций $\prec$»? Модифицированная задача составления расписания с единичными длительностями представляет собой вопрос: «существует ли расписание длины ω, если заданы длительности $\tau_i = 1$, $i = 1, \ldots, n$, отношение предшествования $\prec$ и $m(t)$ — количество процессоров, имеющихся в момент времени t, $0 \leqslant t \leqslant \omega$»?

Покажем, что ко второй из этих задач сводится задача выполнимости 3-КНФ, а вторая задача сводится к первой.

Лемма 2.1. *Модифицированная задача составления расписания с единичными длительностями операций NP-полна.*

Доказательство. Сведем задачу 3-выполнимости к модифицированной задаче составления расписания. Пусть $E = E_1 E_2 \ldots E_k$ — булева формула в 3-КНФ, и пусть $E_j = x_j + y_j + z_j$, $1 \leqslant j \leqslant k$, где x_j, y_j, z_j — литералы. Пусть $e_1, \ldots, e_p$ — переменные, встречающиеся в E. Построим вход модифицированной задачи составления расписания.

Операциями являются:

1) v_{ij} и $\bar{v}_{ij}$, $1 \leqslant i \leqslant p$, $0 \leqslant j \leqslant p$;
2) w_i и $\bar{w}_i$, $1 \leqslant i \leqslant p$;
3) D_{ij}, $1 \leqslant i \leqslant k$, $1 \leqslant j \leqslant 7$.

Операции v и w представляют переменные; операции D — сомножители выражения E.

Отношение предшествования $\prec$ задается следующим образом:

1) $v_{ij} \prec v_{i,j+1}$, $\bar{v}_{ij} \prec \bar{v}_{i,j+1}$, $1 \leqslant i \leqslant p$, $0 \leqslant j \leqslant p$.
2) $v_{i,i-1} \prec w_i$, $v_{i,i-1} \prec \bar{w}_i$, $1 \leqslant i \leqslant p$.
3) Рассмотрим D_{ij}, и пусть $a_1 a_2 a_3$ — двоичное представление j (заметим, что случай $a_1 = a_2 = a_3 = 0$ не может иметь места). Пусть литералы x_i, y_i и z_i, входящие в E_i, являются соответственно переменными e_r, e_s и e_t

или их отрицаниями. Пусть $v_{rp} \prec D_{ij}$, если $x_i = e_r$ и $a_1 = 1$, или $x_i = \bar{e}_r$ и $a_1 = 0$. В противном случае пусть $\bar{v}_{rp} \prec D_{ij}$. Аналогично, $v_{sp} \prec D_{ij}$, если $y_i = e_s$ и $a_2 = 1$, или $y_i = \bar{e}_s$ и $a_2 = 0$, иначе $\bar{v}_{sp} \prec D_{ij}$. Наконец, $v_{tp} \prec D_{ij}$, если $z_i = e_t$ и $a_3 = 1$, или $z_i = \bar{e}_t$ и $a_3 = 0$, иначе $\bar{v}_{tp} \prec D_{ij}$.

Время завершения $\omega = p + 3$.

Число имеющихся процессоров $m(t)$ в момент времени t таково:

$$m(0) = p,$$
$$m(1) = 2p + 1,$$
$$m(t) = 2p + 2, \quad 2 \leqslant t \leqslant p,$$
$$m(p+1) = k + p + 1,$$
$$m(p+2) = 6k.$$

Заметим, что для выполнения всех операций за $(p + 3)$ единицы времени необходимо, чтобы все процессоры были полностью загружены при любом t, $0 \leqslant t \leqslant p + 2$. Действительно, общее количество операций, которые могут выполнить все процессоры на этом интервале времени, равно $\sum_{t=0}^{p+2} m(t) = 2p(p+2) + 7k$, что в точности равно общему числу операций сети. В частности, для любого j, $1 \leqslant j \leqslant p$, общее количество операций, которые могут быть выполнены к моменту времени j, не превосходит

$$\sum_{t=0}^{j} m(t) = 2j(p+1) + p - 1.$$

Можно показать, что расписание существует тогда и только тогда, когда задача выполнимости разрешима. Идея доказательства следующая. Представим, что e_i (или $\bar{e}_i$) истинно тогда и только тогда, когда v_{i0} (или $\bar{v}_{i0}$ соответственно) начинает выполняться в момент времени 0. Мы увидим, что наличие w и $\bar{w}$ обуславливает то, что точно одна из операций v_{i0} и $\bar{v}_{i0}$ должна начинаться в момент времени 0, а остальные должны начинаться в момент времени 1. Тогда требование, что $(k + m + 1)$ операций должны начинаться в момент времени $(p + 1)$, равносильно тому, чтобы для каждого i было одно j такое, что D_{ij} может начинаться в тот же момент времени. Но это условие эквивалентно тому, что сумма термов E_i имеет значение «истина», когда e_i и $\bar{e}_i$ такие, что соответствующие v_{i0} или $\bar{v}_{i0}$ выполняются в момент времени 0, принимают значение «истина».

Покажем вначале, что в случае модифицированной задачи мы не можем начинать обе операции v_{i0} и $\bar{v}_{i0}$ в момент времени 0 ни для какого i. Предположим обратное. Тогда, поскольку $m(0) = p$, найдется некоторое j такое, что ни v_{j0} ни $\bar{v}_{j0}$ не могут начинаться в момент времени 0. Тогда ни w_j, ни $\bar{w}_j$ не начинаются в момент времени j или ранее, поскольку, например, w_j предшествуют операции $v_{j0}, v_{j1}, \ldots, v_{j,\,j-1}$, причем каждая выполняется строго одна за другой. Операции, которые могут быть начаты не позднее j, таковы:

1) не более $p(2j+1)$ операций v и $\bar{v}$, т. е. $v_{i0}, v_{i1}, \ldots$ $\ldots, v_{ij}$, если v_{i0} была выполнена в момент времени 0, и $v_{i0}, v_{i1}, \ldots, v_{i,\,j-1}$ в противном случае;

2) не более $2(j-1)$ операций w, а именно $w_1, \bar{w}_1, w_2, \bar{w}_2, \ldots, w_{j-1}, \bar{w}_{j-1}$.

Следовательно, общее число операций, которые могли быть начаты к моменту времени j, не превосходит $(2pj + 2j + p - 2)$. Однако при $1 \leqslant j \leqslant p$ общее число загруженных процессоров

$$\sum_{i=0}^{j} m(i) = 3p + 1 + (j-1)(2p+2) = 2pj + 2j + p - 1.$$

Отсюда мы можем заключить, что в каждом решении модифицированной задачи строго одна из операций v_{i0} и $\bar{v}_{i0}$ начинается в момент времени 0. Более того, мы можем определить точно операции, которые начаты в каждый момент времени между 1 и p, если указано, какая из операций v_{i0} и $\bar{v}_{i0}$ выполняется в момент времени 0. А именно, в момент j мы должны начать v_{ij}, если $\bar{v}_{i0}$ была начата в момент времени 0, и начать $\bar{v}_{i,\,j-1}$ в противном случае. Более того, мы должны начать w_j (соответственно $\bar{w}_j$) в момент времени j, если v_{j0} (соответственно $\bar{v}_{j0}$) была начата в момент времени 0, и начать w_{j-1} (соответственно $\bar{w}_{j-1}$) в момент времени j, если v_{j0} (соответственно $\bar{v}_{j0}$) была начата в момент времени 1.

Начиная с момента времени $(p+1)$, мы можем выполнить p оставшихся операций v и $\bar{v}$ и одну оставшуюся операцию w или $\bar{w}$. Поскольку $m(p+1) = p + k + 1$, нам следует выполнить также k из операций D. Заметим, что для каждой пары D_{ij} и D_{ih}, $j \neq h$, найдется по меньшей мере одно l такое, что выполнено $v_{lp} \prec D_{ij}$ и $\bar{v}_{lp} \prec D_{ih}$ или обратное условие, т. е. $v_{lp} \prec D_{ih}$ и $\bar{v}_{lp} \prec D_{ij}$. Поскольку мы уже показали, что в точности одна из операций v_{lp} и $\bar{v}_{lp}$ может быть начата в момент времени p, для каждого i

самое большее одна из операций $D_{i1}, \ldots, D_{i7}$ может быть начата в момент времени $(p+1)$.

Более того, если мы назначим значение «истина» переменной e_j (соответственно $\bar{e}_j$) тогда и только тогда, когда операция v_{j0} (соответственно $\bar{v}_{j0}$) была начата в момент времени 0, мы можем начать одну из операций $D_{i1}, \ldots, D_{i7}$ в момент $(p+1)$ тогда и только тогда, когда E_j истинно при этом назначении переменных. Таким образом, решение модифицированной задачи существует тогда и только тогда, когда исходное произведение сумм (КНФ) выполняется.

Теорема 2.5. *Задача составления расписания с единичными операциями NP-полна.*

Доказательство. Сведем модифицированную задачу к немодифицированной. Если модифицированная задача имеет временной интервал ω и $m(i)$ процессоров в момент времени i, $0 \leqslant i \leqslant \omega$, пусть $m = 1 + \max\limits_i m(i)$. Тогда для $0 \leqslant i \leqslant \omega$ создадим $(m - m(i))$ новых операций $v_{i1}, \ldots, v_{i, m-m(i)}$. Пусть $v_{i-1, j} < v_{ik}$ для всех $0 < i < \omega$ и каждых j и k. Тогда, если исходная модифицированная задача имела допустимое расписание, расширенная задача также будет иметь его. Сведение в обратную сторону очевидно.

2.5.3. Задачи с двумя идентичными процессорами и длительностями операций, равными 1 или 2. Рассмотрим задачу, в которой число процессоров и период планирования не являются параметрами, а длительности операций хотя и являются параметрами, могут принимать значения только 1 или 2. Она также оказывается *NP*-полной. Этот результат ясно показывает, что комбинаторная структура отношения предшествования операций сама по себе способна делать задачу трудной (*NP*-полной).

Задача z представляет собой вопрос: «существует ли расписание длины $\omega = \frac{1}{2}\sum\limits_{i=1}^{n} \tau_i$ выполнения на двух процессорах множества из n операций с длительностями $\tau_i = 1$ или 2»? Заметим, что, если такое расписание существует, оно не имеет простоев. Для доказательства того, что эта задача *NP*-полна, потребуется следующая

Лемма 2.2. *Задача с единичными длительностями и $n = \omega m$ NP-полна.*

Доказательство такое же, как и в лемме 2.1 и теореме 2.5.

Теорема 2.6. *Задача z NP-полна.*

Доказательство. Сведем задачу с единичными длительностями и условием $n = \omega m$ к задаче z. Пусть задана задача с единичными длительностями с временным интервалом ω, числом процессоров m и отношением предшествования $\prec$ на множестве V, состоящем из ωm операций. Построим задачу z; операциями являются:

1) X_i для $0 \leqslant i < \omega'$, где $\omega' = (4m + 1)\omega$;

2) Y_{ij} для $0 \leqslant i \leqslant \omega$ и $0 \leqslant j \leqslant m$;

3) v и v' для каждой операции v из V.

Длительности операций v равны 2 для всех v из V и равны 1 для всех других операций.

Отношение $\prec'$ определяется следующими условиями:

а) $X_i \prec' X_{i+1}$ для $0 \leqslant i \leqslant \omega' - 1$;

б) $X_u \prec' Y_{ij} \prec' X_{u+2}$, где $u = (4m + 1)i + 2j - 1$ (в случае $i = j = 0$ $u = -1$ и мы имеем только отношение $Y_{00} \prec' X_1$);

в) $v' \prec' v$ для всех v из V;

г) $w \prec' v'$ для w и v из V, если и только если $w \prec v$.

Срок завершения равен ω'. Заметим вначале, что если необходимо выдержать срок, то по условию а) в каждый момент времени один процессор должен быть выделен для обработки одной из операций X. Пусть этим процессором является P_1. По условию б) операции Y должны выполняться на втором процессоре в определенные моменты времени, как это показано на рис. 2.1.

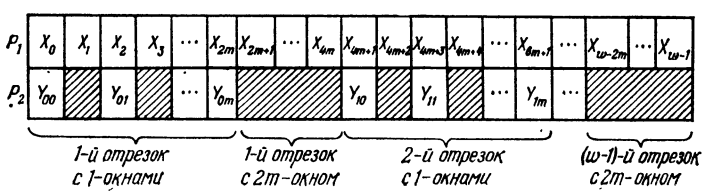

Рис. 2.1.

Картина во времени такова: мы имеем чередование отрезков, в которых имеется одна единица времени на процессоре P_2 (1-окна), и полос, в которых имеется $2m$ последовательных единиц времени на P_2 ($2m$-окна). Поскольку операции v в V требуют двух единиц времени, ясно, что они должны выполняться только в отрезках с $2m$-окном. Поскольку имеется ωm операций v, они должны полностью заполнять эти отрезки, что означает, что операции v' для v из V должны выполняться последовательно в течение отрезков с 1-окнами.

Как следствие, если операции w и v выполняются в одном и том же отрезке с $2m$-окном, случай $w \prec v$ невозможен. Из этого следовало бы, что v' также должна выполняться в этом отрезке (поскольку $w \prec' v' \prec v$), что противоречит нашему предыдущему заключению. Следовательно, если задача z имеет решение, мы можем найти решение исходной задачи с единичными длительностями, начиная в момент времени i точно те операции, которые выполняются в i-м отрезке с $2m$-окном.

Обратно, если имеется некоторое решение рассматриваемой задачи с единичными длительностями, можно найти решение построенной задачи z, выполняя v' в отрезке i с 1-окнами и v в отрезке i с $2m$-окном исходной задачи.

2.6. *NP*-полнота задач с несколькими разнородными процессорами и независимыми цепями операций

2.6.1. Одномаршрутная задача. Формально задача «m машин — n деталей» с одинаковыми технологическими маршрутами обработки деталей (задача Джонсона) задается множеством цепей (или работ) $\bar{c}$, где каждая работа $c \in \bar{c}$ есть последовательность операций $c[1], c[2], \ldots, c[m]$, а каждая операция $c[i]$ имеет длительность $\tau(c[i])$, которая предполагается неотрицательным целым числом. Расписание для совокупности цепей $\bar{c}$ определяется как функция S: $\{c[i] \mid c \in \bar{c},\ 1 \leqslant i \leqslant m\} \to \mathbf{R}^+$ ($\mathbf{R}^+$ — множество неотрицательных целых чисел), определяющая времена начала каждой операции на соответствующей машине при следующих двух ограничениях:

1) если операция $c[i]$ начата в момент времени $S(c[i])$, то она является единственной операцией, выполняемой по машине P_i в течение следующих $\tau(c[i])$ единиц времени, т. е. для всех $t > 0$ и всех i, $1 \leqslant i \leqslant m$,

$$|\{c \mid S(c[i]) < t < S(c[i]) + \tau(c[i])\}| \leqslant 1;$$

2) ограничение предшествования: для всех $c \in \bar{c}$ и всех i, $1 \leqslant i \leqslant m$, операция $c[i+1]$ не может быть начата ранее чем через $\tau(c[i])$ единиц времени после начала операции $c[i]$, т. е.

$$S(c[i]) + \tau(c[i]) \leqslant S(c[i+1]).$$

Время завершения операции $c[i]$ в расписании S, обозначаемое $f_S(c[i])$, определяется как $f_S(c[i]) = S(c[i]) + \tau(c[i])$. Длительность расписания S для $\bar{c}$ равна макси-

мальному времени завершения операций. Экстремальная задача состоит в определении кратчайшего расписания.

Для анализа вопросов эффективности алгоритмов необходимо точно определить термин «длина входа задачи». Одно из определений, может означать «количество битов, необходимых для полного описания совокупности $\bar{c}$». Обозначим эту меру $m_1(\bar{c})$. Вторая мера, в данном случае более важная,— это сумма длительностей всех операций:

$$m_2(\bar{c}) = \sum_{c \in \bar{c}} \sum_{i=1}^{m} \tau(c[i]).$$

Сложность задачи часто выражают в терминах третьего параметра, зависящего от $\bar{c}$,— количества n цепей c в $\bar{c}$. Утверждение, что алгоритм имеет трудоемкость $T(n)$, содержит неявное предположение о том, что такими числами, как длительности операций, можно оперировать (считывать, записывать) за время, не зависящее от их величины. Это предположение реалистично только для сравнительно малых чисел, и в этих случаях n будет пропорциональным $m_1(c)$. Обычно в доказательствах NP-полноты некоторой задачи используется мера m_1. Однако в задачах составления расписаний этот подход может быть использован с оговорками. Возможны такие случаи, что задача является NP-полной, когда ее вход измеряется величиной m_1, и имеет полиномиальный алгоритм, когда ее вход измеряется величиной m_2. Например, задача определения того, может ли совокупность независимых операций быть выполнена в заданное время (директивный срок) на двух идентичных процессорах (машинах), является NP-полной, когда ее вход измеряется величиной m_1. С другой стороны, та же задача может быть решена методом динамического программирования за время, пропорциональное произведению суммы длительностей операций на общее их число.

Эти замечания не противоречат утверждению о том, что эффективные алгоритмы решения NP-полных задач неизвестны. Поскольку целое n можно представить, используя не более $\log n$ битов, сумма длительностей операций сама по себе может быть экспоненциальной функцией количества битов, необходимого для представления длительностей операций. Следовательно, упомянутый выше алгоритм динамического программирования, хотя и оказывается полиномиальным от m_2, может быть экспоненциальным, если вход задачи измеряется величиной m_1. С дру-

гой стороны, указанный алгоритм все же можно успешно применять. Для нахождения кратчайшего расписания мы могли бы пожелать потратить время, по меньшей мере полиномиальное от длительности искомого расписания. Следовательно, результаты о NP-полноте с использованием меры m_1 все еще оставляют место для полезных алгоритмов, если m_2 является подходящей мерой измерения входа задачи. Чтобы исключить и эту возможность, покажем далее, что задача «машины — детали» NP-полна, даже если вход ее измеряется с помощью меры m_2. Тогда NP-полнота по мере m_1 также будет иметь место.

Как обычно, задача минимизации длительности расписания представляется в виде задачи, допускающей ответ «да» или «нет». Например, для задачи трех машин получаем: заданы совокупность $\bar{c}$ с длительностями операций τ: $\{c[i] \mid c \in \bar{c},\ 1 \leqslant i \leqslant 3\} \to \mathbf{R}^+$, а также директивный срок D; существует ли расписание S для $\bar{c}$, длительность которого не превосходит D? Доказательство NP-полноты этой задачи состоит из двух частей. В первой части показывается, что задача может быть решена за полиномиальное время недетерминированной машиной Тьюринга. В большинстве случаев, как и в данном, эта часть доказательства тривиальна. Далее она не будет приводиться. Основной частью доказательства является «редукция», т. е. сведение некоторой NP-полной задачи к данной задаче Y в следующем смысле: для данного входа x известной NP-полной задачи X показывается, как можно построить вход y задачи Y такой, что ответом на y является «да» тогда и только тогда, когда ответом на x является «да». Более того, длина входа y, а также время построения y должны быть полиномиальной функцией от $m(x)$, где m — мера для измерения длины входа в задаче X.

В данном случае в качестве «известной NP-полной задачи» используется следующая теоретико-числовая задача.

Разбиение на тройки: «даны положительные целые числа n, B и множество целых чисел $A = \{a_1, a_2, \ldots, a_{3n}\}$, где $0 < a_i < B$ для $1 \leqslant i \leqslant 3n$; существует ли разбиение $\langle A_1, A_2, \ldots, A_n \rangle$ множества A на тройки так, что для каждого i, $1 \leqslant i \leqslant n$, $\sum_{a \in A_i} a = B$»?

Внесем некоторые ограничения в задачу, сохраняющие ее NP-полноту. Во-первых, положим, что $\sum_{i=1}^{n} a_i = nB$, иначе не существует ни одного такого разбиения. Во-вторых,

мы можем положить, что каждое a_i удовлетворяет условию $B/4 < a_i < B/2$ и, следовательно, каждое подмножество чисел a_i, сумма которых равна B, должно содержать ровно три элемента (задачу в исходной форме можно преобразовать в редуцированную, положив $a_i' = a_i + B$, $1 \leqslant i \leqslant 3n$, и $B' = 4B$). Итак, будем использовать следующую *NP*-полную задачу [10] о 3-разбиении. Даны положительные целые числа n, B и множество целых чисел $A = \{a_1, \ldots, a_{3n}\}$, причем $\sum_{i=1}^{n} a_i = nB$ и $B/4 < a_i < B/2$ для всех i, $1 \leqslant i \leqslant 3n$. Существует ли разбиение $\langle A_1, \ldots, A_n\rangle$ множества A на 3-элементные подмножества такие, что для каждого i $\sum_{a \in A_i} a = B$?

Теорема 2.7. *Одномаршрутная задача составления расписания для трех машин NP-полна, даже если длина входа измеряется как сумма длительностей операций.*

Доказательство. Покажем, что задача 3-разбиения сводится к данной задаче. Положим, что даны n, B и $A = \{a_1, \ldots, a_{3n}\}$, как требуется в задаче 3-разбиения. Соответствующий вход в задаче составления расписания с $m = 3$ является совокупностью $\bar{c} = \{c_i | 0 \leqslant i \leqslant n\} \cup \cup \{E_j | 1 \leqslant j \leqslant 3n\}$ цепей с длительностями операций, определенными как

$$\tau(c_0) = \langle 0, B, 2B\rangle,$$
$$\tau(c_i) = \langle 2B, B, 2B\rangle, \quad 1 \leqslant i \leqslant n-1,$$
$$\tau(c_n) = \langle 2B, B, 0\rangle,$$
$$\tau(E_j) = \langle 0, a_j, 0\rangle, \quad 1 \leqslant j \leqslant 3n.$$

Директивный срок D равен $(2n+1)B$. Заметим, что $m_2(\bar{c})$, сумма длительностей операций c_i, $0 \leqslant i \leqslant n$, равна $(5n+1)B$, что, очевидно, ограничено полиномом от $\sum_{i=1}^{n} a_i = nB$ — времени, необходимого для конструирования этого входа.

Остается еще показать, что требуемое разбиение множества A существует тогда и только тогда, когда существует расписание для $\bar{c}$ с длительностью, меньшей или равной D.

Предположим, что существует разбиение $\langle A_1, \ldots, A_n\rangle$ требуемой формы, т. е. каждое подмножество A_i состоит из трех элементов $a_{g(i,1)}$, $a_{g(i,2)}$ и $a_{g(i,3)}$, так что для всех i,

$1 \leqslant i \leqslant n$, $\sum_{j=1}^{3} a_{g(i,j)} = B$. Построим расписание S, имеющее длину $D = (2n+1)B$. Сначала составим расписание для цепей c_i:

$$S(c_0) = \langle 0,\ 0,\ B\rangle,$$

$$S(c_i) = \langle 2B(i-1),\ 2Bi,\ 2Bi+B\rangle, \quad 1 \leqslant i \leqslant n.$$

Обратимся теперь к рис. 2.2. Заметим, что построенное

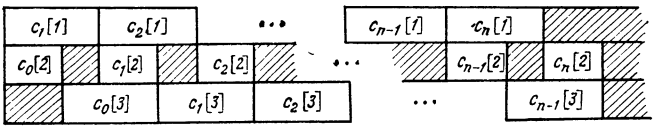

Рис. 2.2.

расписание для цепей c_i оставляет открытыми n «временны́х щелей» на машине P_2, каждая длины B. Они строго следуют одна за другой, так что мы можем положить

$$S(E_{g(i,1)}) = \langle 0,\ 2Bi-B,\ 2Bn+B\rangle,$$

$$S(E_{g(i,2)}) = \langle 0,\ 2Bi-B+a_{g(i,1)},\ 2Bn+B\rangle,$$

$$S(E_{g(i,3)}) = \langle 0,\ 2Bi-B+a_{g(i,1)}+a_{g(i,2)},\ 2Bn+B\rangle$$

для каждого i, $1 \leqslant i \leqslant n$. Поскольку $\sum_{j=1}^{3} a_{g(i,j)} = B$, $1 \leqslant i \leqslant n$ это приводит к допустимому расписанию, длительность которого $l(S) = D$.

Обратное, предположим, что существует расписание длительностью $l(S) \leqslant D$. Заметим, что мы должны иметь $l(S) = D = (2n+1)B$ и что цепи c_i должны быть расписаны по машинам таким же образом, как и на рис. 2.2, за исключением того, что цепи $c_1, \ldots, c_{n-1}$, являясь идентичными, могут быть переставлены. Следовательно, опять имеется n щелей длины B, в которые должны быть помещены операции цепей E_j машины P_2. Поскольку общая длительность операций $E_j[2]$ равна $\sum_{i=1}^{n} a_i = nB$, каждая из n щелей должна быть заполнена полностью и, следовательно, должна содержать множество операций, общая длительность которых равна B. Далее, поскольку $a_i > B/4$, ни одно из таких подмножеств не может содержать более трех операций. Аналогично, поскольку $a_i < B/2$, ни одно подмножество не может содержать менее

трех операций. Таким образом, каждое подмножество содержит ровно три операции $E_j[2]$. Следовательно, положив

$$A_i = \{a_j \mid 2B(i-1) < S(E_j[2]) < 2Bi\}, \quad 1 \leqslant i \leqslant n,$$

получим требуемое расписание.

2.6.2. Разномаршрутная задача. Широко известным обобщением задачи с одинаковыми маршрутами является задача с различными маршрутами. В этой модели каждая цепь $c = (c[1], \ldots, c[k])$ состоит из операций $c[i]$, имеющих длительности $\tau(c[i])$, и назначения машин $m(c[i])$, которое определяет имя (номер) машины, обрабатывающей операцию $c[i]$. Цепи не обязаны содержать равное число операций, однако ограничение «операция $c[i]$ должна быть завершена до начала операции $c[i+1]$» остается.

Теорема 2.8. *Задача составления расписания для двух машин и неодинаковых технологических маршрутов NP-полна, даже если длина входа задачи измеряется как сумма длительностей операций.*

Доказательство. Снова сведем задачу о 3-разбиении к данной задаче. Положим, что даны n, B и $A = \{a_1, \ldots, a_{3n}\}$. Вход в задаче двух машин с неодинаковыми цепями операций получается следующим образом. Совокупность цепей $\bar{c}$ состоит из цепи c_0, которая имеет $2n$ операций, и цепей c_i, $1 \leqslant i \leqslant 3n$, каждая из двух операций. Назначение машин и длительности операций таковы:

$$m(c_0[i]) = \begin{cases} P_1\colon\ 2 \leqslant i \leqslant 2n, & i \text{ четно}, \\ P_2\colon\ 1 \leqslant i \leqslant 2n-1, & i \text{ нечетно}; \end{cases}$$

$$m(c_j[1]) = P_1, \quad 1 \leqslant j \leqslant 3n,$$

$$m(c_j[2]) = P_2, \quad 1 \leqslant j \leqslant 3n,$$

$$\tau(c_0[i]) = B, \quad 1 \leqslant i \leqslant 2n,$$

$$\tau(c_j[1]) = 0, \quad 1 \leqslant j \leqslant 3n,$$

$$\tau(c_j[2]) = a_j, \quad 1 \leqslant j \leqslant 3n.$$

Директивный срок $D = 2nB$. Легко убедиться в том, что для $\bar{c}$ существует расписание с длительностью, меньшей или равной D, тогда и только тогда, когда множество A имеет требуемое разбиение.

2.7. Сложность обобщенной задачи составления расписания с векторными потребностями в ресурсах

Рассмотрим теперь задачи, являющиеся естественным обобщением задач из разделов 2.5 и 2.6. Обобщение заключается в том, что вместо указания имени машины (процессора) описание операции включает в себя вектор потребностей операции в ресурсах. Более точно вход обобщенной задачи составления расписаний определяется следующей информацией: множеством $R = \{R_1, \ldots, R_K\}$ видов ресурсов вместе с верхней границей B_k для каждого вида; множеством $V = \{v_1, \ldots, v_n\}$ операций вместе с частичным порядком $\prec$ на V; длительностью τ_i и потребностями в ресурсах $r_i^1, \ldots, r_i^K$ для каждой операции v_i; директивным сроком D выполнения всех операций.

Функция S: $V \to \{0, 1, \ldots, (D-1)\}$ называется правильным расписанием для заданного входа, если она удовлетворяет следующим условиям:

1) для каждого $v_i \in V$ $S(v_i) + \tau_i \leqslant D$;

2) для всех v_i, $v_j \in V$, если $v_i \prec v_j$, то $S(v_i) + \tau_i \leqslant S(v_j)$;

3) для каждого целого числа t, $0 \leqslant t < D$, и каждого k, $1 \leqslant k \leqslant K$,

$$\sum_{v_i \in E_S(t)} r_i^k \leqslant B_k,$$

где $E_S(t) = \{v_i \in V \mid S(v_i) \leqslant t \leqslant S(v_i) + \tau_i\}$ — множество операций, выполняемых в момент времени t в соответствии с расписанием S.

Как и ранее, будем рассматривать алгоритмы, которые при заданном входе I отвечают на вопрос: «существует ли правильное расписание для I»? Покажем, что обобщенная задача составления расписания является *NP*-полной, сведя к ней известную *NP*-полную задачу о покрытии.

Входом задачи о покрытии является конечное множество $A = \{a_1, \ldots, a_n\}$, некоторое семейство его подмножеств B_i, $i = 1, \ldots, m$, и положительное целое число q. Вопрос: «существует ли подсемейство из q или менее подмножеств B_i такое, что $\cup B_i = A$»?

Теорема 2.9. *Обобщенная задача составления расписания NP-полна.*

Доказательство. Свяжем с данной задачей о покрытии некоторую задачу составления расписания, решение которой существует тогда и только тогда, когда

задача о покрытии имеет решение. А именно, каждому подмножеству B_i поставим в соответствие операцию v_i, а каждому элементу a_i — вид нескладируемого ресурса. Кроме того, введем две новые операции v_{m+1} и v_{m+2} и один новый вид ресурса с номером $(K+1)$. Длительности операций, отношение предшествования, потребности операций в ресурсах, общие уровни наличия ресурсов и директивный срок определим следующим образом:

$$\tau_i = 1, \quad i = 1, \ldots, (m+1); \quad \tau_{m+2} = m - q;$$

$$v_{m+1} \prec v_{m+2};$$

$$r_i^k = \begin{cases} 0, & \text{если } a_k \in B_i, \\ 1 & \text{в противном случае,} \end{cases} \quad i = 1, \ldots, m;$$

$$r_i^{k+1} = 0, \quad i = 1, \ldots, m; \quad r_i^k = 0, \quad k = 1, \ldots, (K+1);$$

$$r_{m+2}^k = 0, \quad k = 1, \ldots, K; \quad r_{m+2}^{K+1} = q - 1;$$

$$B_k = (q-1), \quad k = 1, \ldots, K; \quad B_{K+1} = q;$$

$$D = m - q + 1.$$

Пусть существует покрытие множества A q его подмножествами B_i. Остальные подмножества семейства $\bar{A}$ обозначим $A_{i(q+1)}, \ldots, A_{im}$. Построим расписание вида

$$t_i = 0, \qquad i = 1, \ldots, q, m+1,$$

$$t_i = i - q, \quad i = q+1, \ldots, m,$$

$$t_{m+2} = 1,$$

являющееся допустимым по ресурсным и сетевым ограничениям и имеющее длительность $(m - q + 1)$. Остановимся на ресурсных ограничениях. При $t > 1$ кроме операции v_{m+2} в каждый момент времени выполняется только одна из операций v_i, $i > r$. Следовательно, потребляется не более единицы ресурсов вида с 1-го по K-й и q единиц $(K+1)$-го вида. При $t \leqslant 1$ выполняются операции v_i, $i = 1, \ldots, (n+1)$. Последняя не потребляет ресурсов, а остальные потребляют не более чем по единице ресурсов вида с 1-го по K-й и по единице ресурса $(K+1)$-го вида. Таким образом, потребляется не более $(q-1)$ единиц ресурсов $k = 1, \ldots, K$ и q единиц $(K+1)$-го вида. Действительно, для любого k от 1 до K найдется множество A_i, покрывающее элемент $a_k \in A$, а соответствующая операция v_i, выполняемая в рассматриваемый момент времени t, не потребляет k-го ресурса.

Пусть теперь существует допустимое расписание указанного вида, имеющее длительность $(m-q+1)$. Покажем, что множество A можно покрыть q подмножествами из $\bar{A}$. Поскольку операция v_{m+2} выполняется с $t=1$ по $t=m-q+1$, в каждый момент времени из этого интервала тратится $(q-1)$ единиц $(K+1)$-го вида ресурсов. Одновременно с операцией v_{m+2} может выполняться по одной операции в каждую единицу времени, всего $(m-q)$ операций. Таким образом, не менее q операций из $\{v_1, \ldots, v_m\}$ выполняется с $t=0$ по $t=1$. Пусть это $v_{i1}, \ldots, v_{i\lambda}, \ldots, v_{iD}$. Тогда

$$\sum_{\lambda=1}^{D} r_{i\lambda}^{k} \leqslant q-1, \qquad k=1, \ldots, K.$$

Следовательно, для любого $k=1, \ldots, K$ существует такая операция v_i, что $r_i^k = 0$. Но тогда $a_k \in A_i$ и каждый элемент из A покрыт хотя бы одним множеством A_i, $i = 1, \ldots, q$.

Библиографический комментарий

Публикации по теории сложности сетевых задач оптимального распределения ресурсов содержат в основном результаты, связанные с их *NP*-полнотой в смысле Кука — Карпа [1, 2], т. е. утверждения об относительной сложности тех или иных частных задач. В то же время неизвестны нетривиальные оценки абсолютной сложности этих же задач. Соответственно, в данной главе приведены главным образом результаты о *NP*-полноте ряда простейших сетевых задач составления расписаний. Охарактеризуем вкратце публикации, связанные с содержанием главы.

Во вводном разделе 2.1 определяются основные характеристики, в терминах которых может идти речь о количественной сложности алгоритмов и задач. Более широкое и глубокое изложение этих вопросов можно найти в [3]; отметим также обзор [4] по методам анализа алгоритмов; интересные экскурсы в историю исследований сложности содержатся в [5].

Следующие далее разделы 2.2—2.4 также являются вспомогательными и служат фундаментом для изложения собственно результатов о *NP*-полноте задач составления расписаний (разделы 2.5—2.7). Наше описание машин Тьюринга, классов P и NP, NP-полноты следует книге [6]. Фундаментальная теорема Кука о *NP*-полноте задачи выполнимости булевой формулы также изложена по [6].

Основные результаты о *NP*-полноте простейших сетевых задач теории расписаний (раздел 2.5) получены Ульманом [7, 8]; *NP*-полнота задач типа «станки — детали» (раздел 2.6) показана в [9]; *NP*-полнота сетевых задач с векторными потребностями в ресурсах показана в [10, 11]. Проблематика *NP*-полноты в целом с исчерпывающей полнотой изложена в недавней книге [12]. Обзор [13] освещает результаты, относящиеся к традиционным задачам теории расписаний.

ГЛАВА III

ВАРИАЦИИ ДЛИТЕЛЬНОСТИ КРАТЧАЙШИХ РАСПИСАНИЙ. ПРИБЛИЖЕННЫЕ АЛГОРИТМЫ С ОЦЕНКОЙ ПОГРЕШНОСТИ

3.1. Алгоритмы с оценками

Как показано в главе II, многие сетевые задачи оптимального распределения нескладируемых ресурсов являются NP-полными. Очевидные вычислительные трудности численного решения NP-полных задач оптимизации стимулировали исследования, направленные на поиски эффективных (полиномиальных) приближенных алгоритмов. Целью приближенных алгоритмов, вообще говоря, является не нахождение оптимального решения задачи при любых входных данных, а гарантия того, что найденное решение в определенном смысле близко к оптимуму. Например, может утверждаться, что результатом алгоритма является решение задачи, которое по своему значению никогда, т. е. ни при каком допустимом наборе входных данных задачи, не отличается от оптимального более чем на фиксированную мультипликативную или аддитивную константу. Для заданного приближенного в указанном смысле алгоритма возникает естественный вопрос: какова наилучшая (наименьшая) граница погрешности алгоритма, которая может быть доказана? Чтобы оценить, достижима ли граница, можно попытаться сконструировать входные данные, на которых данный алгоритм приводит к наихудшим результатам, т. е. к наибольшей погрешности по функционалу. Если могут быть построены такие примеры входных данных, на которых некоторый алгоритм дает погрешность в пределах доказанных границ, то эти границы являются наилучшими (точными) и говорят, что найдена характеристика алгоритма в худшем. Во всяком случае, подобные примеры дают пользователю ориентир, отмечая, как далеко от действительного оптимума может находиться решение задачи, найденное посредством данного алгоритма. Таким образом, данный подход связан с получением гарантированных оценок погрешности алгоритмов оптимизации и характеризует поведение алгоритмов в худшем.

Уточним основные идеи применительно к задаче составления оптимального расписания. Каждую задачу удобно представить состоящей из двух частей: модели и целевой функции. Модель задачи представляет собой формализованное описание множества индивидуальных операций и условий допустимости расписаний. Целевая функция определяет значение каждого допустимого расписания. Вход задачи представляет собой описание в терминах модели некоторой конкретной задачи данного класса. Для заданного входа I значение оптимального расписания обозначается $OPT(I)$ и равно минимальному (иногда максимальному) значению целевой функции на множестве допустимых расписаний. Оптимальное расписание — это допустимое расписание для I, значение которого равно $OPT(I)$. Посредством алгоритма A в задаче составления расписания строится некоторое допустимое расписание для каждого входа I. Пусть $A(I)$ равно значению допустимого расписания, найденного посредством алгоритма A на входе I. Если $A(I)$ всегда, т. е. на всех допустимых I, равно $OPT(I)$, то A называется точным алгоритмом, иначе A — приближенный алгоритм. Оценки поведения приближенного алгоритма в худшем (гарантированные оценки погрешности) обычно имеют вид теорем, устанавливающих верхние границы погрешности конкретного приближенного алгоритма. Для задачи на минимум соответствующая теорема состоит в утверждении, что для всех входов I имеет место $A(I) \leqslant r \cdot OPT(I) + d$, где $r \geqslant 1$ и d — определенные константы. При этом доминирующим фактором является отношение r, а константа d во многих случаях равна 0 или асимптотически равна 0. Анализируя некоторый алгоритм, мы стремимся найти наименьшее r, для которого может быть доказано соответствующее утверждение, т. е. найти наилучшую гарантированную оценку для этого алгоритма. Далее, мы можем утверждать, что не существует лучшей гарантированной оценки погрешности, если сконструируем хотя бы один вход задачи, на котором погрешность алгоритма равна доказанному значению верхней границы погрешности. Точнее говоря, если мы можем показать, что для некоторой константы d' и всех N найдется вход I такой, что $OPT(I) > N$ и $A(I) > r \cdot OPT(I) - d'$, то действительно не существует оценки, в которой $r < r'$.

Скажем, что определено поведение некоторого алгоритма в худшем, если значение относительной погрешности r в верхней границе для $A(I)$ равно достигаемому

значению r'. Это общее значение относительной погрешности называется оценкой в худшем приближенного алгоритма. Оценки в худшем, как правило, сильно завышены для почти всех входов задачи и достигаются на отдельных специально подобранных «неестественных» входах. Однако они справедливы для всех входов задачи. В настоящей главе нашли отражение некоторые результаты об оценках в худшем для сетевых задач с неравнодлительными операциями, а также оценки в задачах с равнодлительными операциями. Кроме того, для задач типа «станки — детали» приводятся некоторые результаты, позволяющие построить в определенном смысле асимптотически оптимальные алгоритмы в указанных задачах. Приближенные алгоритмы этого типа имеют абсолютную погрешность по функционалу (в задачах построения кратчайшего расписания), не зависящую от числа работ. Другими словами, когда число работ (т. е. «деталей» в задаче «станки — детали») стремится к ∞, относительная погрешность стремится к 0.

3.2. Вариации длительности приоритетных расписаний

3.2.1. Приоритетные (списочные) расписания. Алгоритмы, о которых пойдет речь, будут исследоваться применительно к задачам построения кратчайших мультипроцессорных расписаний, описанных в разделе 1.4. Приоритетное (перестановочное, списочное) расписание определяется как функция заданного списка операций

$$\sigma = (v_1, \ldots, v_n).$$

В простейших задачах составления мультипроцессорных расписаний единственным видом нескладируемых ресурсов являются процессоры. Положим, что имеется несколько идентичных процессоров с номерами $p = 1, \ldots, P$. Рассмотрим следующую условную схему организации процесса выполнения операций. В каждый момент времени каждый из процессоров может выполнять или не выполнять одну из операций. В начальный момент времени $t = 0$ все процессоры свободны и по крайней мере некоторые из операций могут быть начаты. Если в момент времени $t \geqslant 0$ некоторый процессор свободен (не выполняет ни одну из операций), он мгновенно просматривает заданный список операций σ и выбирает первую по порядку операцию v_i из числа тех, которые могут быть начаты в данный момент времени. Тогда t_i — время начала вы-

бранной операции — полагается равным t. По определению допустимого расписания, в момент времени t могут быть начаты такие операции v_j, что все v_i, $v_i \prec v_j$, уже завершены, т. е. они были начаты в момент времени t_i, $t_i + \tau_i \leqslant t$, и такие, что они не выполняются уже каким-нибудь другим процессором. Если в данный момент времени свободны два или несколько процессоров, полагаем, что у них различные приоритеты (например, процессор с меньшим порядковым номером первым просматривает список операций). Время завершения всех операций ω определяется как время окончания последней операции и зависит от списка σ, отношения предшествования $\prec$, числа процессоров m и длительностей операций τ_i:

$$\omega = \omega(\sigma;\ \prec;\ m;\ \tau_i,\quad i = 1, \ldots, n).$$

Расширенная (обобщенная) модель задачи мультипроцессорного расписания включает в себя также информацию о ресурсах, отличных от процессоров. Положим, что $V = \{v_i\}_{i=1}^n$ — множество операций; $k = 0, 1, \ldots, s$ — индексы нескладируемых видов ресурсов; $k = 0$ соответствует процессорам; $r_i^k = r^k(v_i)$, $i = 1, \ldots, n$; $k = 0, \ldots, s$,— потребности операций в ресурсах, причем $r_i^0 = 1$, $i = 1, \ldots, n$, $0 \leqslant r_i^k \leqslant 1$, для всех остальных k; общие количества ресурсов каждого вида соответственно равны

$$r_0^0 = m, \quad r_0^k = 1, \qquad k = 1, \ldots, s.$$

Уточним некоторые термины. Для данного списка операций σ отображение

$$\Delta_\sigma\colon\ V \to 2^{[0,\ \omega)}$$

определяет интервал $\Delta_\sigma(v_i) = [t_i(\sigma),\ t_i(\sigma) + \tau_i)$, где $t_i(\sigma)$ — время начала операции v_i в расписании, сопоставленном списку σ. Путь также отображение

$$V_\sigma\colon\ [0,\ \omega) \to 2^V$$

определено как

$$V_\sigma(t) = \{v_i \in V \mid t \in \Delta_\sigma(v_i)\},$$

т. е. $V_\sigma(t)$ — это множество операций, выполняемых в момент времени t в расписании, сопоставленном списку σ. Тогда общие уровни использования ресурсов выражаются как

$$r_\sigma^k(t) = \sum_{v_i \in V_\sigma(t)} r^k(v_i), \quad k = 0, \ldots, s, \quad t \in [0,\ \omega),$$

а ограничения на общие уровни использования ресурсов выражаются как

$$r_{\sigma}^{0}(t) \leqslant m \tag{3.1}$$

для процессоров

$$r_{\sigma}^{k}(t) \leqslant 1, \quad k = 1, \ldots, s, \tag{3.2}$$

для остальных видов нескладируемых ресурсов, учитываемых в обобщенной задаче составления мультипроцессорного расписания.

Для обобщенной модели следует также уточнить термин «списочное расписание». А именно, будем считать, что освободившийся в момент времени t процессор выбирает первую по порядку операцию такую, что все предшествующие ей операции уже завершены и, кроме того, для множества операций $V_{\sigma}(t)$, включающего выбранную операцию v_j, должно выполняться условие (3.2). Заметим, что, поскольку в момент времени t имеется свободный процессор, условие (3.1) автоматически выполняется.

Далее в этой главе приводятся некоторые результаты, связанные с задачей построения кратчайшего списочного расписания. По определению, для произвольного списка σ $\omega = \omega(\sigma)$ — время выполнения всех операций в соответствии со списком σ. Пусть также $\omega^* = \omega(\sigma^*)$ обозначает минимальное значение $\omega(\sigma)$ на множестве всех списков σ; σ^* — список, на котором достигается значение ω^*. Заметим, что случай $m \geqslant n$ эквивалентен тому, что общее количество процессоров не лимитировано.

3.2.2. Задачи с неравнодлительными операциями.

Теорема 3.1. *В случае* $s = 1$ *и произвольных* σ, $\prec$, τ_i *и* m *выполняется неравенство*

$$\frac{\omega}{\omega^*} \leqslant m. \tag{3.3}$$

Доказательство. Достаточно заметить, что

$$\omega \leqslant \sum_{i=1}^{n} \tau_i \leqslant m\omega^*,$$

поскольку в каждый момент времени из интервала $[0, \omega)$ хотя бы один процессор занят, а общее число занятых процессоров в каждый момент времени не превосходит m.

Следующий пример показывает, что оценка (3.3)— наилучшая из возможных.

Пример. Пусть

$$V = \{v_1, \ldots, v_m, v_1', \ldots, v_m'\}, \quad s = 1;$$

$$\tau_i = 1, \quad \tau_i' = \varepsilon > 0;$$

$$r^1(v_i) = \frac{1}{m}, \quad r^1(v_i') = 1, \quad i = 1, \ldots, m.$$

Отношение $\prec$ таково:

$$v_i' \prec v_i, \quad 1 \leqslant i \leqslant m.$$

Рассмотрим два списка:

$$\sigma = (v_1, \ldots, v_m, v_1', \ldots, v_m'), \quad \sigma = (v_1', \ldots, v_m', v_1, \ldots, v_m).$$

Очевидно,

$$\omega = m + m\varepsilon, \quad \omega^* = \omega' = 1 + m\varepsilon.$$

Следовательно,

$$\frac{\omega}{\omega^*} = \frac{m + m\varepsilon}{1 + m\varepsilon} \to m, \quad \text{когда} \quad \varepsilon \to 0.$$

Теорема 3.2. *В случае пустого отношения* $\prec$, $m \geqslant n$ *и произвольных* s, σ, τ_i *выполняется неравенство*

$$\frac{\omega}{\omega^*} \leqslant s + 1.$$

Для доказательства теоремы нам понадобятся некоторые определения. Пусть $G = (V, E)$ — граф с множеством вершин $V(G)$ и множеством ребер $E(G)$. Функция λ: $V \to [0, \infty)$ определяет правильное помечивание G, если для всех пар вершин $(v_i, v_j) \in E$ имеет место $\lambda(v_i) + \lambda(v_j) \geqslant 1$. Ядро графа G обозначим $\rho^*(G)$ и определим как

$$\rho^*(G) = \inf_{\lambda} \sum_{v \in V} \lambda(v),$$

где операция inf определена на множестве правильных помечиваний λ.

Лемма 3.1. *Для каждого графа* G *существует правильное помечивание* λ: $V \to \{0, 1/2, 1\}$ *такое, что*

$$\rho^*(G) = \sum_{v \in V} \lambda(v).$$

Доказательство. В случае двудольного графа, когда G не имеет нечетных циклов, по теореме Кёнига количество ребер в максимальном паросочетании равно

минимальному числу вершин, инцидентных всем ребрам. Поэтому для двудольного графа G существует правильное помечивание λ: $V \to \{0, 1\}$ такое, что $\rho^*(G) = \sum_{v \in V} \lambda(v)$. В случае произвольного графа G построим двудольный граф G_B следующего вида: для каждой вершины $v \in V(G)$ создадим две вершины v_1, $v_2 \in V(G_B)$; для каждого ребра $(u, v) \in E(G)$ создадим два ребра (u_1, v_2), $(u_2, v_1) \in E(G_B)$. Нетрудно проверить, что $\rho^*(G_B) = 2\rho^*(G)$; более того, если λ_B: $V(G_B) \to \{0, 1\}$ — правильное помечивание G_B, то λ: $V(G) \to \{0, 1/2, 1\}$, $\lambda(v) = \frac{1}{2}(\lambda_B(v_1) + \lambda_B(v_2))$ — правильное помечивание G.

Для положительных целых чисел k и s пусть $G(k, s)$ обозначает граф с множеством вершин $\{0, 1, \ldots, (s+1)k-1\}$ и множеством ребер, состоящим из всех пар (a, b), для которых $|a - b| \geqslant k$.

Лемма 3.2. *Пусть граф $G(k, s)$ разбит на s покрывающих подграфов H_i, $1 \leqslant i \leqslant s$. Тогда*

$$\max_{1 \leqslant i \leqslant s} \rho^*(H_i) \geqslant k. \tag{3.4}$$

Доказательство. Предположим, что лемма неверна, т. е. существует разбиение графа $G(k, s)$ на подграфы H_i, $1 \leqslant i \leqslant s$, такое, что $\rho^*(H_i) < k$ для всех i, $1 \leqslant i \leqslant s$. Тогда по лемме 3.1 для каждого i существует правильное помечивание λ_i: $V(H_i) \to \{0, 1/2, 1\}$ такое, что

$$\sum_{v \in V(H_i)} \lambda_i(v) = \rho^*(H_i) < k. \tag{3.5}$$

Пусть $A = \{a_1 < \ldots < a_p \mid \lambda_i(a_j) \leqslant 1/2$ для всех i, $1 \leqslant i \leqslant s\}$, и пусть $\hat{\rho}$ обозначает $\sum_{i=1}^{s} \rho^*(H_i)$.

Возможны три случая.

1. Пусть $p \leqslant k$. В этом случае имеем

$$\hat{\rho} \geqslant k(s+1) - p \geqslant k(s+1) - k = ks,$$

что противоречит (3.5).

2. Пусть $k < p < 2k+1$. Для каждого ребра (a_j, a_{j+k}), $1 \leqslant j \leqslant p-k$, должно существовать такое i, что $\lambda_i(a_j) + \lambda_i(a_{j+k}) \geqslant 1$. Следовательно,

$$\hat{\rho} \geqslant k(s+1) - p + (p-k) = ks,$$

что также противоречит (3.5).

3. Пусть $p \geqslant 2k+1$. Заметим, что для каждой вершины $v \in V(G(k, s))$ существует такое i, что $\lambda_i(v) \geqslant 1/2$. Предположим, что верно обратное утверждение, т. е. что $\lambda_i(v) = 0$ для всех i, $1 \leqslant i \leqslant s$. Тогда должно существовать такое a_j, что $|a_j - v| \geqslant k$. Но поскольку $\lambda_i(a_j) \leqslant 1/2$ для всех i, имеет место $\lambda_i(a_j) + \lambda_i(v) \leqslant 1/2$ для всех i, что неверно.

Пусть n_i для каждого i обозначает количество таких вершин v, что $\lambda_i(v) = 1$. Тогда имеем

$$|\{v|\lambda_i(v) > 0\}| \leqslant 2k - 1 - n_i,$$

поскольку иначе

$$\sum_{v \in V(H_i)} \lambda_i(v) \geqslant n_i \cdot 1 + (2k - 2n_i)\frac{1}{2} = k,$$

что противоречит (3.5). Следовательно,

$$\sum_{i=1}^{s} |\{v \mid \lambda_i(v) > 0\}| \leqslant (2k-1)s - \sum_{i=1}^{s} n_i. \tag{3.6}$$

Пусть q обозначает количество таких вершин v, что существует точно одно i, для которого $\lambda_i(v) > 0$. Тогда

$$\sum_{i=1}^{s} |\{v \mid \lambda_i(v) > 0\}| \geqslant 2(k(s+1) - q) + q. \tag{3.7}$$

Комбинируя (3.6) и (3.7), получаем

$$q \geqslant 2k + s + \sum_{i=1}^{s} n_i. \tag{3.8}$$

Можно положить, не теряя в общности, что если $\lambda_i(v) = 1$, то $\lambda_j(v) = 1$ для всех $j \neq i$. Следовательно, по определению n_i должно быть самое меньшее $(2k+s)$ вершин, скажем $b_1 < \ldots < b_{2k+s}$, таких, что $\sum_{i=1}^{s} \lambda_i(b_j) = 1/2$, т. е. для каждого b_j существует единственное λ_i такое, что $\lambda_i(b_j) = 1/2$ и $\lambda_l(b_j) = 0$ для всех $l \neq i$. Следовательно, если $|b_j - b_l| \geqslant k$, имеет место $\lambda_i(b_j) = \lambda_i(b_l) = 1/2$ для некоторого i. Поскольку $|b_1 - b_{2k+s}| \geqslant k$, существует i_0 такое, что $\lambda_{i_0}(b_1) = \lambda_{i_0}(b_{2k+s}) = 1/2$. Однако на том же основании мы должны также иметь $\lambda_{i_0}(b_{k+j}) = \lambda_{i_0}(b_1) = 1/2$ и $\lambda_{i_0}(b_{2k+s}) = \lambda_{i_0}(b_j) = 1/2$ для всех j, $1 \leqslant j \leqslant k+s$. Следовательно,

$$\rho(H_{i_0}) = \sum_{v \in V(H_{i_0})} \lambda_{i_0}(v) \geqslant (2k+s)\frac{1}{2} \geqslant k,$$

что также является противоречием. Это завершает доказательство леммы 3.2.

Перейдем к доказательству теоремы 3.2. Заметим, что в каждом списочном расписании время начала каждой операции данного списка операций является суммой длительностей некоторых других операций этого списка.

Не теряя в общности, можно положить, что $\omega^* = 1$. Предположим, что $\omega > s+1$. Далее предположим, что каждое τ_i можно записать как $\tau_i = k_i/k$, где k_i — положительное целое число. Следовательно, $k_i \leqslant k$, поскольку $\tau_i \leqslant \omega^* = 1$. Кроме того, для всех i, $1 \leqslant i \leqslant s$, каждое $r^i(t)$ — константа на каждом интервале $[l/k,\ (l+1)/k]$ и равно $r^i(l/k)$. Заметим, что поскольку отношение предшествования пусто и $m \geqslant n$, то для $t_1,\ t_2 \in [0,\ \omega)$ и $(t_2 - t_1) \geqslant 1$ мы должны иметь

$$\max_{1 \leqslant i \leqslant s} \{r^i(t_1) + r^i(t_2)\} > 1.$$

Иначе говоря, каждая операция, выполняемая в момент времени t_2, должна быть завершена в момент времени t_1 или ранее. Следовательно, для каждого i, $1 \leqslant i \leqslant s$, мы можем определить граф H_i следующего вида:

$$V(H_i) = \{0,\ 1,\ \ldots,\ (s+1)k - 1\},$$

$(a,\ b)$ — ребро графа H_i, если и только если

$$r^i\left(\frac{a}{k}\right) + r^i\left(\frac{b}{k}\right) > 1. \tag{3.9}$$

Заметим, что если $|a - b| \geqslant k$, то $(a,\ b)$ является ребром по крайней мере одного графа H_i. Следовательно, $G(k,\ s) \subseteq \bigcup\limits_{1 \leqslant i \leqslant s} H_i$. Из (3.9) следует, что отображение λ_i: $V(H_i) \to [0,\ \infty)$, определяемое как $\lambda_i(a) = r^i\left(\frac{a}{k}\right)$, является правильным помечиванием H_i. Поскольку $G \subseteq G'$ влечет $\rho(G) \leqslant \rho(G')$ и условие на r^i в (3.9) является строгим неравенством, то из леммы 3.2 следует, что

$$\max_{1 \leqslant i \leqslant s} \sum_{l=0}^{(s+1)k-1} r^i\left(\frac{l}{k}\right) = \max_{1 \leqslant i \leqslant s} \sum_{v \in V(H_i)} \lambda_i(v) > \max_{1 \leqslant i \leqslant s} \rho(H_i) \geqslant k.$$

Но мы должны иметь

$$\frac{1}{k} \sum_{l=0}^{(s+1)k-1} r^i\left(\frac{l}{k}\right) \leqslant \int_0^{\infty} r^i(t)\,dt \leqslant 1, \qquad 1 \leqslant i \leqslant s,$$

т. е.

$$\sum_{l=0}^{(s+1)k-1} r^i\left(\frac{l}{k}\right) \leqslant k, \quad 1 \leqslant i \leqslant s.$$

Это противоречие, и теорема 3.2, таким образом, доказана для случая $\tau_i = k_i/k$, где k, k_i — положительные целые числа. Отсюда следует, что теорема 3.2 верна, когда все τ_i — рациональные числа. Доказательство теоремы будет полностью завершено следующей леммой.

Лемма 3.3. *Пусть* $\tau = (\tau_1, \ldots, \tau_n)$ — *последовательность положительных действительных чисел. Тогда для каждого* $\varepsilon > 0$ *существует последовательность* $\tau' = (\tau'_1, \ldots, \tau'_n)$ *такая, что*

1) $|\tau'_i - \tau_i| < \varepsilon$ *для* $1 \leqslant i \leqslant n$;

2) *для всех* A, $B \subseteq \{1, \ldots, n\}$

$$\sum_{a \in A} \tau_a \leqslant \sum_{b \in B} \tau_b$$

если и только если

$$\sum_{a \in A} \tau'_a \leqslant \sum_{b \in B} \tau'_b;$$

3) *все* τ'_i — *положительные рациональные числа.*

Замечание. Условие 2) гарантирует, что порядок выполнения операций v_i в соответствии со списком σ — одинаковый для τ и τ'. Таким образом, если σ используется для выполнения V, один раз с длительностями τ_i, другой — с длительностями τ'_i, то соответствующие времена завершения ω и ω' удовлетворяют условию $|\omega - \omega'| \leqslant n\varepsilon$. Следовательно, если бы был пример системы V с $\omega/\omega^* > s+1$ и некоторыми из τ_i иррациональными, то можно было бы построить пример другой системы V', поменяв τ_i на рациональные τ'_i так, что соответствующие времена завершения ω^* и $\omega^{*'}$ удовлетворяли бы условиям $|\omega - \omega^*| \leqslant n\varepsilon$ и $|\omega' - \omega^{*'}| \leqslant n\varepsilon$. Поэтому, если ε достаточно мало, $\omega^*/\omega^{*'} > s+1$. Однако это противоречило бы тому, что уже доказано.

Лемму 3.3 влечет следующая лемма.

Лемма 3.3′. *Пусть* φ, *обозначает конечную систему неравенств в форме*

$$\sum_{i=1}^{n} a_i x_i \geqslant a_0 \text{ или } \sum_{i=1}^{n} a_i x_i > a_0,$$

где a_i — *рациональные числа. Для каждого* $\varepsilon > 0$, *если* φ *имеет действительное решение* $(x_1, \ldots, x_n)$, *то* φ *имеет рациональное решение* $(x_1', \ldots, x_n')$ *такое, что* $|x_i - x_i'| < \varepsilon$ *для всех* i.

Доказательство. Используем индукцию по n. Для $n = 1$ результат, очевидно, имеет место. Пусть φ — система неравенств с n переменными, разрешимая в действительных числах. Система φ естественно расщепляется на два класса: φ_0 — подмножество неравенств, содержащих x_n, и $\varphi_1 = \varphi - \varphi_0$. Каждое неравенство в φ_1 можно записать одним из четырех способов:

$$a\colon\ \alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i \leqslant x_n;$$

$$b\colon\ \alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i < x_n;$$

$$c\colon\ \beta_0 + \sum_{i=1}^{n-1} \beta_i x_i \geqslant x_n;$$

$$d\colon\ \beta_0 + \sum_{i=1}^{n-1} \beta_i x_i > x_n.$$

Для каждой пары неравенств, одно типа a и одно типа c, рассмотрим неравенство

$$e\colon\ \alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i \leqslant \beta_0 + \sum_{i=1}^{n-1} \beta_i x_i.$$

Аналогично, пары типа $\{a, d\}$, $\{b, c\}$ и $\{b, d\}$ приводят к неравенствам вида

$$f\colon\ \alpha_0 + \sum_{i=1}^{n-1} \alpha_i x_i < \beta_0 + \sum_{i=1}^{n-1} \beta_i x_i.$$

Пусть φ^* — множество неравенств типа e и f, которые мы получаем из φ_1. Поскольку по предположению $\varphi = \varphi_0 \cup \varphi_1$ имеет действительное решение $(x_1, \ldots, x_n)$, то $\varphi_0 \cup \varphi^*$ имеет действительное решение $(x_1, \ldots, x_{n-1})$. Но $\varphi_0 \cup \varphi^*$ включает только $(n-1)$ переменную, так что по предположению индукции $\varphi_0 \cup \varphi^*$ имеет рациональное

решение $(x'_1, \ldots, x'_{n-1})$ с $|x_i - x'_i| < \varepsilon'$ для всех i и каждого наперед заданного $\varepsilon' > 0$. Подставляя x'_i в a, b, c и d, получим множество неравенств

$$g\colon\ a' \leqslant x_n, \quad b' < x_n, \quad c' \geqslant x_n, \quad d' > x_n,$$

где a', b', c' и d' — рациональные числа. Поскольку x_i удовлетворяют e и f, имеем $a' \leqslant c'$, $b' < c'$, $a' < d'$, $b' < d'$. Следовательно, для каждого ε, если ε' достаточно мало, имеется рациональное x'_n, удовлетворяющее g, и $|x_n - x'_n| < \varepsilon$, что завершает доказательство леммы 3.3′, а следовательно, леммы 3.3 и теоремы 3.2.

Следующий пример показывает, что граница, указываемая теоремой 3.2, не может быть улучшена.

Пример. Пусть

$$V = \{v_1, \ldots, v_{s+1}, v'_1, \ldots, v'_{sN}\};$$

$$\prec\ = \varnothing; \quad m \geqslant s(N+1) + 1 = n;$$

$$\tau_i = 1, \quad 1 \leqslant i \leqslant s+1; \quad \tau'_i = \frac{1}{N}, \quad 1 \leqslant i \leqslant sN;$$

$$r^i(v_i) = 1 - \frac{1}{N}, \quad r^i(v_j) = \frac{1}{sN}, \quad j \neq i, \quad 1 \leqslant i \leqslant s;$$

$$r^i(v'_j) = \frac{1}{N}, \quad 1 \leqslant j \leqslant sN, \quad 1 \leqslant i \leqslant s;$$

$$\sigma = (v_1, v'_1, \ldots, v'_N, v_2, v'_{N+1}, \ldots, v_{k+1}, v'_{kN+1}, v'_{kN+2}, \ldots$$
$$\ldots, v'_{(k+1)N}, v_{k+2}, \ldots, v'_{sN}, v_{s+1});$$

$$\sigma' = (v'_1, v'_2, \ldots, v'_{sN}, v_1, v_2, \ldots, v_{s+1}).$$

Легко проверить, что в этом случае $\omega = s+1$, $\omega' = 1 + s/N$, так что ω/ω' (а следовательно, и ω/ω^*) как угодно близко к $s+1$, когда $N \to \infty$.

3.2.3. Задачи с равнодлительными операциями. Основная модель остается прежней, за исключением того, что длительности операций предполагаются равными 1. А именно, предполагается заданной система $(m, s, V, \prec, R)$, где m и s — положительные целые числа; $V = \{v_1, \ldots, v_i, \ldots, v_n\}$ — множество операций; $\prec$ — частичный порядок на V; R — векторная функция с компонентами, определенными на V,

$$R(v_i) = (r^1(v_i), \ldots, r^k(v_i), \ldots, r^s(v_i)).$$

Предполагается, что $0 \leqslant r^k(v_i) \leqslant 1$ для всех $i = 1, \ldots, n$; $k = 1, \ldots, s$. Расписание S в данном случае определяется

как конечная последовательность непустых подмножеств операций $V_1, \ldots, V_l, \ldots, V_\omega$ такая, что

1) $\bigcup_{l=1}^{\omega} V_l = V$ и $V_{l_1} \cap V_{l_2} = \varnothing$ для всех $l_1 \neq l_2$;

2) если $v_i \prec v_j$, $v_i \in V_{l_1}$, $v_j \in V_{l_2}$, то $l_1 < l_2$;

3) $|V_l| \leqslant m$ для всех l;

4) $\sum_{v_i \in V_l} r^k(v_i) \leqslant 1$ для всех k, l.

Число ω называется длительностью расписания S.

Интерпретация основных терминов следующая: m — количество процессоров; s — количество видов ресурсов; $r_i^k = r^k(v_i)$ — количество ресурсов k-го вида, требуемое для выполнения операции v_i; V_l — множество операций, выполняемых в l-м по порядку единичном интервале времени, т. е. в интервале $(l-1, l)$.

Для заданного списка операций

$$\sigma = (v_{i_1}, \ldots, v_{i_n})$$

расписание S формируется по следующим правилам.

Шаг 1. Положить $l \leftarrow 1$.

Шаг 2. Пусть $V_l \leftarrow \varnothing$.

Шаг 3. Если список пуст — останов. Иначе просмотреть список слева направо и найти первую по порядку операцию v_i такую, что, если мы положим $V_l \leftarrow V_l \cup \{v_i\}$, условия 2)—4) не будут нарушены; положить $V_l \leftarrow V_l \cup \{v_i\}$ и удалить операцию v_i из списка.

Шаг 4. Если $|V_l| = n$ или на шаге 3 не найдется ни одной подходящей операции v_i, то положить $l \leftarrow l+1$ и перейти к шагу 2. Иначе перейти к шагу 3.

Определим теперь ряд параметров операций и множеств операций. Для заданной операции параметры w и W определяются как

$$w(v_i) = \max_{1 \leqslant k \leqslant s} r^k(v_i),$$

$$W(v_i) = \sum_{k=1}^{s} r^k(v_i).$$

Пусть $A \subseteq V$ — множество операций. Параметры $r^k(A)$, $w(A)$, $W(A)$ определяются как

$$r^k(A) = \sum_{v_i \in A} r^k(v_i),$$

$$w(A) = \max_{1 \leqslant k \leqslant s} r^k(A),$$

$$W(A) = \sum_{k=1}^{s} r^k(A).$$

Следующие далее алгоритмы A, B, C генерируют списки, которые используются для построения расписаний по правилу, описанному выше.

A. Произвольный список. Алгоритм состоит в формировании произвольного списка операций.

B. Уровневый список. Алгоритм использует так называемую функцию уровня, определенную на множестве всех операций сети следующим образом: $H(v_i) = L$, если в заданной сети операций наибольшая цепь, начинающаяся с вершины v_i, состоит из L операций. Тогда уровневый алгоритм определяет список σ, причем соответствующий этому списку линейный порядок (который мы также будем обозначать через σ) задается так:

1) $v_i \sigma v_j$, если $H(v_i) > H(v_j)$;

2) $v_i \sigma v_j$, если $H(v_i) = H(v_j)$ и $i < j$.

C. Ресурсно убывающий список. В этом случае линейный порядок σ определяется такими условиями:

1) $v_i \sigma v_j$, если $w(v_i) > 1/2$ и $w(v_j) \leqslant 1/2$;

2) если 1) неприменимо, то $v_i \sigma v_j$, если $W(v_i) > W(v_j)$;

3) если 1) и 2) неприменимы, то $v_i \sigma v_j$, если $i < j$.

Далее будет приведено несколько теорем, характеризующих поведение алгоритмов A, B, C в худшем.

Пусть ω^* — длительность кратчайшего расписания, а ω_σ — длительность расписания, порожденного списком σ.

Теорема 3.3. *Если* $m \geqslant n$, *то*

$$\frac{\omega_\sigma}{\omega^*} = \frac{1}{2} s (\omega^* + 7s)$$

для произвольного списка σ.

Пусть S — расписание, порожденное списком σ:

$$S = (V_1, V_2, \ldots, V_\omega).$$

Установим несколько предварительных результатов.

Лемма 3.4. *Для каждой операции* v_{j_1} *существуют последовательность целых чисел* $a_q < a_{q-1} < \ldots < a_1$ *и цепь операций*

$$v_{j_q} \prec v_{j_{q-1}} \prec \ldots \prec v_{j_2} \prec v_{j_1}$$

такие, что

1) $v_{j_l} \in V_{a_l}$, $l = 1, \ldots, q$;

2) $w(v_{j_q}) > 1/2$, *если для всех* l, $1 \leqslant l \leqslant a_q$, *не выполнено* $w(V_l) > 1/2$;

3) *если* $l \neq a_i$, $i = 1, \ldots, n$, *и* $a_q < l < a_1$, *то* $w(V_l) > 1/2$.

Доказательство. Следующая процедура генерирует последовательность целых чисел $a_1, \ldots, a_q$ и цепь операций $v_{j_1}, \ldots, v_{j_q}$, удовлетворяющие требуемым условиям.

Шаг 1 (начало). Пусть a_1 таково, что

$$v_{j_1} \in V_{a_1}, \quad t \leftarrow 1, \quad f \leftarrow a_1.$$

Шаг 2 (условие останова). Если $w(v_{j_t}) > 1/2$ или $f = 1$, останов.

Шаг 3 (попытка найти такую операцию $v_{j_{(t+1)}} \in V_{f-1}$, что $v_{j_{(t+1)}} \prec v_{j_t}$). Если существует такое u, что $v_u \prec v_{j_t}$ и $v_u \in V_{f-1}$, то $f \leftarrow (f-1)$, $t \leftarrow (t+1)$, $j_t \leftarrow u$, $a_t \leftarrow f$, иначе $f \leftarrow f-1$.

Шаг 4. Перейти к шагу 2.

Условие 2) леммы удовлетворяется условием останова на шаге 2.

Лемма 3.5. *Для некоторого p найдется p независимых цепей операций $c_1, \ldots, c_p$ таких, что для некоторой цепи c_i вида*

$$v_{j_{i,q_i}} \prec v_{j_{i,q_i-1}} \prec \ldots \prec v_{j_{i,1}}$$

выполнены условия:

1) $w(v_{j_{i,q_i}}) > 1/2, \quad i = 1, 2, \ldots, (p-1)$;

2) *если V_l таково, что $V_l \cap c_i = \varnothing$ для $i = 1, 2, \ldots, p$, то $w(V_l) > 1/2$.*

Доказательство. Выберем произвольную операцию $v_{j_{1,1}} \in V_\omega$. Найдем цепь c_1, используя процедуру, описанную в доказательстве леммы 3.4. Пусть это цепь $v_{j_{1,q_1}} \prec v_{j_{1,q_1-1}} \prec \ldots \prec v_{j_{1,1}}$. Положим $v_{j_{1,q_1}} \in V_l$. Если $w(v_{j_{1,q_1}}) \leqslant 1/2$ или $l = 1$, то останов. Иначе выберем произвольную операцию $v_{j_{2,1}} \in V_{l-1}$. Построим повую цепь c_2 в соответствии с процедурой из леммы 3.4. Пусть c_2 такова: $v_{j_{2,q_2}} \prec v_{j_{2,q_2-1}} \prec \ldots \prec v_{j_{2,1}}$. Опять проверяем, имеет ли место $w(v_{j_{2,q_2}}) \leqslant 1/2$ или $v_{j_{2,q_2}} \in V_1$, и принимаем решение, строить ли c_3. Процесс повторяется до тех пор, пока не получим $w(v_{j_{p,q_p}}) \leqslant 1/2$ или $v_{j_{p,q_p}} \in V_1$, где $v_{j_{p,q_p}}$ — максимальный элемент в последней из полученных цепей. Получено p цепей $c_1, \ldots, c_p$ с длинами $q_1, \ldots, q_p$. Непосредственно проверяется, что они удовлетворяют условию 2) леммы 3.4.

Лемма 3.6. *Пусть* $\{j_{ih} | 1 \leqslant i \leqslant p,\ 1 \leqslant h \leqslant q_i\}$ — *те же, что и в условиях леммы* 3.5. *Тогда существуют целое число* k, $1 \leqslant k \leqslant s$, *и множество целых чисел* $D \subset \{1, 2, \ldots, p-1\}$ *такие, что*

1) $$r^h(v_{j_{i,q_i}}) > 1/2, \quad i \in D; \tag{3.10}$$

2) $$\sum_{i \in D} q_i + q_p \geqslant \frac{1}{s} \sum_{i=1}^{p} q_i. \tag{3.11}$$

Доказательство. Определим s множеств целых чисел

$$D_l = \{i | 1 \leqslant i \leqslant p-1,\ r^l(v_{j_{i,q_i}}) > 1/2\}, \quad l = 1, \ldots, s.$$

Поскольку $\bigcup_{l=1}^{s} D_l = \{1, 2, \ldots, p-1\}$, имеем

$$\sum_{l=1}^{s} \sum_{i \in D_l} q_i \geqslant \sum_{i=1}^{p-1} q_i.$$

Поэтому существует k такое, что

$$\sum_{i \in D_k} q_i \geqslant \frac{1}{s} \sum_{i=1}^{p-1} q_i.$$

Пусть $D = D_k$; тогда

$$\sum_{i \in D} q_i + q_p \geqslant \frac{1}{s} \sum_{i=1}^{p-1} q_i + q_p \geqslant \frac{1}{s} \sum_{i=1}^{p} q_i.$$

Лемма доказана.

Теперь мы в состоянии доказать теорему 3.3. Пусть

$$c = \left\{l | V_l \cap \left(\bigcup_{i=1}^{p} c_i\right) = \varnothing\right\}.$$

Очевидно,

$$\sum_{i=1}^{p} q_i + |c| = \omega. \tag{3.12}$$

Поскольку общее количество видов ресурсов в каждый момент времени равно s, имеем

$$\omega^* \geqslant \frac{\sum_{i=1}^{s} W(v_i)}{s} \geqslant \frac{1}{s}\left(\sum_{l \in c} W(V_l) + \sum_{i=1}^{p-1} W(v_{j_{i,q_i}})\right) \geqslant$$

$$\geqslant \frac{1}{s}\left(\sum_{l \in c} w(V_l) + \sum_{i=1}^{p-1} w(v_{j_{i,q_i}})\right).$$

В силу (3.10) и (3.11) имеем

$$\omega^* \geqslant \frac{1}{s}\left(\frac{1}{2}|c| + \frac{1}{2}(p-1)\right).$$

Следовательно,

$$2s\omega^* \geqslant |c| + p - 1. \tag{3.13}$$

В соответствии с леммой 3.6 существует множество D такое, что выполняются условия (3.10) и (3.11). Обозначим цепи c_i, $i \in D$, через $c_1', c_2', \ldots, c_{p_1}'$, где $p_1 = |D|$. Пусть длина цепи c_i' равна q_i', и пусть ее максимальная операция есть v_{t_i}. Из (3.11) следует, что

$$\sum_{i=1}^{p_1} q_i' + q_p \geqslant \frac{1}{s}\sum_{i=1}^{p} q_i. \tag{3.14}$$

Далее, ни одна пара операций из множества $\{v_{t_i},\ i = 1, \ldots, p_1\}$ не может выполняться одновременно в каждом расписании, поскольку каждая из них потребляет более чем 1/2 единиц k-го вида ресурсов. Можно положить без потерь в общности, что в оптимальном расписании операция v_{t_i} выполняется до $v_{t_{i+1}}$, $1 \leqslant i \leqslant p_1 - 1$. Если время отсчитывается от начала операции v_{t_1}, то последняя операция в цепи c_i' не может быть завершена ранее чем в $[q_i' + (i-1)]$-ю единицу времени. Следовательно,

$$\omega^* \geqslant \max\left(q_p, q_1', q_2' + 1, q_3' + 2, \ldots, q_{p_1}' + (p_1 - 1)\right) \geqslant$$

$$\geqslant \frac{1}{p_1 + 1}\left(q_p + q_1' + (q_2' + 1) + \ldots + (q_{p_1}' + (p_1 - 1))\right) =$$

$$= \frac{1}{p_1 + 1}\left(q_p + \sum_{i=1}^{p_1 - 1} q_i' + \frac{1}{2}p_1(p_1 - 1)\right) \geqslant$$

$$\geqslant \frac{1}{p_1 + 1}\left(\frac{1}{s}\sum_{i=1}^{p} q_i + \frac{1}{2}p_1(p_1 - 1)\right), \tag{3.15}$$

где на последнем шаге используется неравенство (3.14). Неравенство (3.15) можно записать в виде

$$s(p_1 + 1)\omega^* \geqslant \sum_{i=1}^{p} q_i + \frac{1}{2}sp_1(p_1 - 1). \tag{3.16}$$

Из (3.12), (3.13) и (3.16) непосредственно следует, что

$$-\frac{1}{2}sp_1(p_1-1)+s(p_1+1)\omega^*+2s\omega^*\geqslant\omega+p-1\geqslant\omega.$$

Последнее можно записать как

$$-\frac{1}{2}s\left(p_1-\left(\omega^*+\frac{1}{2}\right)\right)^2+3s\omega^*+\frac{1}{2}s\left(\omega^*+\frac{1}{2}\right)^2\geqslant\omega. \quad (3.17)$$

Так как ω^*, p_1 — целые числа, имеет место неравенство

$$\left|p_1-\left(\omega^*+\frac{1}{2}\right)\right|\geqslant\frac{1}{2}.$$

Поэтому (3.17) влечет

$$-\frac{1}{2}s\left(\frac{1}{2}\right)^2+3s\omega^*+\frac{1}{2}s\left(\omega^*+\frac{1}{2}\right)^2\geqslant 0.$$

Таким образом,

$$\frac{1}{2}s\omega^{*2}+\frac{7}{2}s\omega^*\geqslant\omega,$$

что завершает доказательство теоремы 3.3.

Теорема 3.4. *Существуют такие системы операций с как угодно большим* ω^* *и* $m\geqslant n$, *что*

$$\frac{\omega_\sigma}{\omega^*}\geqslant\frac{1}{2}s(\omega^*-2s)$$

для некоторого списка σ.

Доказательство. Необходимо построить систему операций и список σ такие, что

$$\frac{\omega_\sigma}{\omega^*}>\frac{1}{2}s\omega^*-s^2.$$

Пусть

$$V=\{v^k, v^k_{jl}\,|\,1\leqslant k\leqslant s,\ 1\leqslant j\leqslant q,\ 1\leqslant l\leqslant j\}.$$

Частичный порядок $\prec$ представлен на рис. 3.1.

Потребности операций в ресурсах таковы:

$$r(v^k)=(1,1,\ldots,1),\quad k=1,\ldots,s;$$

$$r(v^k_{i1})=\left(0,0,\ldots,0,1-\frac{\delta}{2^{i-1}},0,\ldots,0\right),$$

$$k=1,\ldots,s;\quad i=1,2,\ldots,q;$$

$$r(v^k_{ij})=\left(0,0,\ldots,0,\frac{\delta}{2^{i-1}},0,\ldots,0\right),$$

$$k=1,2,\ldots,s;\quad i=1,2,\ldots,q;\quad j=1,2,\ldots,i,$$

где δ — малое положительное число. Пусть $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_s$, где для всех $k = 1, 2, \ldots, s$

$$\sigma_k = (v^k, v^k_{11}, v^k_{21}, v^k_{22}, \ldots, v^k_{q1}, v^k_{q2}, \ldots, v^k_{qq}).$$

Для этого списка только одна операция выполняется в

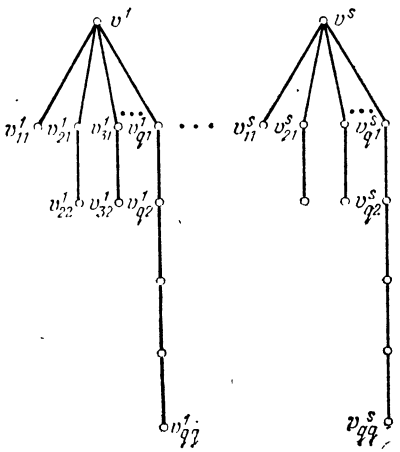

Рис. 3.1.

каждый момент времени. Поэтому длительность расписания равна $s(1+1+2+3+\ldots+q)$. Следовательно,

$$\omega_\sigma = s\left(1 + \frac{1}{2}q(q+1)\right).$$

С другой стороны, оптимальное расписание порождается списком

$$\sigma^* = (v^1, v^2, \ldots, v^s), \sigma'_1, \sigma'_2, \ldots, \sigma'_s$$

где для каждого k

$$\sigma'_k = (v^k_{q1}, v^k_{q2}, \ldots, v^k_{qq}, v^k_{q-1,1}, v^k_{q-1,2}, \ldots \ldots, v^k_{q-1,q-1}, \ldots, v^k_{21}, v^k_{22}, v^k_{11}).$$

В этом случае $\omega^* = q + s$. Следовательно,

$$\frac{\omega_\sigma}{\omega^*} = \frac{s\left(1 + \frac{1}{2}q(q+1)\right)}{q+s} > \frac{1}{2}s\omega^* - s^2.$$

Теорема 3.5. *Пусть список* σ *порожден уровневым алгоритмом. Тогда*

$$\frac{\omega_\sigma}{\omega^*} \leqslant \frac{m-1}{m}(2s+1)+1 \quad \textit{для произвольного } m, \tag{3.18}$$

$$\frac{\omega_\sigma}{\omega^*} \leqslant 2s+1 \quad \textit{для} \quad m \geqslant n. \tag{3.19}$$

Доказательство. Покажем справедливость лишь неравенства (3.19). Неравенство (3.18) может быть получено аналогичным путем. Рассмотрим некоторое расписание, порожденное уровневым алгоритмом. Пусть ω — длительность расписания, а V_1, V_2, ..., V_ω — образующие его подмножества операций. Введем функцию u следующего вида:

$$u(l) = \max_{v_i \in V_l} H(v_i), \qquad l = 1, 2, \ldots, \omega,$$

где H — функция уровня, определенная выше.

Лемма 3.7.

$$u(l) \geqslant u(l+1), \qquad l = 1, \ldots, (\omega - 1).$$

Доказательство. Предположим обратное. Тогда найдется такое l, что $u(l) < u(l+1)$. Пусть v_j из V_{l+1} — операция с $H(v_j) = u(l+1)$. Тогда $H(v_j) > H(v_i)$ для всех $v_i \in V$. Поэтому операция v_j должна предшествовать всем $v_i \in V_l$ в списке σ, поскольку он порожден уровневым алгоритмом. Далее, не найдется ни одной операции $v_i \in V_l$, для которой $v_i < v_j$. Следовательно, операция v_j должна быть выполнена не позднее каждой из операций в V_l. А это противоречит предположению, что $v_j \in V_{l+1}$.

Лемма 3.8. *Если* $u(l) = u(l+1)$, *то* $W(V_l) + W(V_{l+1}) > 1$.

Доказательство аналогично доказательству леммы 3.7.

Пусть теперь $A = \{l \mid u(l) = u(l+1)\}$; тогда

$$2\sum_{i=1}^{\omega} W(V_i) \geqslant \sum_{l \in A} (W(V_l) + W(V_{l+1})) > |A|.$$

Следовательно, имеет место неравенство

$$\omega^* \geqslant \frac{1}{s}\sum_{i=1}^{\omega} W(V_i) > \frac{1}{2s}|A|. \tag{3.20}$$

С другой стороны, в соответствии с леммой 3.7 имеем

$$u(l) \geqslant u(l+1)+1, \qquad l \in A, \quad 1 \leqslant l \leqslant \omega - 1.$$

Поэтому

$$\omega^* \geqslant u(1) \geqslant \omega - |A|. \tag{3.21}$$

Неравенства (3.20) и (3.21) влекут $(2s+1)\omega^* \geqslant \omega$. Это завершает доказательство теоремы 3.5.

Теорема 3.6. *Пусть список* σ *является ресурсно убывающим. Тогда*

$$\frac{\omega_\sigma}{\omega^*} \leqslant \frac{m-1}{m}\left(\frac{7}{4}s+1\right)+1 \quad \textit{для всех} \quad m, \tag{3.22}$$

$$\frac{\omega_\sigma}{\omega^*} \leqslant \frac{7}{4}s+1, \quad \textit{если} \quad m \geqslant n. \tag{3.23}$$

Доказательство. Покажем справедливость (3.23). Неравенство (3.22) устанавливается аналогично. Во-первых, разобьем множество $\{1, 2, \ldots, \omega\}$ на три части:

$A = \{p \mid 1 \leqslant p \leqslant \omega$, существует операция $v_i \in V_p$ такая, что $w(v_i) > 1/2\}$;

$B = \{p \mid 1 \leqslant p \leqslant \omega, \quad W(V_p) \geqslant 2/3\} \backslash A$;

$C = \{1, 2, \ldots, \omega\} \backslash \{A \cup B\}$.

Далее, определим подмножества операций

$$V_k = \{v_i \mid r^k(v_i) > 1/2\}, \qquad k = 1, \ldots, s.$$

Из определения A и V_k следует, что $\sum\limits_{k=1}^{s} |V_k|$ имеет по меньшей мере $|A|$ элементов:

$$\sum_{k=1}^{s} |V_k| \geqslant |A|.$$

Пусть d, $1 \leqslant d \leqslant s$, таково, что $|V_d| = \max\limits_{1 \leqslant k \leqslant s} |V_k|$. Тогда $|V_d| \geqslant \frac{1}{s}|A|$. Поскольку ни одна пара операций в V_d не может выполняться одновременно в каждом расписании, мы должны иметь

$$\omega^* \geqslant |V_d| \geqslant \frac{1}{s}|A|. \tag{3.24}$$

Из определения множеств A, B, C также следуют два новых неравенства:

$$\omega^* \geqslant \frac{1}{s}\left(\frac{1}{2}|A| + \frac{2}{3}|B|\right), \tag{3.25}$$

$$|A| + |B| + |C| = \omega. \tag{3.26}$$

Временно предположим, что справедливо неравенство

$$\omega^* \geqslant |C|. \tag{3.27}$$

Тогда из четырех неравенств (3.24)—(3.27) можно исключить $|A|$, $|B|$ и $|C|$, получив

$$\left(\frac{7}{4}s+1\right)\omega^* \geqslant \omega,$$

что является формулой, которую мы хотим доказать.

Покажем теперь справедливость неравенства (3.27).

Лемма 3.9. *Пусть* $l \in C$ *и* $v_i \in V_{l'}$, *где* $l' > l$. *Пусть также все* l'', $l < l'' < l'$, *не содержат операций* v_h, $v_h \prec v_i$. *Тогда должна существовать операция* $v_j \in V_l$ *такая, что* $v_j \prec v_i$.

Доказательство. Предположим обратное. Тогда в V_l должна найтись некоторая операция, которая в списке σ предшествует v_i. Покажем, что это приводит к противоречию.

Случай 1. Только одна операция v_p из V_l предшествует v_i в списке σ. Ясно, что $w(v_p) \leqslant 1/2$, поскольку $l \in C$. Далее, поскольку список σ — ресурсно убывающий, имеем

$$w(v_i) \leqslant W(v_i) \leqslant W(v_p) \leqslant \frac{1}{2}.$$

Следовательно, v_i могла бы выполняться одновременно с v_p. Поскольку v_i имеет приоритет по сравнению со всеми операциями в V_l, отличными от v_p, v_i следует выполнять одновременно с v_p. Это противоречие.

Случай 2. В списке σ имеется по меньшей мере две операции v_p и $v_{p'}$, предшествующие v_i. Опять имеем

$$W(v_i) \leqslant W(v_p), \quad W(v_i) \leqslant W(v_{p'}).$$

Следовательно,

$$W(v_i) \leqslant \frac{1}{2}(W(v_p) + W(v_{p'})) \leqslant \frac{1}{2}\sum_{v_j \in V_l} W(v_j),$$

$$W(v_i) + \sum_{v_j \in V_l} W(v_j) \leqslant \frac{3}{2}\sum_{v_j \in V_l} W(v_j) < \frac{3}{2}\cdot\frac{2}{3} = 1,$$

где на последнем шаге используется то, что $l \in C$. Значит, v_i могла бы выполняться одновременно со всеми v_j из V_l. Это противоречит тому, что $v_i \in V_{l'}$. Лемма доказана.

Лемма 3.10. *Существует цепь* $v_{j_q} \prec v_{j_{q-1}} \prec \ldots \prec v_{j_1}$ *такая, что для всех* $l \in C$ *в цепи найдется операция* $v_{j_p} \in V_l$.

Доказательство. Искомая цепь может быть легко сконструирована «снизу вверх» с использованием леммы 3.9.

Непосредственным следствием леммы 3.10 является то, что существует цепь длиной $|C|$ или более. Следовательно, $\omega^* \geqslant |C|$, что и завершает доказательство теоремы 3.6.

3.3. Один класс приближенных асимптотически оптимальных алгоритмов в задачах типа «станки — детали»

3.3.1. Постановка задачи. Рассматривается широкоизвестная задача календарного планирования следующего вида. Заданы m различных станков $i = 1, \ldots, m$ и n деталей $j = 1, \ldots, n$, подлежащих обработке на станках. Порядок обработки детали j определяется некоторой перестановкой $\sigma(j)$ на множестве станков. Последовательность $\sigma(j)$ называется технологическим маршрутом детали j. Длительность обработки детали j на станке i равна τ_{ij}, $i = 1, \ldots, m$; $j = 1, \ldots, n$. Требуется обработать заданную совокупность деталей за минимальное время, т. е. найти кратчайшее расписание при условиях:

1) каждый станок одновременно обрабатывает не более одной детали;

2) каждая деталь одновременно обрабатывается не более чем на одном станке;

3) каждая деталь обрабатывается последовательно на всех станках в соответствии со своим технологическим маршрутом;

4) технологические маршруты различных деталей одинаковы:

$$\sigma(j_k) = \sigma(j_l), \quad j_k \neq j_l; \quad j_k, j_l \in \{1, \ldots, n\};$$

5) начатые операции не прерываются.

Указанную задачу называют задачей Джонсона.

Нетрудно видеть, что задача Джонсона является специальной сетевой задачей распределения ограниченных нескладируемых ресурсов, в которой:

— имеется mn операций;

— τ_{ij} — длительность операции ij;

— имеется m видов ресурсов, по единице каждого вида;

— потребности операций в ресурсах могут быть описаны вектором размерности m, причем $r_{ij}^k = 1$, если $k = i$, иначе $r_{ij}^k = 0$, $i, k = 1, \ldots, m$; $j = 1, \ldots, n$;

— сеть операций состоит из n независимых цепей, каждая из m операций.

Как обычно, расписание обработки деталей на станках может быть определено в терминах сроков начала или завершения операций: $S = \{t_{ij}(S),\ i = 1, \ldots, m;\ j = 1, \ldots, n\}$. Пусть $\bar{S}$ — множество всех расписаний. Предполагается, что обработка каждой детали может быть начата в момент времени $t = 0$. Тогда, при соблюдении условий 1)—5), $T(S) = \max\limits_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant n}} (t_{ij}(S) + \tau_{ij})$ — длительность расписания S; $T^* = T(S^*) = \min\limits_{S \in \bar{S}} T(S)$ — длительность кратчайшего расписания. В теории календарного планирования, однако, принято описывать расписание в задаче Джонсона не как набор чисел t_{ij}, удовлетворяющий условиям 1)—5), а в виде совокупности из m последовательностей $\pi(i)$ обработки деталей на станках:

$$\pi(i):\ j_1, \ldots, j_n;\quad i = 1, \ldots, m,\quad j_k \in \{1, \ldots, n\}.$$

Нетрудно убедиться, что имеет место взаимно однозначное соответствие между двумя способами описания расписаний. Далее будем использовать второй способ, а именно описывать расписания в задаче Джонсона в виде набора из m перестановок $\pi(i)$, $i = 1, \ldots, m$. Заметим, что порядки обработки деталей на различных станках не обязаны совпадать, т. е. в оптимальном расписании может иметь место $\pi(i_k) \neq \pi(i_l)$, когда $i_k \neq i_l$.

Далее, нам понадобится один специальный случай задачи Джонсона, когда дополнительно к условиям 1)—5) предполагается, что $\pi(i_k) = \pi(i_l)$, $i_k \neq i_l$. Такую задачу называют задачей очередности.

Как показано в главе II, задача Джонсона является *NP*-полной, а в исследовании алгоритмов для решения *NP*-полных задач оптимизации представляют интерес эффективные приближенные методы, допускающие оценку погрешности по функционалу. Некоторые оценки такого сорта в сетевых задачах общего вида уже были представлены в данной главе. Ниже будет представлен один способ построения алгоритмов в задаче Джонсона, приводящих к получению асимптотически оптимальных решений. Этот способ основан на использовании некоторых экстремальных свойств частичных сумм норми-

рованных векторов из $\mathbf{R}^m$ — евклидова m-мерного пространства.

3.3.2. Вспомогательная задача оптимального упорядочения векторов.

Теорема 3.7. *Пусть* $X = \{x_1, \ldots, x_N\}$ — *множество векторов* m-*мерного евклидова пространства* $\mathbf{R}^m$ *со скалярным произведением* $\langle x, y\rangle$ *и нормой* $\|x\| = \sqrt{\langle x, x\rangle}$ *такое, что* $\|x_j\| \leqslant 1$ *для всех* $j = 1, \ldots, n$, $\sum_{j=1}^{N} x_j = 0$. *Тогда существует такая константа* c_m, *зависящая только от размерности пространства* m, *что для некоторой перестановки* σ *индексов* 1, 2, ..., N *выполняется неравенство*

$$\max_{1 \leqslant r \leqslant N} \left\| \sum_{j=1}^{r} x_{\sigma(j)} \right\| \leqslant c_m. \tag{3.28}$$

Геометрический смысл теоремы 3.7 заключается в следующем: если длины звеньев замкнутой векторной ломаной не превосходят единицы, то их можно переставить так, что вся ломаная разместится внутри шара радиуса c_m.

Доказательство. Вначале докажем теорему для $m = 1$. При этом векторы x_i — просто числа. Выпишем столько положительных чисел, чтобы их сумма не превосходила 1. Затем выпишем столько отрицательных чисел, чтобы сумма всех выписанных чисел оказалась меньше 0, но не менее -1, затем опять вернемся к положительным числам и т. д., пока не исчерпаем все заданное множество из N чисел. Таким образом, для $m = 1$ имеет место $c_m = 1$.

Индукция. Пусть дано конечное множество векторов x_j, $j = 1, \ldots, N$, таких, что

$$\sum_{j=1}^{N} x_j = 0, \quad \langle x_j, x_j\rangle \leqslant 1.$$

Выберем подмножество векторов, сумма которых имеет максимальную длину. Пусть это будут векторы $x_1, \ldots, x_M$. Обозначим $\sum_{j=1}^{M} x_j = x_0$. Из определения x_0 следует, что $\langle x_0, x_0\rangle \geqslant \langle x_0 - x_j, x_0 - x_j\rangle$, $j = 1, \ldots, M$, откуда

$$2\langle x_0, x_j\rangle \geqslant \langle x_j, x_j\rangle > 0, \tag{3.29}$$

т. е. проекции всех x_j на x_0 имеют одно направление.

Аналогично имеем

$$2\langle x_0,\ x_j\rangle < -\langle x_j,\ x_j\rangle < 0 \qquad (3.30)$$

для $j=(M+1), \ldots, N$. Обозначим через y_j, $j=1, \ldots, N$, проекции векторов x_j на плоскость, ортогональную вектору x_0. Очевидно, что

$$\sum_{j=1}^{M} y_j = \sum_{j=M+1}^{N} y_j = 0, \qquad \langle y_j, y_j\rangle \leqslant 1.$$

Согласно допущению индукции можно упорядочить оба полученных множества так, что

$$\left\langle \sum_{j=1}^{p} y_j, \sum_{j=1}^{p} y_j \right\rangle \leqslant c_{m-1}^2,$$
$$\left\langle \sum_{j=M+1}^{s} y_j, \sum_{j=M+1}^{s} y_j \right\rangle \leqslant c_{m-1}^2, \quad 1 \leqslant p \leqslant M \leqslant s \leqslant N. \qquad (3.31)$$

Для упрощения записи будем считать, что они уже упорядочены таким образом. Если теперь векторы $\{y_j\}_{j=1}^{M}$ и $\{y_j\}_{j=M+1}^{N}$ разместить в общую последовательность, не нарушающую порядка внутри каждого множества, то длина любой усеченной суммы элементов последовательности будет не более $2c_{m-1}$. Обратимся к проекциям векторов x_j на x_0. Это будут векторы

$$x_j - y_j = \lambda_j x_0,$$

причем, согласно (3.29) и (3.30),

$$0 < \lambda_j \leqslant \frac{1}{\|x_0\|}, \qquad j = 1, \ldots, M,$$
$$-\frac{1}{\|x_0\|} \leqslant \lambda_j < 0, \qquad j = (M+1), \ldots, N, \qquad (3.32)$$

и множества положительных и отрицательных λ_j упорядочены так же, как и y_j. Расположим теперь оба множества λ_j в общую последовательность, не нарушая порядка в каждом из них, так, чтобы любая усеченная сумма векторов этой последовательности по модулю не превосходила $1/\|x_0\|$. Это и будет окончательное упорядочение множества векторов x_j: так как существует взаимно однозначное соответствие $x_j \leftrightarrow y_j \leftrightarrow \lambda_j$, то, упорядочивая одно из этих множеств, мы упорядочиваем и другое.

В заключение определим порядок роста константы c_m. Имеем

$$\left\langle \sum_{j=1}^{\nu} x_j, \sum_{j=1}^{\nu} x_j \right\rangle = \left\langle \sum_{j=1}^{\nu} y_j, \sum_{j=1}^{\nu} y_j \right\rangle + \\ + \left\langle \sum_{j=1}^{\nu} \lambda_j x_0, \sum_{j=1}^{\nu} \lambda_j x_0 \right\rangle, \quad \nu = 1, \ldots, N$$

(найденный порядок векторов мы снова записали как $x_1, \ldots, x_N$). Из (3.31) и (3.32) получаем $c_m^2 \leqslant 4c_{m-1}^2 + 1$, что в свою очередь влечет $c_m \leqslant \left(4c_{m-1}^2 + 1\right)^{1/2}$, т. е.

$$c_m \leqslant \left(\frac{4^m - 1}{3}\right)^{1/2}.$$

С другой стороны, можно показать, что $\lim\limits_{m\to\infty} c_m = \infty$. Действительно, для векторов $x_k \in \mathbf{R}^m$, $k = 1, 2, \ldots, (m+1)$, таких, что

$$\langle x_i, x_j \rangle = \begin{cases} -\dfrac{1}{m}, & i \neq j, \\ 1, & i = j, \end{cases}$$

и для любой перестановки σ имеем

$$\left\| x_{\sigma(1)} + \ldots + x_{\sigma\left(\left[\frac{m+1}{2}\right]\right)} \right\|^2 = \\ = \left[\frac{m+1}{2}\right] - \frac{\left[\dfrac{m+1}{2}\right]\left(\left[\dfrac{m+1}{2}\right] - 1\right)}{n} \geqslant \left[\frac{m+1}{2}\right]^2 \cdot \frac{1}{m} \geqslant \frac{m^2}{4}.$$

Таким образом, $c_m \geqslant \frac{1}{2} m^{1/2}$.

Замечание. Недавно установлены новые оценки константы c_m (константы Штейница): $c_m \leqslant m$ для любой, не обязательно симметричной нормы и $c_m \leqslant m/2$ в случае симметричной нормы [24].

3.3.3. Метод решения задачи упорядочения векторов. Ниже будет показано, что для приближенного решения некоторых задач типа «станки — детали» с погрешностью, не зависящей от числа работ (т. е. числа независимых цепей операций в задаче Джонсона), важно уметь решать следующую задачу: найти перестановку $\sigma^* = \sigma^*(X)$, для которой выполняется условие (3.28). Назовем ее задачей 1. Метод построения перестановки $\sigma(X)$ следует из доказательства теоремы 3.7. Опишем соответствующий алгоритм.

Обозначим исходные векторы через $x_1^1, x_2^1, \ldots, x_N^1$. Положим $V_1^1 = \{x_i^1\}_{i=1}^N$, $\Gamma_1^1 = \{1, \ldots, N\}$. Алгоритм состоит из 5 этапов. Этапы 1—4 выполняются при следующих значениях k, l: индекс k пробегает все целые числа от 1 до $m-2$; индекс l при каждом k изменяется от 1 до 2^{k-1}.

1. Из множества Γ_l^k выбирается подмножество Γ_{2l-1}^{k+1} такое, что величина $\|s_l^k\|$, где $s_l^k = \sum\limits_{i \in \Gamma_{2l-1}^{k+1}} x_i^k$, максимальна.

2. Определяется множество $\Gamma_{2l}^{k+1} = \Gamma_l^k \setminus \Gamma_{2l-1}^{k+1}$.

3. Строятся множества V_{2l-1}^{k+1} и V_{2l}^{k+1}, состоящие из векторов

$$x_i^{k+1} = x_i^k - \frac{\langle s_l^k, x_i^k \rangle s_l^k}{\langle s_l^k, s_l^k \rangle},$$

где $i \in \Gamma_{2l-1}^{k+1}$ либо $i \in \Gamma_{2l}^{k+1}$ соответственно. Размерность этих векторов — $(m-k)$.

4. Строятся последовательности M_{2l-1}^{k+1} и M_{2l}^{k+1} чисел

$$p_i^{k+1} = \frac{\langle s_l^k, x_i^k \rangle}{\|s_l^k\|},$$

где $i \in \Gamma_{2l-1}^{k+1}$ либо $i \in \Gamma_{2l}^{k+1}$ соответственно.

5. Пусть k пробегает все целые значения от $(m-1)$ до 2; индекс l при каждом значении k пробегает все целые значения от 1 до 2^{k-1}. Для каждой пары k, l множество $\Gamma_l^{k-1} = \Gamma_{2l-1}^k \cup \Gamma_{2l}^k$ упорядочивается так, чтобы порядок элементов множеств Γ_{2l-1}^k и Γ_{2l}^k не нарушался и для частичных сумм, соответствующих этому порядку, выполнялось неравенство

$$\left| \sum_j p_j^{k+1} \right| \leqslant 1.$$

Результатом работы алгоритма является новая последовательность Γ_1^1 чисел 1, 2, ..., N.

Сложность этого алгоритма определяется трудоемкостью решения следующей задачи: для произвольного множества $X \subset \mathbf{R}^m$ найти его подмножество M такое, что

$$\left\| \sum_{x \in M} x \right\| = \max_{s \subset X} \left\| \sum_{x \in s} x \right\|. \tag{3.33}$$

Назовем ее задачей 2. Семейство множеств M, удовлет-

воряющих условию (3.33), обозначим $\mathscr{M}(X)$. Анализ доказательства теоремы 3.7 показывает, что для построения перестановки $\sigma(X)$ полезно рассматривать множество D, обладающее следующими свойствами:

$$\left\langle y, \sum_{x \in D} x \right\rangle > 0, \quad y \in D,$$

$$\left\langle y, \sum_{x \in D} x \right\rangle < 0, \quad y \Subset D.$$

Семейство таких множеств будем обозначать $\mathscr{D}(X)$. Как следует из теоремы 3.7, $\mathscr{M}(X) \subset \mathscr{D}(X)$.

Построим алгоритм полиномиальной сложности для нахождения некоторого множества $D \subset \mathscr{D}(X)$. Обозначим через $F(X)$ семейство подмножеств $F \subset X \subset \mathbf{R}^m$, для которых существует такой элемент $h \in \mathbf{R}^m$, что

$$\langle h, x \rangle > 0, \quad x \in F,$$

$$\langle h, x \rangle \leqslant 0, \quad x \Subset F,$$

т. е. $F(X)$ является семейством подмножеств множества X, лежащих в каком-либо открытом полупространстве. Очевидно, $D(X) \subset F(X)$.

Теорема 3.8. *Если* $X \subset R^m$, $|X| = N \geqslant m - 1$, *то*

$$|F(X)| \leqslant \binom{N}{m-1} 2^m. \tag{3.34}$$

Доказательство. Предположим сначала, что любая система s, содержащая $(m-1)$ векторов из X, линейно независима. Для произвольного $s \subset X$ при $|s| = m - 1$ обозначим через A_s^1, A_s^2 множества элементов X, лежащих в открытых полупространствах, определяемых гиперплоскостью, порожденной множеством s. Тогда

$$F(X) \subset \{A_s^i \cup B \mid i = 1, 2;\ B \subset s \subset X,\ |s| = m - 1\}, \tag{3.35}$$

откуда следует (3.34).

Предположим теперь, что среди множеств $s \subset X$, $|s| = m - 1$, есть линейно зависимые системы. Пусть $H = \{h_1, h_2, \ldots, h_q\}$, $q = |F(X)|$,— набор векторов из $\mathbf{R}^m$ такой, что множества вида

$$F_k = \{x \in X \mid \langle h_k, x \rangle > 0\}, \quad 1 \leqslant k \leqslant q,$$

составляют семейство $F(X)$. Положим

$$L_1(k) = \{x \in \mathbf{R}^m \mid \langle x, h_k \rangle > 0\},$$

$$L_2(k) = \{x \in \mathbf{R}^m \mid \langle x, h_k \rangle < 0\}.$$

Обозначим через $\mathfrak{A}$ множество всех непустых областей вида $\bigcap_{k=1}^{q} L_i(k)$ при различных наборах i, k. Пусть $Y \subset \subset \mathbf{R}^m$ — некоторое множество, обладающее свойствами: а) любая система $(m-1)$ векторов из Y линейно независима; б) в каждой области $G \in \mathfrak{A}$ содержится не более одного элемента из Y; в) некоторый элемент $y \in Y$ содержится в области $G \in \mathfrak{A}$ тогда и только тогда, когда существует $x \in X$, содержащийся в G.

Из определения множества Y и из (3.35) следует

$$|F(X)| \leqslant |F(Y)| \leqslant \binom{N}{m-1} \cdot 2^m,$$

что доказывает теорему.

Пусть $M_0, M_1, \ldots, M_k$ — некоторая цепь подмножеств множества $X = \{x_1, \ldots, x_N\}$, причем каждое M_{i+1} получается из M_i добавлением или удалением одного элемента из X так, что

$$M_{i+1} = M_i \cup x, \quad \langle x, v_i \rangle > 0, \quad x \in M_i, \tag{3.36}$$

либо

$$M_{i+1} = M_i \backslash x, \quad \langle x, v_i \rangle \leqslant 0, \quad x \in M_i, \tag{3.37}$$

где $v_i = \sum_{x \in M_i} x$. Будем называть такую цепь максимальной, если ее нельзя удлинить с помощью правил (3.36) и (3.37).

Теорема 3.9. *Цепь множеств $M_0, M_1, \ldots, M_k$, удовлетворяющая соотношениям* (3.36), (3.37), *содержит не более $N^2\widetilde{M}^2/p^2$ множеств, где*

$$p = \min_{1 \leqslant j \leqslant N} \|x_j\|, \quad \widetilde{M} = \frac{1}{N} \sum_{j=1}^{N} \|x_j\|.$$

Если цепь максимальна, то $M_k \in \mathscr{D}(X)$.

Доказательство. Из определения множеств M_i следует, что для всех i от 0 до $k-1$ существует такое $x \in X$, что

$$\|v_{i+1}\|^2 > \|v_i\|^2 + \|x\|^2,$$

откуда

$$\|v_i\|^2 > ip^2, \quad i = 0, \ldots, k. \tag{3.38}$$

С другой стороны, для любых i

$$\|v_i\| \leqslant \sum_{i=1}^{N} \|x_j\| = N\widetilde{M}. \tag{3.39}$$

Из (3.38) и (3.39) следует первое утверждение теоремы. Второе утверждение непосредственно следует из определения множеств M_i.

Теорема 3.9 позволяет находить некоторое множество $D \subset \mathcal{D}(X)$ с помощью следующего простого алгоритма полиномиальной сложности. Выбирается произвольное подмножество M_0 множества X и затем по формулам (3.36), (3.37) строится цепь множеств M_i. Последнее множество этой цепи является искомым. В силу теорем 3.8 и 3.9 количество множеств в цепи не превышает величин $\frac{N^2\widetilde{M}^2}{p^2}$ и $\binom{N}{m-1}2^m$, а количество операций при построении одного множества M_i имеет порядок Nm.

3.3.4. Задача Джонсона и задача упорядочения векторов. В одномаршрутной задаче Джонсона предполагается, что каждая деталь проходит обработку на станках в заданной последовательности, одинаковой для всех деталей. В задаче очередности дополнительно предполагается, что порядок прохождения деталей через станки одинаков для всех станков. Пусть длительности операций описываются матрицей $T = (t_{ij})$, $i = 1, \ldots, m$; $j = 1, \ldots, N$, где t_{ij} — время обработки j-й детали на i-м станке. Обозначим через $L(T)$ длину кратчайшего расписания для одномаршрутной задачи с матрицей T, а через $D(T)$ — длину кратчайшего расписания для соответствующей задачи очередности.

Очевидно, имеют место неравенства

$$A \leqslant L(T) \leqslant D(T), \tag{3.40}$$

где

$$A = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{N} t_{ij}. \tag{3.41}$$

Осуществим сведение задачи Джонсона к задаче упорядочения векторов, рассмотренной в разделе 3.3.2. Соответствующий алгоритм решения задачи упорядочения даст верхнюю оценку длительности расписания B, $B \geqslant D(T)$, обладающую следующими свойствами. В классе задач, для которых все значения t_{ij} ограничены сверху константой K, величина $\Delta = B - A$ не зависит от N. Таким образом, оценки A и B для $L(T)$ и $D(T)$ являются асимптотически точными. Отсюда, в частности, следует, что длительности кратчайших расписаний задачи Джонсона и задачи очередности отличаются не более чем на Δ, не зависящую от N, поэтому для решения задачи

Джонсона с точностью Δ достаточно рассматривать соответствующую задачу очередности.

Пусть S — некоторое расписание в задаче Джонсона; $D_S(T)$ — длительность расписания S; $L(T)$ — длительность кратчайшего расписания; $D_0(T)$ — длительность кратчайшего расписания в задаче очередности с матрицей T. При построении верхней оценки для $L(T)$ и $D_0(T)$ окажется удобным рассматривать матрицы длительностей, у которых суммы элементов строк совпадают. Обозначим $K = \max\limits_{\substack{1\leqslant i\leqslant m\\ 1\leqslant j\leqslant N}} t_{ij}$.

Лемма 3.11. *Для любой матрицы T существует матрица $\bar{T} = (\bar{t}_{ij})$ такая, что*

$$\max_{\substack{1\leqslant i\leqslant m\\ 1\leqslant j\leqslant N}} \bar{t}_{ij} = K, \quad \sum_{j=1}^{N} \bar{t}_{ij} = A, \quad i = 1, 2, \ldots, m, \quad (3.42)$$

и для любого расписания S выполняется

$$D_S(T) \leqslant D_S(\bar{T}).$$

Доказательство. Для всех i, для которых $\sum\limits_{j=1}^{N} t_{ij} = A$, полагаем $\bar{t}_{ij} = t_{ij}$, $j = 1, \ldots, N$. Пусть для некоторого i имеет место $\sum\limits_{j=1}^{N} t_{ij} < A$. Поскольку $NK \geqslant A$, то найдется такое n, $0 < n \leqslant N$, что

$$(p-1)K + \sum_{j=n}^{N} t_{ij} < A, \quad nK + \sum_{j=n+1}^{N} t_{ij} \geqslant A.$$

Положим

$$\bar{t}_{ij} = \begin{cases} K, & i < n, \\ A - (n-1)K - \sum\limits_{j=n+1}^{N} t_{ij}, & i = n, \\ t_{ij}, & i > n \end{cases} \quad (3.43)$$

Очевидно, что построенная таким образом матрица $\bar{T}$ удовлетворяет условиям (3.42); кроме того, для всех i, j имеет место $\bar{t}_{ij} \geqslant t_{ij}$. Поэтому для любого расписания S выполняется $D_S(T) \leqslant D_S(\bar{T})$, что доказывает лемму. Таким образом, для получения верхней оценки для $D(T)$ и $D_0(T)$ достаточно оценить величины $D(\bar{T})$ и $D_0(\bar{T})$ соответственно.

Пусть, как и ранее, $T = (t_{ij})$ — матрица длительностей для задачи очередности, π — некоторая перестановка чи-

сел 1, 2, ..., N и $\{M\} = (M_1, M_2, \ldots, M_{m-1})$ — набор чисел таких, что $0 \leqslant M_1 \leqslant M_2 \leqslant \ldots \leqslant M_{m-1} \leqslant N$. Как известно, длина кратчайшего расписания при обработке деталей в порядке $\pi(1), \ldots, \pi(N)$ равна

$$D_\pi(T) = \max_{\{M\}} \left[t_{1,\pi(1)} + \ldots + t_{1,\pi(M_1)} + \right.$$
$$\left. + t_{2,\pi(M_1)} + \ldots + t_{2,\pi(M_2)} + t_{m,\pi(M_{m-1})} + \ldots + t_{m,\pi(N)}\right]. \tag{3.44}$$

Таким образом, $D(T) = \min_\pi D_\pi(T)$. Вычислим верхнюю оценку для $D(T)$.

Теорема 3.10. *Если в матрице* $T = (t_{ij})$ *с* $\max\limits_{\substack{1<i<m \\ 1<j<N}} t_{ij} =$ $= K$ *суммы элементов каждой строки равны* A, *то*

$$D(T) \leqslant A + \Delta,$$

где

$$\Delta = K(M-1)\lfloor c_{m-1}(m-1)^{1/2} + 1\rfloor, \tag{3.45}$$

c_m — *константа из теоремы* 3.7.

Доказательство. Положим

$$\alpha_{ij} = t_{i+1,\,j} - t_{ij}, \quad i = 1, \ldots, (m-1); \quad j = 1, \ldots, N. \tag{3.46}$$

Из (3.44) и (3.45) получаем

$$D_\pi(T) = A + \max_{\{M\}} \left[\left(\alpha_{1,\pi(M_1)} + \ldots + \alpha_{1,\pi(N)}\right) + \right.$$
$$+ \left(\alpha_{2,\pi(M_2)} + \ldots + \alpha_{2,\pi(N)} + \alpha_{N-1,\pi(M_{m-1})} + \ldots + \right.$$
$$\left.\left. + \alpha_{N-1,\pi(N)}\right) + \beta_1 + \ldots + \beta_{m-1}\right], \tag{3.47}$$

где величины $\beta_1, \ldots, \beta_{m-1}$ совпадают с некоторыми t_{ij}. Рассмотрим векторы

$$\alpha_k = \left(\alpha_k^1, \ldots, \alpha_k^{m-1}\right) \in \mathbf{R}^{m-1}, \quad k = 1, \ldots, N.$$

Из (3.42) и (3.46) следует

$$\sum_{k=1}^{N} \alpha_k = 0, \quad \|\alpha_k\|^2 = \langle \alpha_k, \alpha_k \rangle \leqslant K^2(m-1).$$

В силу теоремы 3.7 существует перестановка π, для которой

$$\left\| \sum_{k=r}^{N} \alpha_{\pi(k)} \right\| \leqslant c_{m-1} \max_{1<k<N} \|\alpha_k\| \leqslant$$
$$\leqslant c_{m-1} K (m-1)^{1/2}, \quad r = 1, \ldots, N.$$

Отсюда следует, что для всех $1 \leqslant i \leqslant M-1$

$$\alpha^i_{\pi(M_i)} + \ldots + \alpha^i_{\pi(N)} \leqslant c_{m-1} K (m-1)^{1/2}.$$

Из (3.42) и (3.47) получаем

$$D_\pi(T) \leqslant A + c_{m-1} K (m-1)^{3/2} + (m-1)K = \\ = A + K(m-1)[c_{m-1}(m-1)^{1/2} + 1],$$

что доказывает теорему.

Из теоремы 3.10 легко получается верхняя оценка длительности кратчайшего расписания $L(T)$ для задачи Джонсона.

Теорема 3.11. *Для задачи Джонсона с матрицей длительностей* $T = (t_{ij})$, $i = 1, \ldots, m$; $j = 1, \ldots, N$, *имеет место неравенство*

$$L(T) \leqslant A + \Delta, \tag{3.48}$$

где

$$\Delta = K(m-1)[c_{m-1}(m-1)^{1/2} + 1],$$

$$K = \max_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant N}} t_{ij}, \quad A = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{N} t_{ij}.$$

Доказательство. В силу леммы 3.11 для матрицы T существует такая матрица $\overline{T}$, что $D(T) \leqslant D(\overline{T})$ и выполнены условия (3.43). Применив к матрице $\overline{T}$ теорему 3.10 и учитывая, что $D(T) \leqslant D(\overline{T})$, получаем оценку (3.48).

Обозначим $B = A + \Delta$. Поскольку A является нижней оценкой для любого расписания, то

$$A \leqslant L(T) \leqslant D(T) \leqslant B.$$

Из (3.45) видно, что $\Delta = B - A$ не зависит от N, т. е. разность верхней и нижней оценок длительности расписания не возрастает с ростом N при фиксированном K.

Следствие 3.1. *Пусть* $L(T)$ *— длина кратчайшего расписания в задаче Джонсона;* $D(T)$ *— длина кратчайшего расписания в задаче очередности. Тогда* $D(T) - L(T) \leqslant \Delta$.

Таким образом, при поиске приближенного с точностью Δ расписания в задаче Джонсона можно ограничиться расписаниями в задаче очередности.

Библиографический комментарий

Проблема оценки погрешности эффективных приближенных алгоритмов в экстремальных комбинаторных задачах, в частности в задачах теории расписаний, возникла в связи с их *NP*-полнотой и отсутствием эффективных алгоритмов для точного решения этих

задач. Исчерпывающая мотивировка — в книгах [1, 2], а также в обзорах [3 — 6]. В теории расписаний эта проблема вначале исследовалась как проблема прогноза и стабилизации характеристик списочных мультипроцессорных расписаний. По-видимому, первые работы в этой области — [7, 8]. Первые результаты были расширены или уточнены в ряде последующих работ, среди которых отметим работы [9 — 13].

В содержании данной главы отображена лишь небольшая часть относящихся к теме результатов. В основном представлены задачи, в которых речь идет о сетях операций общего вида, а не деревьях или других специальных видах сетей, а также задачи с несколькими видами специализированных (невзаимозаменяемых) ресурсов. При указанных предположениях гарантированные оценки длительности кратчайших расписаний получены в [14] для случая неравнодлительных операций и в [15, 12] для единичных (равнодлительных) операций.

Другой тип оценок приближенных алгоритмов в экстремальных комбинаторных задачах связан с изучением поведения алгоритмов в среднем и асимптотического поведения. Содержание оценок этого класса, по-видимому, впервые определено в [4]. Некоторые результаты в конкретных задачах перечислены в обзоре [16]; отметим также работы [17 — 19]. В данной главе представлена лишь часть исследований, посвященных асимптотически оптимальным алгоритмам в задаче Джонсона [20—22], базирующихся на более раннем результате [23].

ГЛАВА IV

ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

4.1. Алгоритм динамического программирования

В этом разделе алгоритм динамического программирования будет описан на примере сетевой задачи определения кратчайшего расписания. В задаче полагается, что операции неделимы и имеют постоянные во времени уровни потребления ресурсов каждого вида. Принятые допущения, однако, не повлияют на структуру алгоритмов, о которых пойдет речь.

Как обычно, исходная сеть операций формально может быть представлена в виде ориентированного графа (V, E) без петель и контуров с множеством вершин V, отображающих заданное множество операций, и множеством дуг $E \subseteq V \times V$, отображающих заданное отношение предшествования операций. Таким образом, для каждой пары операций из V, связанных отношением предшествования, в сети операций (V, E) найдется дуга $(v_i, v_j) \in E$. Операции $v_i \in V$, $i = 1, \ldots, n$, характеризуются длительностями $\tau_i > 0$, τ_i — целые числа, и векторами потребностей в ресурсах $r_i = \{r_i^k, k = 1, \ldots, K\}$. Задан интервал планирования $[0, T]$; переменная времени $t \in [0, T]$ принимает целые значения. Поскольку операции неделимы, расписание S достаточно определить как вектор времен начала операций:

$$S = \{t_i(S), i = 1, \ldots, n\}, \quad t_i(S) \in [0, T].$$

Для заданного расписания представляют интерес функции $T(S)$ — длительность расписания и $r^k(t|S)$ — потребности в ресурсах k-го вида, $k = 1, \ldots, K$, в момент времени $t \in [0, T]$. Они определяются следующим образом:

$$T(S) = \max_{1 \leqslant i \leqslant n} (t_i(S) + \tau_i),$$

$$r^k(t \mid S) = \sum_{v_i \in V_t(S)} r_i^k, \qquad k = 1, \ldots, K,$$

где $V_t(S)$ — множество операций, выполняющихся в момент времени t:

$$V_t(S) = \{v_i | t_i(S) \leqslant t \leqslant t_i(S) + \tau_i\}.$$

Пусть также заданы общие уровни наличия ресурсов каждого вида во времени:

$$r_t = \{r_t^k, k = 1, \ldots, K\}, \quad t \in [0, T].$$

Далее будем полагать для простоты, что r_t не зависит от времени, и обозначать $r_0 = \{r_0^k, k = 1, \ldots, K\}$. Определяя задачу составления кратчайшего расписания и алгоритм динамического программирования, наряду с исходной сетью (V, E) будем использовать также расширенную сеть операций $G' = (V', E')$, которая может быть получена из (V, E) следующим образом:

1. Каждую операцию v_i сети (V, E) заменим τ_i равнодлительными операциями v'_{im} сети (V', E') так, что $\tau_{im} = 1$, $m = 1, \ldots, \tau_i$, а $r_{im}^k = r_i^k$ для всех $i = 1, \ldots, n$; $m = 1, \ldots, \tau_i$; $k = 1, \ldots, K$.

2. Отношение предшествования E заменим на E', в котором

$$v'_{im} E' v'_{i(m+1)}, \quad m = 1, \ldots, (\tau_i - 1), \ \forall\, i = 1, \ldots, n, \tag{4.1}$$

$$v'_{i\tau_i} E' v'_{j1} \Leftrightarrow v_i E v_j \quad \forall\, (v_i, v_j) \in E. \tag{4.2}$$

Таким образом, (V', E') также является ациклической сетью операций с общим числом вершин $n' = \sum_{i=1}^{n} \tau_i$. В дальнейшем нам понадобится различать в сети (V', E') пары операций, удовлетворяющих отношениям (4.1) и (4.2). Условимся обозначать E'_1 отношение (4.1) и E'_2 отношение (4.2). В отличие от операций сети (V, E), операции сети (V', E') условимся называть элементарными.

Задача построения кратчайшего расписания на сети операций (V', E') состоит в том, чтобы найти минимальное семейство непустых подмножеств $V'_t \subset V'$:

$$S: V'_1, \ldots, V'_t, \ldots, V_{T_1}$$

при следующих условиях:

$$\bigcup_{t=1}^{T_1} V'_t = V', \tag{4.3}$$

$$V'_{t_1} \cap V'_{t_2} = \varnothing, \quad t_1, t_2 = 1, \ldots, T_1, \quad t_1 \neq t_2, \tag{4.4}$$

$$r^k(V'_t) \leqslant r_0^k, \quad k = 1, \ldots, K, \tag{4.5}$$

где $r^k(V'_t) = \sum\limits_{v'_{im} \in V'_t} r^k_{im}$;

если $v'_1 \in V'_{t_1}$, $v'_2 \in V'_{t_2}$ и $v'_1 E'_2 v'_2$, то $t_1 \leqslant t_2$; (4.6)

если $v'_1 \in V'_t$ и $v'_1 E'_1 v'_2$, то $v'_2 \in V'_{t+1}$. (4.7)

Экстремальная задача минимизации T_1 при условиях (4.3)—(4.7) может быть сведена к задаче поиска кратчайшего пути между двумя выделенными вершинами графа $H = (W, F)$, однозначно определяемого посредством сети (V', E'). А именно, вершинами графа H являются некоторые подмножества единичных (элементарных) операций из V'; дуги графа H взаимно однозначно сопоставлены парам подмножеств операций, завершаемых в смежные интервалы времени. Построение кратчайшего расписания состоит в определении пути из начальной вершины графа H в конечную, состоящего из минимального числа дуг.

Существенным элементом алгоритмов данного класса является генератор правильных в определенном смысле подмножеств вершин сети (V', E'). Подмножество $w \subseteq V'$ вершин сети $G' = (V', E')$ называется правильным, если вместе с каждой вершиной v'_{jn} оно содержит все вершины v'_{im}, предшествующие v'_{jn} в G', т. е. $(v'_{jn} \in w) \wedge (v'_{im}, v'_{jn}) \in E')$ влечет $v'_{im} \in w$ (здесь $\wedge$ — логическое «и»). Заметим, что, хотя потребности операций в ресурсах не входят явно в определение правильного подмножества, с каждым правильным подмножеством связан вектор $R(w)$, равный сумме потребностей всех операций, содержащихся в w:

$$R(w) = \sum_{v'_{im} \in w} r_{im}.$$

Пусть $W = \{w_p, p = 1, \ldots, P\}$ — семейство всех правильных подмножеств вершин сети (V', E'), причем $w_0 = \varnothing$, $w_P = V'$. Основная проблема построения алгоритмов на сети H (условимся называть ее H-сетью) связана с тем, что общее число правильных подмножеств P может быть весьма большим по сравнению с размерами оперативной памяти ЭВМ, так что практически важным яв-

ляется вопрос экономного (в терминах операций ЭВМ) построения и анализа правильных подмножеств.

Процесс генерации всех правильных подмножеств состоит из нескольких этапов. На первом этапе выделяются все элементарные операции, не имеющие предшественников. Обозначим это множество элементарных операций через V_0'. Любое подмножество множества V_0' является правильным. Элементарные операции, входящие в V_0', считаются помеченными, а все остальные операции из V' — непомеченными. Непосредственный потомок правильного подмножества w_p определяется как элементарная операция, которая является непосредственным потомком по крайней мере одной элементарной операции из w_p и не является предшественником ни одной другой элементарной операции из w_p. Для любого правильного подмножества w_p l-го этапа, $l \geqslant 1$, его непосредственные потомки группируются в список, обозначаемый $F(w_p)$. Тогда для любого подмножества $V_p' \subset F(w_p)$, $w_p \cup V_p'$, есть правильное подмножество $(l+1)$-го этапа. Если на l-м этапе все правильные подмножества сформированы, то каждая элементарная операция в $F(w_p)$ оказывается помеченной. Когда все элементарные операции из V' помечены, генератор правильных подмножеств завершает свою работу.

Можно показать, что описанный выше генератор правильных подмножеств обладает следующими свойствами:

1) каждое правильное подмножество элементарных операций сети (V', E') порождается данным генератором;

2) ни одно правильное подмножество не порождается дважды;

3) все множества операций, порождаемые генератором, являются правильными;

4) если $\underline{t}_{im}$ — наиболее ранний срок начала операции v_{im}', то содержащие v_{im}' правильные подмножества впервые появляются на этапе t_{im};

5) общее число этапов равно количеству единиц времени в критическом пути сети (V', E'), т. е. количеству вершин в наибольшей по длине цепи вида

$$v_{i1}' \to \ldots \to v_{im}' \to v_{i(m+1)}' \to \ldots \to v_{jn}',$$

причем $(v_{im}', v_{i(m+1)}') \in E'$; v_{i1}' не имеет предшественников, v_{jn}' не имеет потомков.

Уточним определение H-сети (W, F). Вершинами ее являются все правильные подмножества элементарных

операций сети (V', E'). Пара вершин (w_p, w_q) связана дугой $(w_p, w_q) \in F$, если и только если выполнены условия:

1) w_p является собственным подмножеством w_q;

2) все предшественники каждой элементарной операции из w_q должны содержаться в w_p, т. е. если $v'_{jn} \in w_q$ и $v'_{im} E' v'_{jn}$, то $v'_{im} \in w_p$;

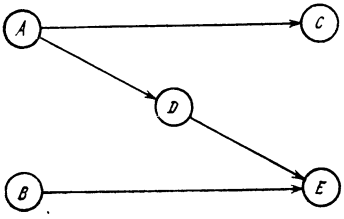

Рис. 4.1.

3) если $v'_{im} \in w_p$ и если $v'_{im} E'_1 v'_{jn}$, то $v'_{jn} \in w_q$ (условия непрерывности операций исходной сети);

4) $R(w_q) - R(w_p) \leqslant r_0$ (условия ограниченности ресурсов).

Дуги H-сети сопоставляются периодам времени длины 1; так, если выполнение элементарных операций из w_p завершается в момент времени t и в H-сети имеется дуга (w_p, w_q), то элементарные операции из $w_q - w_p$ могут выполняться в интервале времени $(t;\ t+1)$.

Рассмотрим иллюстративный пример. На рис. 4.1 изображена исходная сеть (V, E), состоящая из пяти операций. В табл. 4.1 указаны длительности и потребности операций в ресурсах. Наличные ресурсы системы таковы: $r_0^1 = r_0^2 = 5$, $r_0^3 = 3$. На рис. 4.2 изображена расширенная сеть (V', E'), в которой каждая операция сети (V, E), имеющая длительность $\tau_i > 1$, заменена цепью операций единичной длительности.

Таблица 4.1

Входные данные

i	τ_i	r_i^1	r_i^2	r_i^3
A	1	2	2	1
B	2	0	2	1
C	2	3	3	3
D	3	2	1	3
E	2	1	1	0

В табл. 4.2 представлен список правильных подмножеств вершин расширенной сети, т. е. список вершин H-сети. Использованы следующие обозначения: p — номер правиль-

Таблица 4.2

Список правильных подмножеств в расширенной сети

$M(p)$	$L(p)$	p	w_p	$R(p)$	$F(p)$
—	0	0	$\varnothing$	(0, 0, 0)	A, B1
A	1	1	{A}	(2, 2, 1)	C1, D1
B1		2	{B1}	(0, 2, 1)	B2
		3	{A, B1}	(2, 4, 2)	B2, C1, D1
B2	2	4	{A, C1}	(5, 5, 4)	C2
C1		5	{A, D1}	(4, 3, 4)	D2
D1		6	{A, C1, D1}	(7, 6, 7)	C2, D2
		7	{B1, B2}	(0, 4, 2)	
		8	{A, B1, B2}	(2, 6, 3)	
		9	{A, B1, C1}	(5, 7, 5)	C2
		10	{A, B1, D1}	(4, 5, 5)	D2
		11	{A, B1, B2, C1}	(5, 9, 6)	C2
		12	{A, B1, B2, D1}	(4, 7, 6)	D2
		13	{A, B1, C1, D1}	(7, 8, 8)	C2, D2
		14	{A, B1, B2, C1, D1}	(7, 10, 9)	C2, D2
C2	3	15	{A, C1, C2}	(8, 8, 7)	
D2		16	{A, D1, D2}	(6, 4, 7)	D3
		17	{A, C1, C2, D1}	(10, 9 10)	
		18	{A, C1, D1, D2}	(9, 7, 10)	D3
		19	{A, C1, C2, D1, D2}	(12, 10, 13)	D3
		20	{A, B1, C1, C2}	(8, 10, 8)	
		21	{A, B1, D1, D2}	(6, 6, 8)	D3
		22	{A, B1, B2, C1, C2}	(8, 12, 9)	
		23	{A, B1, B2, D1, D2}	(6, 8, 9)	D3
		24	{A, B1, C1, C2, D1}	(10, 11, 11)	
		25	{A, B1, C1, D1, D2}	(9, 9, 11)	D3
		26	{A, B1, C1, C2, D1, D2}	(12, 12, 14)	D3
		27	{A, B1, B2, C1, C2, D1}	(10, 13, 12)	
		28	{A, B1, B2, C1, D1, D2}	(9, 11, 12)	D3
		29	{A, B1, B2, C1, C2, D1, D2}	(12, 14, 15)	D3
D3	4	30	{A, D1, D2, D3}	(8, 5, 10)	
		31	{A, C1, D1, D2, D3}	(11, 8, 13)	
		32	{A, C1, C2, D1, D2, D3}	(14, 11, 16)	
		33	{A, B1, D1, D2, D3}	(8, 7, 11)	
		34	{A, B1, B2, D1, D2, D3}	(8, 9, 12)	E1
		35	{A, B1, C1, D1, D2, D3}	(11, 10, 14)	
		36	{A, B1, C1, C2, D1, D2, D3}	(14, 13, 17)	
		37	{A, B1, B2, C1, D1, D2, D3}	(11, 12, 15)	E1
		38	{A, B1, B2, C1, C2, D1, D2, D3}	(14, 15, 18)	E1

Продолжение табл. 4.2

$M(p)$	$L(p)$	p	w_p	$R(p)$	$F(p)$
E1	5	39	{A, B1, B2, D1, D2, D3, E1}	(9, 10, 12)	E2
		40	{A, B1, B3, C1, D1, D2, D3, E1}	(12, 13, 15)	E2
		41	{A, B1, B2, C1, C2, D1, D2, D3, E1}	(15, 16, 18)	E2
E2	6	42	{A, B1, B2, D1, D2, D3, E1, E2}	(10, 11, 12)	
		43	{A, B1, B2, C1, D1, D2, D3, E1, E2}	(13, 14, 15)	
		44	{A, B1, B2, C1, C2, D1, D2, D3, E1, E2}	(16, 17, 18)	

ного подмножества; $L(p)$ — номер этапа, на котором появляется правильное подмножество w_p; $M(p)$ — помеченные операции на шаге p; $R(p)$ — вектор потребляемых ресурсов; $F(p)$ — множество непосредственных потомков правильного подмножества w_p.

Используя табл. 4.2, проиллюстрируем несколько шагов генератора правильных подмножеств (п. п.). Вначале, на шаге 0 этапа 0, $p = L(p) = 0$, $w_p = M(p) = \varnothing$. Построим $F(0) = \{A, B1\}$. Этап 0 завершен. Каждое п. п. w_p этапа 1 образовано как $w_0 \cup V_0'$, где $V_0' \subseteq \{A, B1\}$. Для каждого w_p, $p = 1, 2, 3$, находились $R(p)$ и $F(p)$. Кроме того, каждый раз, когда к w_0 добавлялось по одной

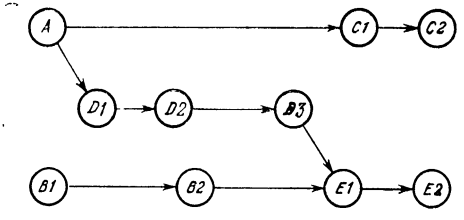

Рис. 4.2.

элементарной операции, эта операция помечалась. В результате помеченными оказались A и B1. Этап 1 завершен. На этапе 2 (шаги 4 — 14) процесс, описанный на этапе 1, повторяется для каждого п. п., построенного на этапе 1. В результате п. п. w_1 порождает п. п. w_4, w_5, w_6, $w_2 - w_7$, $w_3 - w_8$, ..., w_{14}.

На рис. 4.3 изображена H-сеть; номера правильных подмножеств указаны в вершинах H-сети; некоторые дуги опущены. В H-сети каждому пути вида

$$\varnothing = w_0 \to w_1 \to w_2 \to \ldots \to w_P = V'$$

взаимно однозначно соответствует некоторое расписание, причем операциям из w_1 сопоставлено время начала $t = 0$, операциям из $(w_2 - w_1)$ — время $t = 1$ и т. д.

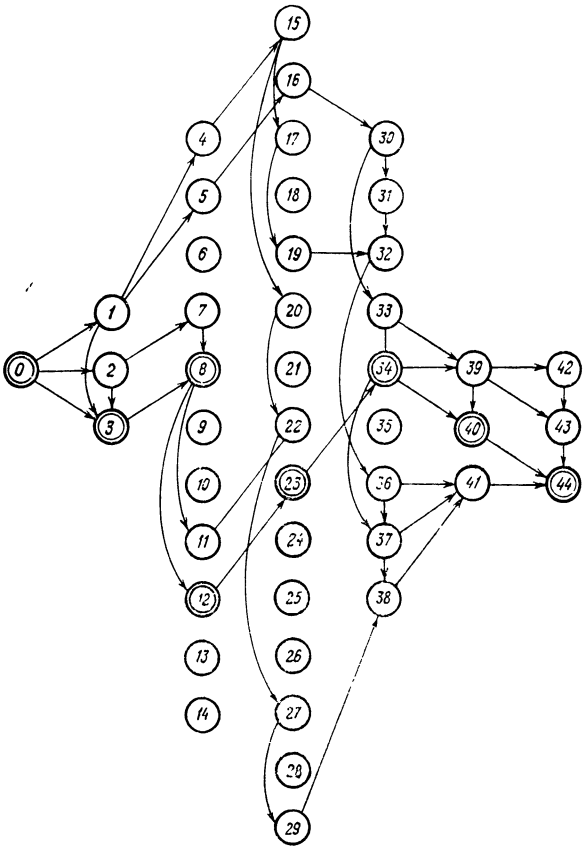

Рис. 4.3.

В H-сети на рис. 4.3, кратчайший путь из w_0 в w_P включает последовательность вершин (0, 3, 8, 12, 23, 34, 40, 44). Соответствующее расписание представлено в табл. 4.3. Таким образом, длительность кратчайшего расписания в данной задаче равна 7.

Замечание. Поскольку, как отмечалось, общее число правильных подмножеств может быть практически весьма большим, представляет интерес использование в генераторе правильных подмножеств специальных тестов-ограничителей для исключения некоторых подмножеств. Тесты должны гарантировать, что в процессе счета будет найдено хотя бы одно искомое расписание, если таковое

Таблица 4.3

Кратчайшее расписание

Время t	Номер вершины H-сети	Допустимые подмножества	Назначение операций
7	44	{A, B1, B2, C1, C2, D1, D3, E1, E2}	C2, E2
6	40	{A, B1, B2, C1, D1, D2, D3, E1}	C1, E1
5	34	{A, B1, B2, D1, D2, D3}	D3
4	23	{A, B1, B2, D1, D2}	D2
3	12	{A, B1, B2, D1}	D1
2	8	{A, B1, B2}	B2
1	3	{A, B1}	A, B1

существует в условиях применения тестов. Основной прием состоит в выборе «цели» — некоторого значения длительности расписания T_0 и поиске хотя бы одного расписания данной длительности. В процессе поиска могут быть исключены из дальнейшего рассмотрения те правильные подмножества вершин H-сети, которые

а) не могут принадлежать ни одному пути длины T_0 или менее;

б) не могут быть предшественниками правильных подмножеств, которые могли бы принадлежать пути заданной длины.

Если выбрана цель T_0, могут быть предложены два независимых множества тестов: первый использует информацию о длительностях и отношении предшествования операций, второй — информацию о потребностях операций в ресурсах.

Тест 1. *Использование отношения предшествования операций.* Известными методами расчета временны́х характеристик сетевых графиков можно определить $\bar{t}_i$ — наиболее поздний срок выполнения каждой операции расширенной сети при условии, что все операции выполняются не позднее T_0. В табл. 4.4 показаны наиболее поздние

времена выполнения операций расширенной сети при условии $T_0 = 6$.

Пусть теперь D_t равно множеству операций этой сети таких, что $\bar{t}_i \leqslant t$. В табл. 4.5 приведены множества D_t, соответствующие табл. 4.4. Из свойств генератора правильных подмножеств следует, что в H-сети число дуг в пути из вершины w_0 в вершину w_p, полученную на l-м этапе генератора, по меньшей мере равно l. Поскольку дуги

Таблица 4.4

Поздние времена выполнения операций

i	A	B1	B2	C1	C2	D1	D2	D3	E1	E2
$\bar{t}_i$	1	3	4	5	6	2	3	4	5	6

соответствуют единицам времени, каждое правильное подмножество w_p l-го этапа должно содержать все операции из D_l; в противном случае оно не может принадлежать полному пути из w_0 в w_P длиной T_0 или менее. Следовательно, если $D_l \nsubseteq w_p$, w_p может быть исключено. Нетрудно показать, что каждое правильное подмножество, являющееся потомком w_p, также не представляет интереса.

Тест 2. *Использование ресурсных ограничений.* Имея общее количество ресурсов, требуемое для выполнения совокупности всех операций расширенной сети,

Таблица 4.5

Множества D_t

t	D_t
1	{A}
2	{A, D1}
3	{A, B1, D1, D2}
4	{A, B1, B2, D1, D2, D3}
5	{A, B1, B2, C1, D1, D2, D3}
6	{A, B1, B2, C1, C2, D1, D2, D3, E1, E2}

можно для каждого t из интервала $[0, T_0]$ определить минимальный объем ресурсов каждого вида, который должен быть израсходован в подынтервале $[0, t]$, с тем чтобы общие потребности еще не выполненных к моменту

времени t операций были достаточно малыми и могли быть удовлетворены в оставшееся время, т. е. в интервале $[t+1, T_0]$. Пусть R^k — общие объемы потребления ресурсов k-го вида на всех операциях сети (V', E'):

$$R^k = \sum_{v_i \in V'} r_i^k, \quad k = 1, \ldots, K.$$

Тогда вектор

$$\underline{R}_t = (\underline{R}_t^k, \ k = 1, \ldots, K).$$

с компонентами

$$\underline{R}_t^k = \max\left[0, R^k - (T_0 - t)\, r_0^k\right]$$

определяет минимальные необходимые уровни использования ресурсов каждого вида в единичном интервале

Таблица 4.6

Значения ограничителей $\underline{R}_t$

t	1	2	3	4	5	6
$\underline{R}_t$	(0, 0, 3)	(0, 0, 6)	(1, 2, 9)	(6, 7, 12)	(11, 12, 15)	(16, 17, 18)

t, $t = 1, \ldots, T$, при условии, что все операции сети (V', E') завершаются не позднее T_0. Значения $\underline{R}_t$ могут применяться для исключения правильных подмножеств в процессе их порождения точно так же, как и при использовании критерия, основанного на поздних сроках и отношении предшествования. А именно, на этапе l генератора правильных подмножеств исключаются из дальнейшего рассмотрения (отсеиваются) такие правильные подмножества w_p, что $R(w_p) < \underline{R}_l$. Значения R_t в случае $T_0 = 6$ приведены в табл. 4.6.

Рассмотрим теперь выбор целевого срока T_0. Два типа тестов для исключения правильных подмножеств независимы и могут использоваться совместно. Следовательно, на этапе l генератора правильных подмножеств запоминаются только такие w_p, для которых имеет место

$$R(w_p) \geqslant \underline{R}_l, \quad D_l \subset w_p.$$

Обратимся к иллюстративной задаче. Как видно из табл. 4.5 и 4.6, использование ограничителей при $T_0 = 6$ не приводит к решению задачи, поскольку она не имеет

допустимого расписания с длительностью $T_0 = 6$. Положим $T_0 := T_0 + 1$ и попытаемся найти допустимое расписание длины 7. В табл. 4.7 приведены соответствующие значения $\underline{R}_t$ и D_t. Как мы уже знаем, в этом случае найдется допустимое расписание длины 7 (кратчайшее расписание), однако в общем случае также может не найтись допустимого расписания длины T_0 и текущее значение T_0 следует скорректировать.

Таблица 4.7

Значения ограничителей $\underline{R}_t$ и D_t в случае $T_0=7$

t	$\underline{R}_t$	D_t
1	(0, 0, 0)	
2	(0, 0, 3)	{A}
3	(0, 0, 6)	{A, D1}
4	(1, 2, 9)	{A, B, D1, D2}
5	(6, 7, 12)	{A, B1, B2, D1, D2, D3}
6	(11, 12, 15)	{A, B1, B2, C1, D1, D2, D3, E1}
7	(16, 17, 18)	{A, B1, B2, C1, C2, D1, D2, D3, E1, E2}

Альтернативный метод поиска минимального значения T_0, для которого существует расписание,— дихотомический. Пусть нам известно, что длительность кратчайшего расписания находится в пределах $[\underline{T}_0, \overline{T}_0]$. Установим некоторое T_0', $\underline{T}_0 \leqslant T_0' \leqslant \overline{T}_0$. Если существует расписание длины T_0', в следующей попытке устанавливаем $T_0'' = \left\lceil \frac{\underline{T}_0 + T_0'}{2} \right\rceil$, иначе $T_0'' = \left\rfloor \frac{T_0' + \overline{T}_0}{2} \right\lfloor$ и т. д.

Дальнейшего сокращения объема вычислений можно достигнуть за счет совмещения процесса построения и анализа правильных подмножеств с процессом динамического программирования. Состояния процесса динамического программирования образованы множеством вершин H-сети, а управления — множеством ее дуг. Определим соответствующее функциональное уравнение. Пусть $N_l(p)$ обозначает длину кратчайшего пути из вершины w_0 в вершину w_p, сгенерированную на l-м этапе генератора правильных подмножеств. Пусть, далее, множество M_l состоит из правильных подмножеств w_p этапа l, которые могут принадлежать пути (w_0, w_p), содержащему не более

чем T_0 дуг. Эти подмножества определяются как удовлетворяющие условиям

$$D_{l(p)} \subset w_p \text{ и } R(p) \geqslant \underline{R}_{l(p)},$$

где $l(p) = N_l(p)$.

Для вновь сгенерированной вершины w_q этапа l в поиске вершин w_p, из которых исходят дуги вида (w_p, w_q), можно ограничиться вершинами из списка M_{l-1}, а также уже сгенерированными вершинами из списка M_l. Функциональное уравнение динамического программирования имеет вид

$$N_l(q) = \min\Big(\min_{w_{p'} \in M_{l-1}} (\delta_{p'q} + N_{l-1}(p')\Big);$$

$$\min_{w_{p''} \in M_l} (\delta_{p''q} + N_l(p'')),$$

$$a \leqslant p' \leqslant b, \quad c \leqslant p'' \leqslant q,$$

где a — номер первой по порядку вершины этапа $(l-1)$; b — номер вершины-родителя вершины q (т. е. вершины этапа $(l-1)$, порождающей вершину q этапа l в генераторе правильных подмножеств); c — номер первой вершины этапа l; w_p — потенциальная вершина, из которой может исходить дуга, входящая в вершину w_q; $\delta_{pq} = 1$, если и только если выполнено каждое из условий:

1) $w_p \subset w_q$;
2) $v'_{im} \in w_p$, где $v'_{im} E' v'_{jn}$ и $v'_{jn} \in w_q$;
3) $v'_{jn} \in w_q$, где $v'_{im} \in w_p$ и $v'_{im} E'_1 v'_{jn}$;
4) $R(q) - R(p) \leqslant r_0$;

иначе $\delta_{pq} = \infty$.

Пусть для вершины w_p этапа l определено значение $N_l(p)$. Тогда w_p включается в список M_l, если и только если $R(p) > \underline{R}_{l(p)}$ и $D_{l(p)} \subset w_p$, где $l(p) = N_l(p)$.

Как и ранее, вычислительный процесс начинается с правильного подмножества $w_0 = \varnothing$. Вершины этапа 1 с $N_1(p) = 1$ включаются в M_1 так же, как и все вершины с $N_1(p) > 1$, прошедшие тесты $D_{l(p)}$ и $\underline{R}_{l(p)}$ для соответствующих значений $l(p) = N_1(p)$. Вершины с $N_1(p) > 1$, не прошедшие оба теста, нумеруются в порядке их порождения и используются для порождения вершин этапа 2, но их индексы не заносятся в список M_1. После

того как сгенерированы все вершины этапа 2, прошедшие тесты, вычисляется функция $N_2(p)$, причем вершины-кандидаты w_p на соединение дугой (w_p, w_q) с вершиной w_q выбираются из списка M_1, а также из уже сгенерированных вершин списка M_2. Вершины этапа 2, для которых

Таблица 4.8

Сокращенное множество правильных подмножеств иллюстративной задачи

$L(p)$	p	Правильные подмножества w_p	$\underline{R}(p)$	$N_l(p)$
0	0	$\varnothing$		
1	1	{A}	(2, 2, 1)	1
	2	{B1}	(0, 2, 1)	1
	3	{A, B1}	(2, 4, 2)	1
2	4	{A, C1}	(5, 5, 4)	2
	5	{A, D1}	(4, 3, 4)	2
	6	{A, C1, D1}	(7, 6, 7)	2
	7	{A, B1, B2}	(2, 6, 3)	2
	8	{A, B1, D1}	(4, 5, 5)	2
	9	{A, B1, B2, D1}	(4, 7, 6)	2
	10	{A, B1, C1, D1}	(7, 8, 8)	2
	11	{A, B1, B2, C1, D1}	(7, 10, 9)	2
3	12	{A, D1, D2}	(6, 4, 7)	3
	13	{A, B1, D1, D2}	(6, 6, 8)	3
	14	{A, B1, B2, D1, D2}	(6, 8, 9)	3
	15	{A, B1, C1, D1, D2}	(9, 9, 11)	3
	16	{A, B1, C1, C2, D1, D2}	(12, 12, 14)	3
	17	{A, B1, B2, C1, C2, D1, D2}	(9, 11, 12)	3
	18	{A, B1, B2, C1, C2, D1, D2}	(12, 14, 15)	3
4	19	{A, B1, B2, D1, D2, D3}	(8, 9, 12)	4
	20	{A, B1, B2, C1, D1, D2, D3}	(11, 12, 15)	4
	21	{A, B1, B2, C1, C2, D1, D2, D3}	(14, 15, 18)	4
5	22	{A, B1, B2, C1, D1, D2, D3, E1}	(12, 13, 15)	6
	23	{A, B1, B2, C1, C2, D1, D2, D3, E1}	(15, 16, 18)	6
6	24	{A, B1, B2, C1, C2, D1, D2, D3, E1, E2}	(16, 17, 18)	7

$N_2(p) > 2$, добавляются к M_2, если они прошли исключающие тесты. Процедура повторяется последовательно для всех этапов, пока не будет достигнута завершающая вершина w_P и построен кратчайший путь в нее.

Применяя метод динамического программирования к иллюстративной задаче и используя исключающие тесты табл. 4.7, можно получить сокращенное множество пра-

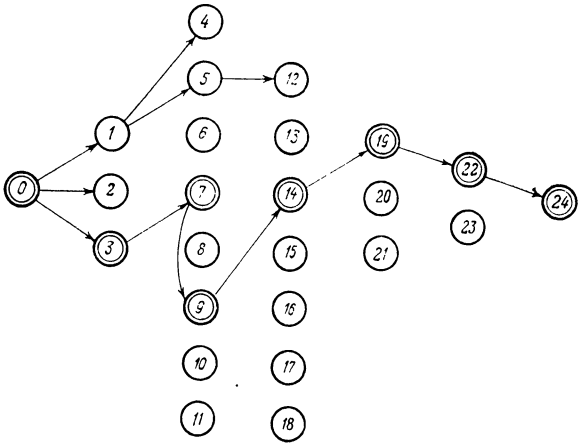

Рис. 4.4.

вильных подмножеств (табл. 4.8). На рис. 4.4 изображена сокращенная H-сеть, а в табл. 4.3 — искомое (кратчайшее) расписание.

4.2. О сложности алгоритмов динамического программирования

4.2.1. Постановка задачи. Подобно алгоритму из раздела 4.1, ряд алгоритмов динамического программирования на частично-упорядоченных множествах строится таким образом, что состояния процесса динамического программирования образованы правильными в определенном смысле подмножествами заданного частично-упорядоченного множества. К задачам, которые могут быть решены посредством указанных алгоритмов, относятся многие задачи разбиения графов, в частности известная задача балансирования сборочных линий, а также многочисленные сетевые задачи оптимального упорядочения. Как видно на примере иллюстративной задачи из раздела 4.1, сложность решения задач посредством алгоритмов этого типа вызвана лавинообразным ростом числа правильных подмножеств. В свою очередь, общее число правильных

подмножеств зависит от структуры исходного частично-упорядоченного множества. Представляет интерес получение оценок числа правильных подмножеств для некоторых классов задач. Ниже будут получены две оценки такого рода. Первая устанавливает максимальное число правильных подграфов у ациклического ориентированного графа с p вершинами и q дугами и характеризует сложность алгоритмов в худшем случае. Вторая оценка устанавливает среднее число правильных подграфов n-вершинного ациклического случайного графа и характеризует сложность алгоритмов в среднем.

4.2.2. Максимальное число правильных подмножеств. Предполагается, что конечное частично-упорядоченное множество представлено в виде ориентированного графа без петель и контуров (V, E). Как и ранее, будем говорить, что v_i предшествует v_j, если $(v_i, v_j) \in E$, и что подмножество вершин графа — правильное, если вместе с каждой вершиной оно содержит все предшествующие ей вершины. Условимся рассматривать графы с числом вершин p и числом дуг q, $|V| = p$, $|E| = q$, и обозначать их $G(p, q)$. Пусть $[G]$ обозначает количество правильных подмножеств графа G; условимся, что $[\varnothing] = 1$.

Как отмечалось, для прогнозирования затрат памяти ЭВМ и времени счета алгоритмов динамического программирования в ряде экстремальных задач на ациклических графах необходимо оценивать число правильных подмножеств вершин таких графов. Далее будет построен критический граф $K(p, q)$, у которого число правильных подмножеств вершин максимально среди всех графов $G(p, q)$ с p вершинами и q дугами.

Теорема 4.1. *Если граф G имеет m компонент G_i, $i = 1, \ldots, m$, то*

$$[G] = \prod_{i=1}^{m} [G_i].$$

Утверждение теоремы непосредственно следует из того, что имеет место взаимно однозначное соответствие между подмножествами вершин W, правильными в G, и m-кортежами $(W_1, \ldots, W_m)$, где W_i — подмножество вершин, правильное в графе G_i, $i = 1, \ldots, m$. Это соответствие таково, что $W = \bigcup_{i=1}^{m} W_i$.

Граф $T = (V, E)$ назовем выходящим деревом с корнем v_0, если в его вершину v_0 не входит ни одной дуги, а во все остальные вершины входит по одной дуге. Вер-

шина графа T называется висячей, если из нее не выходит ни одной дуги.

Теорема 4.2. *Пусть T — выходящее дерево с корнем v_0. Если приписать всем висячим вершинам v вес $d(v) = 2$, а веса остальных вершин подсчитать по формуле*

$$d(v) = \prod_{v_i \in E(v)} d(v_i) + 1,$$

где $E(v) = \{v_i | (v, v_i) \in E\}$, то $[T] = d(v_0)$.

Доказательство. Теорема может быть доказана по индукции. Заметим, что вес вершины $v_i \in V$ дает чис-

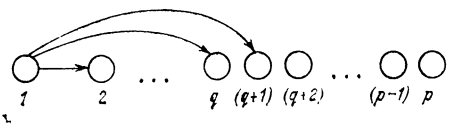

Рис. 4.5.

ло правильных подмножеств в множестве вершин, состоящих из v_i и всех ее потомков. Индукция состоит в переходе от висячих вершин к их непосредственным предшественникам и т. д., пока не достигнем корня v_0. При этом используется теорема 4.1.

Перейдем к построению критического графа $K(p, q)$ с максимальным числом правильных подмножеств вершин. Условимся, что вершины его занумерованы числами i натурального ряда от 1 до p, а дуги образованы парами вершин (i, j), $i < j$, причем $1 \leqslant q \leqslant p(p-1)/2$.

Определим семейство графов

$$\mathcal{K}(p) = \{K_{n(q)}(p, q),\ q \geqslant 1,\ n(q) \geqslant 1\}$$

следующего вида. Граф $K_1(p, 1)$ имеет всего одну дугу $(1, 2)$. Для каждого $q > 1$ граф $K_{n(q)}(p, q)$ получается из $K_{n(q-1)}(p, q-1)$ добавлением к нему определенным образом одной дуги. Индекс $n(q) \geqslant 1$ однозначно определяется числом дуг q и характеризует структуру текущего графа $K_{n(q)}(p, q)$. Графы $K_{n(q)}(p, q)$ строятся рекуррентно по q, начиная с $K_1(p, 1)$.

Итак, пусть $K_1(p, 1) \in \mathcal{K}(p)$ — граф, определенный выше. Рассмотрим сначала случай $2 \leqslant q \leqslant p-1$. Для этих значений q положим $n(q) = 1$. Граф $K_1(p, q)$ получается при этом из $K_1(p, q-1)$ добавлением ребра $(1, q+1)$. На рис. 4.5 изображен граф $K_1(p, q) \in \mathcal{K}(p)$, построенный указанным способом. Из теорем 4.1 и 4.2 получаем, что

$$[K_1(p, q)] = (2^q + 1)2^{p-q-1} = 2^{p-1} + 2^{p-q-1}.$$

Используя указанные теоремы, а также тот факт, что добавление к произвольному ациклическому графу G дуги с сохранением свойств ацикличности не увеличивает величину $[G]$, легко показать, что для любого графа $G(p, q)$, удовлетворяющего условию $1 \leqslant q \leqslant p-1$, выполняется соотношение

$$[G(p, q)] \leqslant [K_1(p, q)].$$

Рассмотрим теперь случай $p \leqslant q \leqslant 2p-3$. При этом $n(q)=2$. Критический граф $K_2(p, p) \in \mathcal{K}(p)$ получается из $K_1(p, p-1)$ добавлением дуги (2, 3) (рис. 4.6). Остальные графы $K_2(p, q) \in \mathcal{K}(p)$ получаются из $K_2(p, p)$ добавлением дуг (2, 4), (2, 5), ..., $(2, q-p+3)$. Из теорем 4.1 и 4.2 в этом случае нетрудно получить, что

$$[K_2(p, q)] = 2^{p-2} + 2^{2p-q-3} + 1.$$

Следующий граф из $\mathcal{K}(p) - K_3(p, 2p-2)$ — получается из $K_2(p, 2p-3)$ добавлением дуги (3, 4) и т. д.

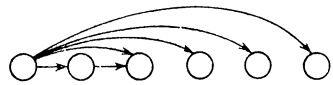

Рис. 4.6.

В общем случае, т. е. при произвольном q, $1 \leqslant q \leqslant p(p-1)/2$, индекс $n(q)$ определяется как целочисленное решение неравенства

$$(n(q)-1)\,p - \frac{n(q)\,(n(q)-1)}{2} < q \leqslant n(q)\,p - \frac{n(q)\,(n(q)+1)}{2}.$$

Соответствующий граф $K_{n(q)}(p, q)$ имеет следующий вид: в нем только первые $n(q)$ вершин имеют исходящие дуги; из них $p-1$ исходят из вершин 1, соединяя ее со всеми остальными вершинами графа; $p-2$ дуг исходят из вершины 2, соединяя ее с вершинами 3, 4, ..., p; $p-n(q)+1$ дуг исходят из вершины $n(q)-1$, соединяя ее с вершинами $n(q)$, $n(q)+1$, ..., p. Наконец, из вершины $n(q)$ исходит

$$l = n(q)\,p - \frac{n(q)\,(n(q)+1)}{2} - q$$

дуг, соединяющих ее с вершинами $n(q)+1$, ..., $n(q)+l$. Применение теорем 4.1 и 4.2 дает в этом случае для числа правильных подмножеств графа $K_{n(q)}(p, q)$ формулу

$$[K_{n(q)}(p, q)] = 2^{p-n(q)} + 2^{n(q)p-n(q)(n(q)+1)\,2-q} + n - 1.$$

Замечание. Нетрудно видеть, что выбрасыванием $q-p+1$ дуг граф $K_{n(q)}(p, q)$ можно преобразовать в дерево $T(p, p-1)$ такое, что

$$[K_{n(q)}(p, q)] = [T(p, p-1)].$$

4.2.3. Среднее число правильных подмножеств. Как и в предыдущем разделе, будем полагать, что исходное частично-упорядоченное множество представлено в виде ациклического графа (без петель и контуров). По-прежнему речь идет о графах с p вершинами и q дугами. Чтобы получить оценку среднего числа правильных подмножеств вершин, введем некоторый случайный механизм порождения графов данного класса. А именно, рассмотрим случайные графы (V, R) следующего вида. Пусть $V = \{1, \ldots, n\}$ — множество вершин. На множестве всевозможных пар (i, j) заданы вероятности появления дуг (i, j) (матрицей $R = \|r_{ij}\|_{i,j=1}^{n}$). Соответствующие случайные величины, относящиеся к различным дугам, условимся считать независимыми. Поскольку нас интересуют только ациклические графы, полагаем $r_{ij} = 0$ для всех $i \geqslant j$. Очевидно, что сумма всех элементов матрицы R равна математическому ожиданию числа дуг в случайном графе (V, R).

По определению, подмножество вершин $V_m = \{j_1, \ldots, j_m\} \subseteq V$ случайного графа (V, R) — правильное, если $j_l < j_{l+1}$ для всех j_l, $l = 1, \ldots, m-1$. Определим вероятность наступления некоторого события, порождающего правильное подмножество. Пусть $P(V_m)$ обозначает вероятность того, что V_m — правильное подмножество; тогда с учетом независимости появления различных дуг имеет место равенство

$$P(V_m) = \prod_{i=1}^{j_1-1} (1 - r_{ij_1}) \prod_{\substack{i=1 \\ i \neq j_1}}^{j_2-1} (1 - r_{ij_2}) \ldots$$

$$\ldots \prod_{\substack{i=1 \\ i \neq j_1, \ldots, j_{m-1}}}^{j_m-1} (1 - r_{ij_m}). \qquad (4.8)$$

Здесь первое произведение означает вероятность отсутствия дуг, входящих в вершину j_1, второе — вероятность отсутствия дуг, входящих в вершину j_2, за исключением, быть может, дуги из вершины j_1 и т. д. Обозначим

$q_{ij} = 1 - r_{ij}$; тогда (4.8) запишется в виде

$$P(V_m) = \prod_{\substack{i \in V \setminus V_m \\ j \in V_m}} q_{ij}.$$

Очевидно, что $P(V) = 1$. Введем функцию-индикатор правильных подмножеств:

$$\omega(i_1, \ldots, i_m) = \begin{cases} 1, \text{ если } \{i_1, \ldots, i_m\} \text{ — правильное} \\ \text{подмножество,} \\ 0 \text{ в противном случае.} \end{cases}$$

Отметим, что в случайном графе (V, R) функция $\omega(i_1, \ldots, i_m)$ является случайной величиной, принимающей значение 1 с вероятностью $P(V_m)$ и 0 с вероятностью $(1 - P(V_m))$. Определим теперь случайную величину, равную общему числу правильных подмножеств. Так как по предположению вершины графа (V, R) пронумерованы числами с 1 по n, а дуги появляются и связывают только пары вершин (i_k, i_l), $i_k < i_l$, то при подсчете общего числа различных правильных подмножеств $\{i_1, \ldots, i_m\}$ можно полагать $i_1 < i_2 < \ldots < i_m$. Тогда общее число правильных подмножеств определяется функцией

$$N(V,R) = \sum_{\substack{i_1 < i_2 \ldots < i_m \\ 1 \leqslant m \leqslant n}} \omega(i_1, \ldots, i_m).$$

Соответственно, математическое ожидание общего числа правильных подмножеств равно

$$M(N) = \sum_{i_1 < \ldots < i_m} P(V_m).$$

Рассмотрим случай, когда вероятности появления дуг одинаковы: $r_{ij} = r$ для всех $i < j$, а в остальных случаях $r_{ij} = 0$. Соответственно обозначим $q = 1 - r$. Подсчитаем математическое ожидание числа различных правильных подмножеств с фиксированным числом элементов k:

$$M(N_k) = \sum_{1 \leqslant i_1 < \ldots < i_k \leqslant n} P(i_1, \ldots, i_k) = \sum_{1 \leqslant i_1 < \ldots < i_k \leqslant n} q^{i_1 - 1} \ldots$$

$$\ldots q^{i_k - k} = \sum_{0 \leqslant j_1 \leqslant \ldots \leqslant j_k \leqslant n - k} q^{j_1 + \ldots + j_k}. \qquad (4.9)$$

Преобразуем выражение (4.9), воспользовавшись матема-

тической индукцией по k. Имеем

$$M(N_1) = \sum_{0 \leqslant j_1 \leqslant n-1} q^{j_1} = \frac{1-q^n}{1-q},$$

$$M(N_2) = \sum_{j_1=0}^{n-2} q^{j_1} \sum_{j_2=j_1}^{n-2} q^{j_2} = \sum_{j_1=0}^{n-2} q^{j_1} \frac{q^{j_1} - q^{n-1}}{1-q} =$$

$$= \frac{1}{1-q} \frac{1-q^{2(n-1)}}{1-q^2} - q^{n-1} \frac{1-q^{n-1}}{1-q} = \frac{(1-q^n)(1-q^{n-1})}{(1-q)(1-q^2)}$$

и, вообще,

$$M(N_k) = \frac{(1-q^n)(1-q^{n-1}) \dots (1-q^{n-k+1})}{(1-q)(1-q^2) \dots (1-q^k)}. \quad (4.10)$$

Замечание. Выражение $\sum_{1 \leqslant i_1 < i_2 \dots < i_k \leqslant n} t^{i_1 + \dots + i_k}$ является перечисляющей производящей функцией (энумератором) количества частей разбиений, в которых все части различны и ни одна из них не превосходит n. В [9], стр. 182, приводится ее выражение

$$P_n(t, k) = t^{k(k+1)/2} \frac{(1-t^n)(1-t^{n-1}) \dots (1-t^{n-k+1})}{(1-t)(1-t^2) \dots (1-t^k)}.$$

Из выражения (4.10) следует, что

$$M(N_k) = M(N_{n-k}).$$

В предельных случаях получаем: при $q = 0$ $M(N_k) = 1$, при $q = 1$ $M(N_k) = \binom{n}{k}$, что согласуется с результатами раздела 4.2.2.

При достаточно малых значениях q для вычисления $M(N_k)$ можно использовать первые члены разложения выражения (4.10) по степеням q:

$$M(N_k) = 1 + q + 2q^2 + 3q^3 + 5q^4 + 7q^5 + 11q^6 + \dots$$

Рассмотрим также поведение $M(N_k)$ для значений q, близких к 1. Воспользуемся тем, что выражение

$$\sum_{1 \leqslant i_1 < \dots < i_k \leqslant n} q^{i_1 + \dots + i_k}$$

не зависит от перестановки индексов $i_1, \dots, i_k$, поэтому

$$k! \sum_{1 \leqslant i_1 < \dots < i_k \leqslant n} t^{i_1 + \dots + i_k} = \sum_{\substack{1 \leqslant i_l \leqslant n \\ l = 1, \dots, k}}' t^{i_1} t^{i_2} \dots t^{i_k},$$

где $\sum'$ означает сумму по всем возможным наборам различных индексов. Используя принцип включения и исключения, запишем

$$k! \sum_{1 \leqslant i_1 < \ldots < i_k \leqslant n} t^{i_1 + \ldots + i_k} = \left(\frac{1 - t^{n+1}}{1 - t}\right)^k \binom{k}{2} \frac{1 - t^{2n+2}}{1 - t^2} \times$$

$$\times \left(\frac{1 - t^{n+1}}{1 - t}\right)^{k-2} + 2\binom{k}{3} \frac{1 - t^{3n+3}}{1 - t^3} \left(\frac{1 - t^{n+1}}{1 - t}\right)^{k-3} + \ldots$$

Подставив этот результат в (4.9), получим

$$M(N_k) = t^{-k(k+1)/2} \cdot \frac{1}{k!} \left(\frac{1 - t^{n+1}}{1 - t}\right)^k \left[1 - \binom{k}{2} \frac{1 + t^{n+1}}{1 + t} \times\right.$$

$$\left. \times \frac{1 - t}{1 - t^{n+1}} + 2\binom{k}{3} \frac{1 + t^{n+1} + t^{2n+2}}{1 + t + t^2} \left(\frac{1 - t}{1 - t^{n+1}}\right)^2 + \ldots\right]. \tag{4.11}$$

Выражение (4.11) можно упростить, если $t^{n+1} \approx 1$; тогда

$$M(N_k) = \frac{t^{-k(k+1)/2}}{k!} n^k \left[1 - \binom{k}{2} \frac{1}{n} + 2\binom{k}{3} \frac{1}{n^2} - \ldots\right].$$

Для остальных случаев $M(N_k)$ следует подсчитывать по формуле (4.10).

Библиографический комментарий

В данной главе описан алгоритм динамического программирования, предложенный в [1]. Алгоритмы этого типа строятся на основе некоторого генератора правильных подмножеств (п. п.)— состояний процесса динамического программирования. Генератору п. п. [1] предшествовали генераторы п. п. из [4, 5]. Другие генераторы п. п. предложены в [6 — 8]. Сложность алгоритмов этого типа определяется общим числом п. п. Оценки числа п. п., представленные в данной главе, получены в [2, 3].

ПОСЛЕДОВАТЕЛЬНЫЙ АНАЛИЗ ВАРИАНТОВ

5.1. Задача с независимыми операциями

Рассмотрим задачу следующего вида. В заданном интервале времени $\Delta = [0, T]$ следует выполнить N работ j, $j = 1, \ldots, N$. Каждая работа описывается тройкой (τ_j, r_j, f_j), где $\tau_j > 0$ — длительность работы j; $r_j \geqslant 0$ — уровень использования ресурсов в процессе выполнения работы j; $f_j = f_j(t)$, $t \in [0, T]$,— функция времени, определяющая условные потери, связанные с завершением работы j в момент времени t. Предполагается, что каждую работу не следует прерывать, т. е. если t_j — время завершения работы j, то $(t_j - \tau_j)$ — время ее начала. Переменная времени t принимает значения $t = 0, 1, 2, \ldots, T$; все τ_j и T — целые числа.

В интервале планирования Δ выделены временны́е вехи $0 = T_0 < T_1 < \ldots < T_m = T$, определяющие соответственно интервалы

$$\Delta_1 = [T_0, T_1], \ldots, \Delta_i = [T_{i-1}, T_i], \ldots, \Delta_m = [T_{m-1}, T_m].$$

Для непрерывных операций расписание S определяется набором целых чисел t_j:

$$S = \{t_j(S)\}, \quad j = 1, 2, \ldots, N,$$

причем $t_j(S) \in [0, T]$. Для заданного расписания S определим условную нагрузку интервала Δ_i, $i = 1, \ldots, m$, как функцию

$$g_i(S) = \sum_{j=1}^{N} g_{ij}(t_j(S)),$$

где

$$g_{ij}(t_j(S)) = r_j |\Delta_i \cap [t_j(S) - \tau_j, t_j(S)]|.$$

Смысл функций $g_{ij}(t_j(S))$ — объем использования ресурсов на интервале Δ_i в процессе выполнения работы j.

Допустимые общие объемы использования ресурсов в каждом интервале Δ_i определяются заданными уровнями g_i^0, $i = 1, 2, \ldots, m$. Задача составления оптимального расписания выполнения работ заключается в следующем: требуется найти набор чисел t_j, на котором функция $F(S) = \sum_{j=1}^{N} f_j(t_j(S))$ достигает минимума при условии, что нагрузки в каждом интервале Δ_i не превосходят соответствующих уровней g_i^0.

В формальном виде задача представляется как нелинейная задача о рюкзаке:

$$G_0(x) = \sum_{j=1}^{N} g_{0j}(x_j) \to \min \tag{5.1}$$

при условиях

$$G_i(x) = \sum_{j=1}^{N} g_{ij}(x_j) \leqslant g_i^0, \quad i = 1, \ldots, m, \tag{5.2}$$

$$0 \leqslant x_j \leqslant T, \quad x_j \text{ — целое}, \quad j = 1, \ldots, N. \tag{5.3}$$

Здесь обозначено $x_j \equiv t_j$, $g_{0j}(x_j) \equiv f_j(t_j)$, $j = 1, \ldots, N$.

Заметим, что сформулированная задача допускает естественное обобщение на случай нескольких ресурсов. Кроме того, обычные в теории расписаний ограничения на сроки начала выполнения работ $t_j - \tau_j \geqslant \underline{T}_j$ и на сроки завершения работ $t_j \leqslant \overline{T}_j$ также легко могут быть учтены посредством соответствующего изменения целевых функций $f_j(t_j)$.

5.2. Метод последовательного анализа вариантов (общее описание)

Многие алгоритмы последовательного анализа вариантов основаны на принципе оптимальности, который представляет собой естественное обобщение принципа оптимальности динамического программирования. Вначале опишем их в общих терминах, затем приведем описание алгоритма применительно к задаче (5.1)—(5.3).

Пусть задано некоторое базовое множество X. Обозначим множество конечных последовательностей вида

$$p = (x_1, \ldots, x_i, \ldots, x_{k_p}), \quad x_i \in X, \quad 1 \leqslant i \leqslant k_p,$$

через $P(X)$. В X выделено некоторое подмножество допустимых последовательностей $W(X) \subseteq P(X)$; в свою оче-

редь, в $W(X)$ выделено подмножество полных допустимых последовательностей $\overline{W}(X) \subseteq W(X)$. Пусть задана последовательность p. Назовем l-м начальным отрезком этой последовательности последовательность вида

$$p_l = (x_1, \ldots, x_l), \quad 1 \leqslant l \leqslant k_p,$$

и q-м конечным отрезком — последовательность вида

$$p^q = (x_q, x_{q+1}, \ldots, x_{k_p}), \quad 1 \leqslant q \leqslant k_p.$$

Если $q = l + 1$, то соответствующие части последовательности p называются сопряженными.

Рассмотрим две произвольные допустимые последовательности p_1 и p_2 и выделим в p_1 l_1-начальный отрезок p_{1l_1} и сопряженный ему конечный отрезок $p_1^{l_1+1}$, а в p_2 выделим l_2-начальный отрезок p_{2l_2} и сопряженный ему конечный отрезок $p_2^{l_2+1}$. Функционал Ф, определенный на множестве $W(X)$, называется монотонно-рекурсивным, если он обладает следующим свойством: из того, что

$$p_{1l_1} \in W(X), \quad p_{2l_2} \in W(X), \quad p_1^{l_1+1} \equiv p_2^{l_2+1},$$
$$\Phi(p_{1l_1}) < \Phi(p_{2l_2}),$$

следует $\Phi(p_1) < \Phi(p_2)$.

Пусть $\Phi^* = \sup_{p \in \overline{W}} \Phi(p)$. Назовем последовательность p^* максимальной, если

$$\Phi(p^*) = \Phi^*, \quad p^* \in \overline{W}(X).$$

Для заданной допустимой последовательности p p-родовым множеством назовем подмножество $R(p) \subseteq \overline{W}(X)$, состоящее из тех полных допустимых последовательностей, у которых p является начальным отрезком. Множеством продолжений $P(p)$ назовем совокупность всех конечных отрезков элементов p-родового множества, сопряженных с p. Обобщенный принцип оптимальности формулируется так.

Теорема 5.1. *Пусть заданы монотонно-рекурсивный функционал* Ф *и две допустимые последовательности* p_1 *и* p_2, *причем*

$$\Phi(p_1) < \Phi(p_2), \tag{5.4}$$

$$P(p_1) \subseteq P(p_2). \tag{5.5}$$

Тогда элементы множества $R(p_1)$ *не могут быть максимальными.*

Доказательство. Допустим, что существует последовательность $p_1^* \in R(p_1)$ такая, что

$$\Phi(p_1^*) = \Phi^*. \tag{5.6}$$

По определению, p_1^* можно представить как состоящую из начального отрезка p_1 и сопряженного с ним конечного отрезка $\bar{p}_1 \in P(p_1)$. Тогда из (5.5) $\bar{p}_1 \in P(p_2)$ и, следовательно, в $R(p_2)$ найдется p_2^* с начальным отрезком p_2 и сопряженным с ним конечным отрезком $\bar{p}_1$. Из определения Φ и условия (5.4) тогда будет следовать $\Phi(p_2^*) > \Phi(p_1^*) = \Phi^*$, что противоречит (5.6). Поэтому не существует последовательности $p_1^* \in R(p_1)$, которая является максимальной, что и требовалось доказать.

Таким образом, метод определения максимального элемента для монотонно-рекурсивных функционалов сводится к следующему:

1. Рассматривается некоторое ограниченное число допустимых последовательностей таких, что объединение их родовых множеств и тех из рассматриваемых последовательностей, которые являются полными допустимыми, в совокупности дает все множество полных допустимых последовательностей.

2. На основе обобщенного принципа оптимальности исключается часть родовых множеств; из рассматриваемых полных допустимых последовательностей оставляются только те, которые дают наибольшее значение функционалу; исключаются из рассмотрения последовательности, для которых родовое множество пусто.

3. Выбирается некоторая допустимая последовательность из числа рассмотренных, для которой родовое множество непусто и не исключалось. Рассматривается некоторое ограниченное число допустимых последовательностей, являющихся продолжением выбранной последовательности и таких, что объединение их родовых множеств и тех из них, которые являются полными, в совокупности дает родовое множество выбранной последовательности.

4. Для множества, состоящего из вновь образованных в п. 3 допустимых последовательностей и неисключенных и непродолженных ранее допустимых последовательностей, производятся операции, указанные в п. 2.

Далее, пп. 2, 3, 4 циклически повторяются. Если на каком-то этапе процесса решения не останется ни одной

допустимой последовательности с непустым или неисключенным родовым множеством, то процесс решения завершен (останавливается) и в качестве решения берется одна из рассмотренных полных допустимых последовательностей с небольшим значением функционала. На каждом этапе процесса решения требуется помнить множество полных последовательностей, остающихся для дальнейшего продолжения.

5.3. Алгоритм последовательного анализа вариантов в задаче с независимыми операциями

Рассмотрим алгоритм последовательного анализа вариантов в задаче (5.1)—(5.3). Решением задачи (5.1)—(5.3) называется последовательность $x = \{x_j,\ j = 1,\ \ldots,\ N\}$ такая, что $x_j = 1,\ \ldots,\ T$. Точка x называется допустимым решением, если неравенства $G_i(x) \leqslant g_i^0$ совместны для всех $i = 1,\ \ldots,\ m$. Последовательность

$$\hat{x}_n = \{x_1,\ \ldots,\ x_j,\ \ldots,\ x_n\}, \quad 1 \leqslant n \leqslant N,$$

называется начальным отрезком (н. о.) длины n. Соответственно, $\hat{x}_n$ — допустимый н. о., если неравенства

$$G_i(\hat{x}_n) = \sum_{j=1}^{n} g_{ij}(x_j) \leqslant g_i^0$$

совместны для всех $i = 1,\ \ldots,\ m$.

Пусть задан н. о. $\hat{x}_n$. Значения функций $G_i(\hat{x}_n)$, $i = 0,\ 1,\ \ldots,\ m$, условимся называть параметрами н. о. $\hat{x}_n$, а $(m+1)$-мерный вектор с компонентами $G_i(\hat{x}_n)$ — вектором параметров н. о. $\hat{x}_n$. В частности, при $n = N$ функции $G_i(\hat{x}_N)$ являются параметрами решений задачи (5.1)—(5.3).

Пусть даны два н. о. длины n: $\hat{x}_n^1$ и $\hat{x}_n^2$. Скажем, что $\hat{x}_n^1$ мажорирует $\hat{x}_n^2$ (соответственно, $\hat{x}_n^2$ мажорируется $\hat{x}_n^1$), и запишем $\hat{x}_n^1 \succ \hat{x}_n^2$, если выполнены неравенства

$$G_i(\hat{x}_n^1) - G_i(\hat{x}_n^2) \leqslant 0, \quad i = 0,\ \ldots,\ m, \tag{5.7}$$

причем хотя бы одно из них — строгое.

Условимся, что в процессе вычислений допустимые н. о. упорядочиваются так, что

$$G_0(\hat{x}_n^1) \leqslant G_0(\hat{x}_n^2) \leqslant \ldots, \quad 1 \leqslant n \leqslant N;$$

тогда первый по порядку допустимый н. о. длины N является искомым решением задачи (5.1)—(5.3).

Обозначим через W_n множество всех допустимых н. о. длины n. Пусть $W_{n_1}^0$, W_n^1 — подмножества W_n такие, что

1) для каждого н. о. $\hat{y}_n \in W_n^0$ найдется такой н. о. $\hat{x}_n \in W_{n_1}^1$ что $\hat{x}_n \succ \hat{y}_n$;

2) в W_n^1 не найдется ни одной пары н. о. $(\hat{x}_n, \hat{z}_n)$ такой, что $\hat{x}_n \succ \hat{z}_n$;

3) $W_n^0 \cup W_n^1 = W_n$, $W_n^0 \cap W_n^1 = \varnothing$.

Следующая процедура выделяет в W^n все н. о., принадлежащие W_n^0:

1) пометить в W_n все н. о. $\hat{x}_n^s$, для которых $\hat{x}_n^1 \succ \hat{x}_n^s$;

2) начиная с первого по порядку непомеченного н. о. $\hat{x}_n^s$, пометить все непомеченные ранее н. о. $\hat{x}_n^{s+l}$, для которых $\hat{x}_n^s \succ \hat{x}_n^{s+l}$;

3) повторять п. 2) до тех пор, пока для некоторого s все $\hat{x}_n^{s+l}$ окажутся помеченными или же список н. о. окажется исчерпанным.

Помеченные отрезки принадлежат множеству $W_{n_1}^0$, непомеченные — множеству W_n^1.

Элементы $\hat{x}_1$ образуют н. о. длины 1. Н. о. $\hat{x}_{n+1}$ длины $(n+1)$ образуются как упорядоченные пары

$$\hat{x}_{n+1} = (\hat{x}_n, x_j),$$

где $x_j = 1, \ldots, T$. Начальные отрезки длины N являются решениями задачи. Н. о. $\hat{x}_n^1$ из W_N^1 является, очевидно, искомым решением.

Алгоритм решения задачи (5.1)—(5.3) может быть, таким образом, охарактеризован как N-шаговый процесс, на каждом шаге которого производится генерация допустимых н. о., последовательная проверка условия мажорирования (5.7) для пар н. о. в соответствии с данной выше процедурой и запоминание мажорирующих н. о. (множество W_n^1).

5.4. Оценки трудоемкости в среднем алгоритма последовательного анализа вариантов

5.4.1. Оптимальность по Парето и множество немажорируемых вариантов. Понятие оптимальности по Парето часто встречается в математической экономике и теории игр. Нам будет достаточно пользоваться следующим его определением. Пусть имеется множество W, элементы

которого оцениваются по нескольким показателям, так что $p_i(w)$, $i=0, \ldots, m$,— значение i-го показателя в точке $w \in W$. Скажем, что $W^1 \subseteq W$ есть множество Парето, если для любого w^2 из $W \setminus W^1$ в W^1 найдется такое w^1, что

а) $p_i(w^1) \geqslant p_i(w^2)$ для всех i;

б) $p_i(w^1) > p_i(w^2)$ хотя бы для одного i;

в) для любого $w \in W^1$ не найдется $w^3 \in W^1$ такого, что для w^1, w^3 выполняются а) и б).

Вернемся теперь к отношению мажорирования (5.7). Покажем, что множество мажорирующих н. о. образует множество Парето. Для этого достаточно положить

$$W = W_n,$$

$$p_i(\hat{x}_n) = -G_i(\hat{x}_n), \quad i=0, \ldots, m.$$

Тогда множество W_n^1 мажорирующих н. о. длины n удовлетворяет свойствам а), б), в) и является множеством Парето. Кроме того, описанная выше процедура пометок является алгоритмом для построения множества Парето на n-м шаге процесса. Потребности в оперативной памяти ЭВМ, необходимые для построения множества Парето W_n^1, главным образом определяются величиной $|W_n^1|$. Важно поэтому оценить ее априори до самих вычислений. Эти оценки могут быть использованы как для планирования объема памяти вычислительных алгоритмов последовательной оптимизации, так и для оценок сложности этих алгоритмов.

5.4.2. Статистический подход к оценке трудоемкости в среднем. Мы будем предполагать, что генерация вариантов значений x_j производится случайным образом независимо для каждого j. Исходя из этого, совокупность параметров $g_{ij}(x_j)$, $i=0, \ldots, m$, определенного варианта x_j можно рассматривать как реализацию некоторого случайного вектора $\xi_j = (\xi_j^0, \xi_j^1, \ldots, \xi_j^m)$, принимающего значение $(g_{0j}(x_j), g_{1j}(x_j), \ldots, g_{mj}(x_j))$ с вероятностью $1/T$, где T — число различных значений x_j, а совокупность параметров каждого варианта из множества W начальных отрезков — как реализацию суммы независимых случайных векторов:

$$\eta_n = \sum_{j=1}^{n} \xi_j;$$

при этом вероятность каждого варианта начального отрезка будет равна $(T^n)^{-1}$. Мы легко можем определить

математическое ожидание и корреляционные моменты случайного вектора η_n. В самом деле,

$$M\eta_n = \sum_{j=1}^{n} M\xi_j, \quad D^{i_1 i_2}\eta_n = \sum_{j=1}^{n} D^{i_1 i_2}\xi_j;$$

$$M\xi_j^i = M_j^i = \frac{1}{T}\sum_{x_j=1}^{T} g_{ij}(x_j), \qquad i = 0, \ldots, m;$$

$$D^{i_1 i_2}\xi_j = M\left(\xi_j^{i_1} - M_j^{i_1}\right)\left(\xi_j^{i_2} - M_j^{i_2}\right) =$$

$$= \frac{1}{T}\sum_{x_j=1}^{T}\left(g_{i_1 j}(x_j) - M_j^{i_1}\right)\left(g_{i_2 j}(x_j) - M_j^{i_2}\right).$$

Нас будет в дальнейшем интересовать оценка числа точек в множестве W_n^1 (число вариантов н. о. длины n в множестве Парето). Это число может меняться от 1 до T^n. Приближенно такую оценку можно получить, если рассматривать совокупность вариантов н. о. длины n как выборку независимых значений случайного вектора.

Таким образом, мы приходим к задаче оценки математического ожидания числа точек множества Парето в конечном множестве, представляющем выборку, составленную из реализаций одинаково распределенных случайных векторов. Далее, в разделах 5.4.3 и 5.4.4, будут рассмотрены два случая, когда случайный вектор η_n имеет независимые одинаково распределенные компоненты и когда η_n имеет нормальное распределение.

5.4.3. Оценка математического ожидания числа крайних точек в случайной выборке независимых, одинаково распределенных случайных векторов с независимыми компонентами. Рассмотрим выборку случайных векторов $\{X_1, \ldots, X_n\}$, где $X_j = (X_j^1, \ldots, X_j^m)$, $j = 1, \ldots, n$, независимы, одинаково распределены и имеют взаимно независимые компоненты. Компоненты каждого вектора X_j имеют непрерывную функцию распределения $F_i(X_j^i) = F_i(X^i)$, $i = 1, \ldots, m$.

Пусть W_n — множество Парето для выборки $\{X_1, \ldots, X_n\}$. Введем на множестве векторов функцию-индикатор

$$I_j = \begin{cases} 1, & \text{если } X_j \in W_n, \\ 0, & \text{если } X_j \overline{\in} W_n, \end{cases} \quad j = 1, \ldots, n.$$

Случайная величина $\pi_n = \sum_{j=1}^{n} I_j$ равна числу векторов, попавших в множество Парето. Выражение для математического ожидания π_n запишется с учетом одинакового их распределения в виде

$$M(\pi_n) = M \sum_{j=1}^{n} I_j = nM(I_1).$$

Так как I_1 принимает значения 0 или 1, то $M(I_1) = P\{X_1 \in W_n\}$ — вероятность события, что не существует вектора

$$X_j > X_1, \qquad j = 2, \ldots, n.$$

Зафиксируем X^*; тогда

$$P\{X_1^* \in W_n\} = \left(1 - \int_{X_1^{*1}}^{\infty} \ldots \int_{X_1^{*m}}^{\infty} dF_1(X^1) \ldots dF_m(X^m)\right)^{n-1} =$$

$$= \left(1 - F_1\left(X_1^{*1}\right) \ldots F_m\left(X_1^{*m}\right)\right)^{n-1}.$$

Для произвольного X_1

$$P\{X_1 \in W_n\} = \int_{-\infty}^{\infty} \ldots \int_{-\infty}^{\infty} \left(1 - F_1\left(X_1^1\right) \ldots F_m\left(X_1^m\right)\right)^{n-1} \times$$

$$\times\, dF_1(X^1) \ldots dF_m(X^m).$$

После замены $F_i\left(X_1^i\right) = x_i$ получаем

$$M(\pi_n) = n \int_0^1 \ldots \int_0^1 (1 - x_1 \ldots x_m)^{n-1}\, dx_1 \ldots dx_m. \qquad (5.8)$$

Для вычисления (5.8) воспользуемся интегрированием по линиям уровня подынтегральной функции; при этом m-мерный интеграл сводится к одномерному:

$$M(\pi_n) = n \int_0^1 (1 - t)^{n-1} v_m'(t)\, dt;$$

здесь $v_m(t)$ — объем m-мерной области, определяемой соотношениями

$$x_1 \ldots x_m \leqslant t,$$

$$0 \leqslant x_i \leqslant 1, \qquad i = 1, \ldots, m.$$

Далее, для удобства записи ограничения $0 \leqslant x_i \leqslant 1$ опус-

каем, считая, что они всегда выполняются. Имеем

$$v_m(t) = \int\limits_{x_1 \dots x_m \leqslant t} \dots \int dx_1 \dots dx_m = \int\limits_{\substack{x_1 \dots x_{m-1} \leqslant t \\ 0 \leqslant x_m \leqslant 1}} \dots \int dx_1 \dots dx_m +$$

$$+ \int\limits_{\substack{x_1 \dots x_{m-1} \geqslant t \\ 0 \leqslant x_m \leqslant t/(x_1 \dots x_{m-1})}} \dots \int dx_1 \dots dx_m =$$

$$= v_{m-1}(t) + \int\limits_0^1 dx_1 \int\limits_{t/x_1}^1 dx_2 \dots$$

$$\dots \int\limits_{t/(x_1 \dots x_{m-2})}^1 dx_{m-1} \int\limits_0^{t/(x_1 \dots x_{m-1})} dx_m =$$

$$= v_{m-1}(t) + (-1)^{m-1} t \int\limits_t^1 d \ln\left(\frac{1}{x_1}\right) \int\limits_{t/x_1}^1 d \ln\left(\frac{1}{x_2}\right) \dots$$

$$\dots \int\limits_{t/(x_1 \dots x_{m-2})}^1 d \ln\left(\frac{1}{x_{m-1}}\right) = v_{m-1}(t) + (-1)^{m-1} \frac{t (\ln t)^{m-1}}{(m-1)!},$$

откуда

$$v_m'(t) = v_{m-1}(t) + \frac{(-\ln t)^{m-1}}{(m-1)!} - \frac{(-\ln t)^{m-2}}{(m-2)!}.$$

Теперь воспользуемся полученным рекуррентным соотношением, учитывая, что $v_1(t) = \int\limits_0^t dx = t$, $v_1'(t) = 1$. Получим

$$v_m'(t) = \frac{(-\ln t)^{m-1}}{(m-1)!};$$

$$M(\pi_n) = \frac{n}{(m-1)!} \int\limits_0^1 (1-t)^{n-1} (-\ln t)^{m-1} dt;$$

$$M(\pi_n) = \frac{n}{(m-1)!} \sum_{i=0}^{n-1} (-1)^i \binom{n-1}{i} \int\limits_0^1 t^i (-\ln t)^{m-1} dt =$$

$$= n \sum_{i=0}^{n-1} (-1)^i \binom{n-1}{i} \frac{1}{(i+1)^m} = \sum_{i=1}^{n} (-1)^{i-1} \binom{n}{i} \frac{1}{i^{m-1}}. \quad (5.9)$$

Чтобы оценить асимптотическое поведение $M(\pi_n)$ при $n \to \infty$, в (5.9) сделаем замену $u = nt$; тогда

$$M(\pi_n) = \frac{1}{(m-1)!} \int_0^n \left(1 - \frac{u}{n}\right)^n (\ln n - \ln u)^{m-1}\, du \sim$$

$$\sim \frac{1}{(m-1)!} \int_0^n e^{-u} \sum_{i=0}^{m-1} \binom{m-1}{i} (\ln n)^{m-1-i} (\ln u)^i\, du =$$

$$= \frac{1}{(m-1)!} (\ln n)^{m-1} \left(1 + O\left(\frac{1}{\ln n}\right)\right). \quad (5.10)$$

Здесь учитывается, что $\int_0^n e^{-u} (\ln u)^i\, du < \int_0^\infty e^{-u} (\ln u)^i\, du$ сходится.

Представляет интерес вывод еще одной зависимости для $M(\pi_n)$, которая получается при m-кратном интегрировании формулы (5.8):

$$M(\pi_n) = \int_0^1 \cdots \int_{(m-1)}^1 \frac{1 - (1 - x_1 \ldots x_{m-1})^n}{x_1 \ldots x_{m-1}}\, dx_1 \ldots dx_{m-1} =$$

$$= \int_0^1 \cdots \int_{(m-1)}^1 \frac{1 - (1 - x_1 \ldots x_{m-1})^n}{1 - (1 - x_1 \ldots x_{m-1})}\, dx_1 \ldots dx_{m-1} =$$

$$= \sum_{i=1}^n \int_0^1 \cdots \int_{(m-1)}^1 (1 - x_1 \ldots x_{m-1})^{i_1 - 1} dx_1 \ldots dx_{m-1} =$$

$$= \sum_{i_1=1}^n \frac{1}{i_1} \int_0^1 \cdots \int_{(m-2)}^1 \frac{1 - (1 - x_1 \ldots x_{m-2})^{i_1}}{1 - (1 - x_1 \ldots x_{m-2})}\, dx_1 \ldots dx_{m-2} =$$

$$= \sum_{i=1}^n \frac{1}{i_1} \sum_{i_2=1}^{i_1} \frac{1}{i_2} \cdots \sum_{m-1}^{i_{m-2}} \frac{1}{i_{m-1}} = \sum_{n \geqslant i_1 \geqslant \ldots \geqslant i_{m-1} \geqslant 1} \frac{1}{i_1 \ldots i_{m-1}}.$$

Легко видеть, что формула (5.10) получается также из следующего соотношения при $n \to \infty$:

$$(m-1)! \sum_{n \geqslant i_1 \geqslant \ldots \geqslant i_{m-1} \geqslant 1} \frac{1}{i_1 \ldots i_{m-1}} \sim \left(\sum_{i=1}^n \frac{1}{i}\right)^{m-1} \sim (\ln n)^{m-1}.$$

5.4.4. Оценка математического ожидания числа точек множества Парето в выборке случайных нормаль-

ных векторов. Пусть $\xi = \{\xi_i\}_{i=1}^N$ — последовательность случайных независимых, одинаково распределенных векторов. Будем говорить, что $\bar{i} \in P_\xi^N$, если для любого $t \neq \bar{i}$ не выполняется условие

$$\xi_i > \xi_{\bar{i}}, \quad 1 \leqslant \bar{i}, \quad i \leqslant N.$$

Пусть $P\{\xi_i \leqslant x\} = \bar{Q}(x)$, $P\{\xi_i > x\} = Q(x)$. Тогда

$$P\{i \in P_\xi^N / \xi_i = x\} = [1 - Q(x)]^{N-1},$$

$$P\{i \in P_\xi^N\} = \int [1 - Q(x)]^{N-1} d\mu_{\bar{Q}}(x),$$

где $\mu_{\bar{Q}}(x)$ — вероятностная мера, соответствующая распределению $\bar{Q}(x)$. Пусть $\left[P_\xi^N\right]$ — число элементов множества P_ξ^N; $M^N(Q)$ — математическое ожидание величины $\left[P_\xi^N\right]$:

$$M^N(Q) = NP\{i \in P_\xi^N\} = N \int [1 - Q(x)]^{N-1} d\mu_{\bar{Q}}(x).$$

Нас будет интересовать асимптотика изменения $M^N(Q)$, когда $N \to \infty$, а $\bar{Q}$ — многомерное нормальное распределение. Рассмотрим вначале случай, когда $\bar{Q}$ — двумерное нормальное распределение, $x = (x_1, x_2)$. Так как принадлежность к множеству Парето не зависит от масштаба измерения координат, то без ограничения общности будем рассматривать распределения следующего вида:

$$Q_\rho(\rho, d, \beta) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_\alpha^\infty \int_\beta^\infty e^{-\frac{1}{2(1-\rho^2)}(x^2 - 2\rho xy + y^2)}\, dx\, dy,$$

где ρ — коэффициент корреляции, $|\rho| < 1$. Сделаем замену переменных:

$$u = \frac{\sqrt{2}}{2}x - \frac{\sqrt{2}}{2}y, \quad v = \frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y;$$

тогда

$$x = \frac{\sqrt{2}}{2}u + \frac{\sqrt{2}}{2}v, \quad y = -\frac{\sqrt{2}}{2}u + \frac{\sqrt{2}}{2}v.$$

В новых координатах $M^N(Q_\rho)$ примет вид

$$M^N(Q_\rho) =$$

$$= \frac{N}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^\infty \int_{-\infty}^\infty e^{-\frac{1}{2}\left(\frac{u^2}{1-\rho} + \frac{v^2}{1+\rho}\right)} [1 - \varphi(u, v)]^{N-1}\, du\, dv,$$

где

$$\varphi(u, v) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{\infty} dx \int_{v+|u-x|}^{\infty} e^{-\frac{1}{2}\left(\frac{x^2}{1-\rho}+\frac{y^2}{1+\rho}\right)} dy.$$

Пусть $v \gg 1$. Тогда

$$\int_{v+|u-x|}^{\infty} e^{-\frac{1}{2}\frac{y^2}{1+\rho}} dy = \frac{(1+\rho)\, e^{-\frac{1}{2}\frac{(v+|x-u|)^2}{1+\rho}}}{v+|x-u|}\left[1+o\left(\frac{1}{v}\right)\right],$$

следовательно,

$$\varphi(u, v) =$$

$$= \frac{(1+\rho)\left[1+o\left(\frac{1}{v}\right)\right]}{2\pi\sqrt{1-\rho^2}} \int_{-\infty}^{\infty} \frac{\exp\left\{-\frac{1}{2}\left[\frac{1}{1-\rho}+\frac{(v+|x-u|)^2}{1+\rho}\right]\right\}}{v+|x-u|} dx =$$

$$= \frac{(1+\rho)\left[1+o\left(\frac{1}{v}\right)\right]}{2\pi\sqrt{1-\rho^2}} \times$$

$$\times \left[\int_{-\infty}^{u} \frac{\exp\left\{-\frac{1}{2}\left[\frac{x^2}{1-\rho}+\frac{(v+u-x)^2}{1+\rho}\right]\right\}}{v+u-x} dx + \right.$$

$$\left. + \int_{u}^{\infty} \frac{\exp\left\{-\frac{1}{2}\left[\frac{x^2}{1-\rho}+\frac{(v-u+x)^2}{1+\rho}\right]\right\}}{v-u+x} dx \right].$$

Вычислим интегралы, входящие в выписанную сумму. Имеем

$$\varphi_1 = \int_{-\infty}^{u} \frac{\exp\left\{-\frac{1}{2}\left[\frac{x^2}{1-\rho}+\frac{(v+u-x)^2}{1+\rho}\right]\right\}}{v+u-x} dx =$$

$$= \int_{-\infty}^{u} \frac{\exp\left\{-\frac{1}{2}\frac{1}{1-\rho^2}\left[2x^2-2x(1-\rho)(u+v)+(1-\rho)(u+v)^2\right]\right\}}{u+v-x} dx =$$

$$= (1-\rho^2) \times$$

$$\times \int_{-\infty}^{u} \frac{\exp\left\{-\frac{1}{2(1-\rho^2)}\left[2x^2-2x(1-\rho)(u+v)+(1-\rho)(u+v)^2\right]\right\}}{(u+v-x)\left[2x-(1-\rho)(u+v)\right]} \times$$

$$\times d\left[\frac{x^2-x(1-\rho)(u+v)}{1-\rho^2}\right].$$

При больших v и $(1+\varepsilon)u \leqslant \frac{1-\rho}{1+\rho}v$, $\varepsilon > 0$,

$$\varphi_1 = \frac{(1-\rho^2)\left[1+o\left(\frac{1}{v}\right)\right]\exp\left\{-\frac{1}{2}\left(\frac{u^2}{1-\rho}+\frac{v^2}{1+\rho}\right)\right\}}{v[(1-\rho)v-(1+\rho)u]}.$$

Аналогично получаем

$$\varphi_2 = \int\limits_{-u}^{\infty} \frac{\exp\left\{-\frac{1}{2}\left[\frac{x^2}{1-\rho}+\frac{(v-u+x)^2}{1+\rho}\right]\right\}}{v-u+x}dx =$$

$$= \frac{(1-\rho^2)\left[1+o\left(\frac{1}{v}\right)\right]\exp\left\{-\frac{1}{2}\left[\frac{u^2}{1-\rho}+\frac{v^2}{1+\rho}\right]\right\}}{v[(1+\rho)u+(1-\rho)v]}$$

при $(1+\varepsilon)u \geqslant -\frac{1-\rho}{1+\rho}v$, $v \gg 1$.

Таким образом, при больших v и $(1+\varepsilon)|u| \leqslant \frac{1-\rho}{1+\rho}v$

$$\varphi(u, v) = \frac{(1+\rho)\left[1+o\left(\frac{1}{v}\right)\right]}{2\pi\sqrt{1-\rho^2}}(\varphi_1+\varphi_2) =$$

$$= \frac{(1-\rho^2)^{3/2}\exp\left\{-\frac{1}{2}\left[\frac{u^2}{1-\rho}+\frac{v^2}{1+\rho}\right]\right\}\left[1+o\left(\frac{1}{v}\right)\right]}{\pi[(1-\rho)^2v^2-(1+\rho)^2u^2]}.$$

Сделаем замену переменных

$$\frac{u}{\sqrt{1-\rho}} = r\cos t, \quad \frac{v}{\sqrt{1+\rho}} = r\sin t.$$

Учитывая, что при $\varphi(u, v) > \frac{2\ln N}{N}$ $[1-\varphi(u, v)]^{N-1} < \delta_N = o\left(\frac{1}{N}\right)$, получим при $N \gg 1$ и $\varepsilon \to 0$:

$$M^N(Q_\rho) = \frac{N\left[1+o\left(\frac{1}{N}\right)\right]}{2\pi}\int\limits_{t_1}^{t_2} dt \int\limits_{R_N(t)}^{\infty} re^{-r^2/2}\times$$

$$\times\left\{1-\frac{(1-\rho^2)^{1/2}e^{-r^2/2}}{\pi r^2[(1-\rho)\sin^2 t-(1+\rho)\cos^2 t]}\right\}^{N-1}dr,$$

где

$$t_1 = \frac{\pi}{2} - \operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}},$$

$$t_2 = \frac{\pi}{2} + \operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}},$$

$R_N(t)$ — корень уравнения (относительно r)

$$\frac{2\ln N}{N} = \frac{(1-\rho^2)^{1/2} e^{-r^2/2}}{\pi r^2 [(1-\rho)\sin^2 t - (1+\rho)\cos^2 t]}.$$

Таким образом, получаем

$$M^N(Q_\rho) = \frac{N\left[1 + o\left(\frac{1}{N}\right)\right]}{2\pi} \times$$

$$\times \int_{t_1}^{t_2} dt \int_0^{e^{-R_N^2(t)/2}} \exp\left\{\frac{(N-1)\, z\, (1-\rho^2)^{1/2}}{2\pi \ln z [(1-\rho)\sin^2 t - (1+\rho)\cos^2 t]}\right\} dz.$$

Для оценки последнего интеграла докажем следующую лемму.

Лемма 5.1.

$$\int_0^R \exp\left\{\frac{cz}{\ln z}\right\} dz = \frac{\ln c}{c}[1 + \delta(c)], \quad c > 1, \quad 0 < R < 1,$$

причем $\delta(c) \to 0$ *при* $c \to \infty$.

Доказательство. Зафиксируем некоторое число $k \gg 1$; тогда

$$\int_0^R \exp\left\{\frac{cz}{\ln z}\right\} dz = \int_0^{\ln c/(kc)} \exp\left\{\frac{cz}{\ln z}\right\} dz +$$

$$+ \int_{\ln c/(kc)}^{k\ln c/c} \exp\left\{\frac{cz}{\ln z}\right\} dz + \int_{k\ln c/c}^{R} \exp\left\{\frac{cz}{\ln z}\right\} dz.$$

В промежутке $z \in [0,\ \ln c/(kc)]$ $1 \geqslant \exp\{cz/\ln z\} \geqslant e^{-2/k}$. В промежутке $[\ln c/(kc),\ k \ln c/c]$

$$\frac{cz}{\ln z} = \frac{-\frac{cz}{\ln c}\cdot \ln c}{\ln\left(-\frac{cz}{\ln c}\right) + \ln\ln c - \ln c} =$$

$$= -\frac{cz/\ln c}{1 - \ln\left(-\frac{cz}{\ln z}\right)\Big/\ln c - \ln\ln c/\ln c} = -\frac{cz}{\ln c}[1 + \delta_z(c)],$$

где $\delta_z(c) \to 0$ при $c \to \infty$ равномерно по z. Отсюда

$$\int_{\ln c/(kc)}^{k\ln c/c} \exp\left\{\frac{cz}{\ln z}\right\} dz \cong \int_{\ln c/(kc)}^{k\ln c/c} \exp\left\{-\frac{cz}{\ln c}\right\} dz = \int_{1/k}^{k} \frac{\ln c}{c} e^{-t}\, dt =$$

$$= \frac{\ln c}{c}\left(e^{-1/k} - e^{-k}\right),$$

где $A(x) \cong B(x)$ означает $\lim\limits_{x\to\infty} \dfrac{A(x)}{B(x)} = 1$. Наконец,

$$\int_{k\ln c/c}^{R} \exp\left\{\frac{cz}{\ln z}\right\} dz \leqslant \int_{k\ln c/c}^{R} \exp\left\{-\frac{cz}{k\ln(c/k\ln c)}\right\} dz \leqslant \frac{ke^{-k}\ln c}{c}.$$

Из этих трех неравенств при условии, что k может быть сколь угодно большим, получаем утверждение леммы.

Из леммы 5.1 имеем

$$M^N(Q_\rho) = \frac{N\left[1 + o\left(\frac{1}{N}\right)\right]\ln(N-1)}{(N-1)\sqrt{1-\rho^2}}\left(\int_{t_1}^{t_2}[(1-\rho)\sin^2 t -\right.$$

$$-(1+\rho)\cos^2 t]\,dt - \frac{1}{\ln(N-1)}\int_{t_1}^{t_2}\{\ln[(1-\rho)\sin^2 t -$$

$$\left.-(1+\rho)\cos^2 t] + \ln(2\pi) - \ln(1-\rho^2)\}[(1-\rho)\sin^2 t -\right.$$

$$\left.-(1+\rho)\cos^2 t]\,dt\right) \cong \frac{\ln N}{\sqrt{1-\rho^2}}\left[\sin\left(2\operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}}\right) -\right.$$

$$\left.-2\rho\operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}}\right] = \frac{\ln N}{\sqrt{1-\rho^2}}\left(\sqrt{1-\rho^2} - 2\rho \times\right.$$

$$\left.\times \operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}}\right) = \ln N\left(1 - \frac{2\rho}{\sqrt{1-\rho^2}}\operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}}\right).$$

Таким образом, в двумерном случае получаем следующую асимптотическую формулу:

$$M^N(Q_\rho) \cong \ln N\left(1 - \frac{2\rho}{\sqrt{1-\rho^2}}\operatorname{arctg}\sqrt{\frac{1-\rho}{1+\rho}}\right).$$

При $\rho = 0$ получаем $M^N(Q_\rho) \cong \ln N$. Этот результат для случая независимых компонент можно найти другим путем. Он сравнительно легко обобщается на случай размерности три.

В случае независимых нормальных компонент в m-мерном пространстве имеем

$$Q(x) = \left(\frac{1}{\sqrt{2\pi}}\right)^m \int_{a_1}^{\infty} \cdots \int_{a_m}^{\infty} \prod_{i=1}^{m} e^{-x_i^2/2}\, dx_i,$$

а интеграл для $M^N(Q)$ принимает вид

$$M^N(Q) = N \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} [1 - Q(x)]^{N-1} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx_i.$$

Далее, поскольку при большом n и $z \cong 1/n$ имеем

$$(1-z)^n \cong e^{-nz},$$

получим для $M^N(Q)$:

$$M^N(Q) \cong N \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp\left\{-(N-1)\prod_{i=1}^{m}[1-\Phi_i(x_i)]\right\} \times \times \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} dx_i,$$

где

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} dt$$

— функция распределения вероятностей стандартной нормальной случайной величины.

После замены переменных

$$u_i = 1 - \Phi_i(x_i), \qquad i = 1, \ldots, m,$$

получим

$$M^N(Q) \cong N \int_0^1 \ldots \int_0^1 \exp\left\{-(N-1)\prod_{i=1}^{m} u_i\right\} \prod_{i=1}^{m} du_i.$$

Далее, интеграл для $M^N(Q)$ вычисляется для больших N в дву- и трехмерном случаях. При этом используется следующая техника интегрирования: фиксируется значение параметра t, соответствующего линии (поверхности) уровня подынтегральной функции, и находится выражение $v(t)$ для площади (объема), ограниченной значением t, затем находится плотность $v'_t(t)$ и по ней

интегрируется функция $e^{-(N-1)t}$. Везде, где это необходимо, используются стандартные интегралы из справочника [15].

В двумерном случае требуется взять интеграл

$$M^N(Q) \cong \bar{N} \int_0^1 \int_0^1 e^{-(N-1)xy} dx\, dy.$$

Делаем замену переменных

$$xy = t.$$

Выражение для площади части плоскости, заключенной между осями x, y и кривой $e^{-(N-1)xy}$, $xy = t$, равно

$$v(t) = t + \int_0^1 \frac{t}{x}\, dx = t - t \ln t.$$

Далее, функция $v(t)$ дифференцируется по t:

$$v'_t = -\ln t,$$

а выражение для $M^N(Q)$ принимает вид

$$N^{-1}M^N(Q) \cong -\int_0^1 e^{-(N-1)t} \ln t\, dt =$$

$$= -\int_0^\infty e^{-(N-1)t} \ln t\, dt + \int_1^\infty e^{-(N-1)t} \ln t\, dt.$$

Второй интеграл в последнем выражении — величина порядка $o\left(\frac{1}{N-1}\right)$. Действительно,

$$\int_1^\infty e^{-(N-1)x} \ln x\, dx = -\frac{1}{N-1} \operatorname{Ei}[-(N-1)] = o\left(\frac{1}{N-1}\right)$$

(здесь $\operatorname{Ei}(z)$ — интегральная показательная функция).

Следовательно,

$$M^N(Q) \cong -N \int_0^\infty e^{-(N-1)t} \ln t\, dt =$$

$$= N(N-1)^{-1}[C + \ln(N-1)] \cong \ln(N-1)$$

([15]; здесь C — постоянная Эйлера).

В трехмерном случае требуется взять интеграл

$$M^N(Q) \simeq N \int_0^1 \int_0^1 \int_0^1 e^{-(N-1)xyz}\, dx\, dy\, dz.$$

Техника интегрирования остается прежней, а именно делается замена $t = xyz$, затем находится выражение для объема $v(t)$:

$$v(t) = t - t \ln t + t \int_t^1 \int_{t/y}^1 \frac{1}{xy}\, dx\, dy.$$

Имеем

$$\int_t^1 \int_{t/y}^1 \frac{dx\, dy}{xy} = \int_t^1 \left(\int_{t/y}^1 \frac{dx}{x} \right) \frac{dy}{y} = -\int_t^1 \frac{1}{y} \ln\left(\frac{t}{y}\right) dy.$$

Сделаем замену

$$u = \ln\left(\frac{t}{y}\right), \quad du = \frac{y}{t}\left(-\frac{t}{y^2}\right) dy = -\frac{dy}{y},$$

получим

$$-\int_t^1 \frac{1}{y} \ln\left(\frac{t}{y}\right) dy = \int_0^{\ln t} u\, du = \frac{1}{2} t (\ln t)^2.$$

Тогда

$$v(t) = t - t \ln t + \frac{1}{2} t (\ln t)^2, \quad v_t'(t) = \frac{1}{2} (\ln t)^2,$$

$$M^N(Q) \simeq N \int_0^1 e^{-(N-1)t} \frac{1}{2} (\ln t)^2\, dt.$$

Последний интеграл не более

$$N \int_0^\infty \frac{1}{2} e^{-(N-1)t} (\ln t)^2\, dt \simeq \frac{1}{2} \left\{ \frac{\pi^2}{6} + [c + \ln(N-1)]^2 \right\}$$

[15]. Окончательно имеем

$$M^N(Q) \simeq \frac{1}{2} (\ln N)^2,$$

что подтверждает результаты раздела 5.4.3.

Библиографический комментарий

Метод последовательного анализа вариантов впервые описан в [1, 2]; математическое обоснование метода — в [3, 4]; развитие метода применительно к задачам теории расписаний — в [6 — 8]; ряд примеров использования метода можно найти в книге [5]. В рамки последовательного анализа вариантов вписываются многие другие вычислительные схемы, предложенные для сетевых задач распределения ресурсов и составления расписаний [9 — 12], задач дискретного программирования [16], задач динамического размещения производства [17]. Охарактеризуем вкратце содержание данной главы. Общее описание метода — согласно [5]; формулировка задачи объемного календарного планирования и сведение ее к нелинейной задаче о рюкзаке — новые; алгоритм последовательного анализа вариантов в нелинейной задаче о рюкзаке — согласно [13]; оценка трудоемкости алгоритма в среднем — согласно [14].

6.1. Основные понятия и структура алгоритмов

Алгоритмы ветвей и границ могут быть описаны с использованием нескольких ключевых понятий: ветвление, релаксация, тест, стратегия.

Ветвление. Для задачи оптимизации (P) пусть $F(P)$ обозначает множество допустимых решений. Говорят, что задача (P) ветвится на подзадачи (P_1), ..., (P_q), если выполнены условия:

1) каждое допустимое решение (P) является допустимым решением ровно одной из подзадач (P_1), ..., (P_q);

2) допустимое решение каждой из подзадач является допустимым решением (P).

Иными словами, ветвление задачи (P) эквивалентно разбиению $F(P)$ на непересекающиеся множества $F(P_1)$, ..., $F(P_q)$. Подзадачи (P_1), ..., (P_q) называются потомками (P). Порождение потомков от потомков (P) эквивалентно дальнейшему разбиению $F(P)$.

В общих чертах стратегия решения задачи (P) такова. Попытаться решить задачу (P) разумной ценой (например, за ограниченное время). Если попытка безуспешна, ветвить задачу (P), образовав тем самым список кандидатов. Выбрать одного из кандидатов — назовем его (CP) — и попытаться решить задачу (CP). Если попытка успешна, запомнить результат, исключить (CP) из списка, возвратиться к списку и выбрать нового кандидата. Иначе ветвить (CP), добавить его потомков к списку кандидатов, а (CP) исключить из списка. Продолжать таким образом, пока список кандидатов не будет исчерпан.

Наилучшее из всех решений задач-кандидатов, рассмотренных (проанализированных) к данному шагу вычислительного процесса, называется рекордным, а соответствующее значение целевой функции — рекордом. Окончательный рекорд должен быть оптимальным реше-

нием задачи (P) (если все (CP) были недопустимы, то и (P) не имеет решений). Очевидно, если $F(P)$ конечно, процесс завершается за конечное число шагов. Эффективность алгоритма зависит от успешности решения подзадач до их ветвления.

Релаксация. Под релаксацией понимается переход от задачи (P) с допустимой областью $F(P)$ к задаче P_R с допустимой областью $F(P_R) \supset F(P)$. Если (P) — задача минимизации, то это приводит к тому, что

A_1. Если (P_R) не имеет допустимых решений, то (P) также их не имеет.

A_2. Значение минимума в P не меньше, чем в (P_R).

A_3. Если оптимальное решение (P_R) допустимо для (P), то оно есть оптимальное решение (P).

Задача (P_R) получается из (P) отбрасыванием (неучетом) некоторых ограничений задачи, например ограничений целочисленности. Выбор того или иного способа релаксации связан с необходимостью обеспечения двух, вообще говоря, несовместимых требований: с одной стороны, релаксированная задача должна быть в определенном смысле проще исходной; с другой стороны, если оптимальное решение релаксированной задачи недопустимо для (P), то оно должно быть как можно ближе к (P) по значению целевой функции.

Тест. Пусть (CP) — задача-кандидат. Конечная цель анализа состоит в том, чтобы определить, содержит ли $F(CP)$ оптимальное решение задачи (P), и если да, то найти его. Говорят, что задача (CP) проанализирована, если установлено, что $F(CP)$ не содержит решения, лучшего чем рекорд, либо найдено оптимальное решение (CP).

Далее, (CP_R) обозначает некоторую релаксацию кандидата (CP); $v(\cdot)$ — оптимальное решение задачи $(\cdot)$; z^* — значение рекорда ($z^* = +\infty$, если не найдено ни одного допустимого значения (P)). Полезно различать три результата анализа.

FC_1. Анализ задачи (CP_R) обнаруживает, что она не имеет допустимых решений, т. е. $F(CP_R) = \varnothing$.

По условию A_1 это же верно для (CP), следовательно, (CP) не может содержать оптимального решения (P).

FC2. Анализ задачи (CP_R) обнаруживает, что задача (CP_R) не имеет допустимых решений, лучших чем рекорд, т. е. $v(CP_R) \geqslant z^*$.

По условию A2 $v(CP) \geqslant v(CP_R)$. Следовательно, $F(CP)$ не содержит решений, лучших чем рекорд.

FC3. В результате решения задачи (CP_R) найдено оптимальное решение, которое является допустимым для CP.

По условию A_3 оно является и оптимальным для (CP).

Стандартный анализ задачи-кандидата состоит в том, чтобы найти оптимальное решение ее релаксации, а затем проверить условия FC1 — FC3.

Стратегия в алгоритмах ветвей и границ определяется порядком выбора задач — кандидатов для их последующего анализа.

Практические сетевые задачи оптимального распределения нескладируемых ресурсов, как правило, имеют большую размерность. В то же время экспериментальные оценки трудоемкости алгоритмов ветвей и границ имеют экспоненциальную зависимость от размеров задач. В связи с этим описанная выше стандартная структура алгоритмов может оказаться практически неэффективной. Введение ограничений на время счета (или общее число подзадач) и на общую память алгоритма приводит в общем случае к погрешности счета. В этом случае результатом алгоритма является приближенное решение исходной задачи оптимизации с гарантированной апостериорной относительной погрешностью по функционалу

$$\varepsilon = \frac{|z^* - v^*(CP_R)|}{|v^*(CP_R)|}.$$

Здесь $v^*(CP_R) = \min v(CP_R)$, где min берется по всем подзадачам (CP), находящимся в списке подзадач в момент останова. Может оказаться также, что начальное значение рекорда z^* не модифицируется за время счета. Для ускорения процесса уменьшения рекорда в алгоритм ветвей и границ, ориентированный на приближенное решение задачи, следует добавить модуль, предназначенный для вычисления верхних границ оптимального значения функционала в задачах (CP). Поскольку в исследуемых нами экстремальных задачах оптимального распределения ограниченных ресурсов найти какое-то допустимое решение исходной задачи сравнительно легко, для получения верхних оценок годится любой известный эвристический алгоритм. Таким образом, эффективность алгоритмов ветвей и границ существенно определяется успешностью решения ряда внутренних подзадач. Наиболее важная подзадача состоит в оценивании границы

функционала на подмножестве допустимых решений исходной задачи оптимизации (оценочная задача). Следующие разделы 6.2—6.4 посвящены методам решения оценочных задач.

6.2. Методы теории двойственности для решения оценочных задач

6.2.1. Метод множителей Лагранжа в задачах дискретной оптимизации. Множители Лагранжа могут быть использованы в задачах целочисленного и дискретного программирования с целью конструирования эффективных алгоритмов в тех случаях, когда эти задачи обладают определенными структурными особенностями. Более того, стремление обнаружить и полезно использовать особенности структуры задач нередко приводит к формулировкам новых классов эффективно решаемых задач дискретного программирования, тем самым расширяя область применения соответствующих алгоритмов.

Рассмотрим задачи следующего вида. Определить

$$v = \min cx$$

при условиях

$$\begin{gathered} Ax \leqslant b, \\ x \in X \subseteq \mathbf{R}^n, \end{gathered} \tag{6.1}$$

где X — дискретное множество со специальной структурой. Например, X может определяться ограничениями вида

$$\sum_{j \in J_k} x_j = 1 \quad \text{для всех } k \text{ из некоторого множества } K,$$

$$x_j = 0 \quad \text{или} \quad 1,$$

причем множества J_k не пересекаются. Этот случай, в частности, характерен для задач типа «станки — детали». В сетевых задачах составления расписаний общего вида множество X может определяться вполне унимодулярной системой линейных неравенств. Указанные случаи представлены в разделе 6.3.

Лагранжиан в задаче (6.1) определяется для произвольного $u \geqslant 0$ как

$$L(u) = -ub + \min_{x \in X} (c + uA)\, x. \tag{6.2}$$

Предполагается, что из-за наличия специальной структуры у множества X $L(u)$ вычислять проще, чем v, или, в терминах главы II, задача (6.2) является менее сложной, чем исходная задача (6.1). Во многих случаях оптимальное значение x в (6.2) является целочисленным и может быть хорошим начальным приближением для использования эвристических или локальных методов в исходной задаче (6.1). Наконец, метод множителей Лагранжа часто применим и для решения дискретных оптимизационных задач, представление которых в терминах целочисленного линейного программирования или неизвестно, или неудобно для работы.

Оптимальный выбор множителей Лагранжа основан на использовании теории двойственности. Пусть исходная дискретная задача оптимизации имеет вид: определить

$$v = \min f(x)$$

при условиях

$$\begin{aligned} g(x) &\leqslant b, \\ x &\in X \subseteq \mathbf{R}^n, \end{aligned} \tag{6.3}$$

где f — скалярная функция, определенная на $\mathbf{R}^n$; g — вектор-функция из $\mathbf{R}^n$ в $\mathbf{R}^m$; X — дискретное множество. Если нет ни одного $x \in X$, удовлетворяющего условию $g(x) \leqslant b$, принимаем $v = \infty$. Без потерь в общности можно положить, что X — конечное множество, причем $X = \{x^t \mid t = 1, \ldots, T\}$.

В методе множителей Лагранжа с каждым ограничением $g(x) \leqslant b$ связывается неотрицательная переменная u. Лагранжиан в задаче (6.3) имеет вид

$$\begin{aligned} L(u) &= -ub + \min_{x \in X} \{f(x) + ug(x)\} = \\ &= -ub + \min_{t=1,\ldots,T} \{f(x^t) + ug(x^t)\}. \end{aligned} \tag{6.4}$$

Поскольку X — конечное множество, функция $L(u)$ определена для всех $u \in \mathbf{R}^m_+$. Более того, можно показать [25], что эта функция является непрерывной, вогнутой, но, вообще говоря, не является дифференцируемой.

Отметим две отличительные особенности применения множителей Лагранжа в дискретном программировании по сравнению с непрерывными задачами. Первая — это комбинаторная природа алгоритмов вычисления лагран-

жиана (6.4). Вторая — это недифференцируемость функции $L(u)$, что является следствием дискретности X. Эта особенность делает двойственную задачу задачей недифференцируемой оптимизации. Задачи (6.3) и (6.4) образуют пару двойственных задач.

Установим, каким условиям должны удовлетворять множители u, чтобы значения x, входящие в оптимальное решение задачи (6.4), были таковыми в исходной задаче (6.3). Будем говорить, что пара $(\bar{u}, \bar{x})$, где $\bar{x} \in X$ и $\bar{u} \geqslant 0$, удовлетворяет условиям оптимальности в дискретной оптимизационной задаче (6.3), если

1) $L(\bar{u}) = -\bar{u}b + f(\bar{x}) + \bar{u}g(\bar{x})$;

2) $\bar{u}(g(\bar{x}) - b) = 0$;

3) $g(\bar{x}) \leqslant b$.

Теорема 6.1. *Если пара* $(\bar{x}, \bar{u})$ *удовлетворяет условиям оптимальности в задаче дискретного программирования* (6.3), *то* $\bar{x}$ *— оптимальное решение.*

Доказательство. Решение $\bar{x}$ допустимо для задачи (6.3), поскольку $\bar{x} \in X$ и $g(\bar{x}) \leqslant b$ по условию 3). Пусть $x \in X$ — другое допустимое решение задачи (6.3). Тогда, по условию 1),

$$L(\bar{u}) = -\bar{u}b + f(\bar{x}) + \bar{u}g(\bar{x}) \leqslant -\bar{u}b + f(x) + \bar{u}g(x) \leqslant f(x)$$

(последнее неравенство следует из $u \geqslant 0$ и $g(x) - b \leqslant 0$). Но по условию 2) $L(\bar{u}) = f(\bar{x})$, поэтому $f(\bar{x}) \leqslant f(x)$ для всех допустимых x.

При доказательстве теоремы 6.1 мы получили следующий важный результат.

Следствие 6.1 (слабая двойственность). *Для всех* $u \geqslant 0$ *справедливо неравенство* $L(u) \leqslant v$, *где* v *— оптимальное решение задачи* (6.4).

Нашей основной целью является выбор такого u, которое делает нижнюю границу наибольшей, или, другими словами, является оптимальным значением в задаче: найти

$$d = \max L(u) \tag{6.5}$$

при условиях $u \geqslant 0$.

Следствие 6.2. *Если пара* $(\bar{x}, \bar{u})$ *удовлетворяет условию оптимальности в задаче дискретного программирования* (6.3), *то* $\bar{u}$ *— оптимальное значение в задаче* (6.5).

Доказательство. По теореме 6.1 имеем

$$L(\bar{u}) = -\bar{u}b + f(\bar{x}) + \bar{u}g(\bar{x}) = f(\bar{x}) = v.$$

Поскольку $L(u) \leqslant v$ для всех $u \geqslant 0$ (следствие 6.1), получаем $L(u) \leqslant L(\bar{u})$ для всех $u \geqslant 0$.

Непосредственный способ использования множителей Лагранжа предусматривает, во-первых, решение двойственной задачи и определение некоторого $\bar{u}$. Затем, вычислив одно или несколько значений x таких, что $L(\bar{u}) = -\bar{u}b + f(x) + \bar{u}g(x)$, следует попытаться найти некоторое $\bar{x} \in X$, для которого выполняются условия оптимальности. Однако нет гарантии, что эта стратегия окажется успешной, если

а) найденное $\bar{u}$ таково, что условия оптимальности не выполняются ни для одной пары $(\bar{x}, \bar{u})$, где $x \in X$;

б) найденное $\bar{x}$ из X (или несколько таковых), выбранное в результате минимизации лагранжиана, не удовлетворяет условиям оптимальности, хотя существует другое оптимальное $\tilde{x} \in X$, для которого условия оптимальности выполняются.

Более общий способ использования теоремы 6.1 и следствий 6.1 и 6.2 состоит во включении процедуры решения двойственной задачи в метод ветвей и границ, позволяющий организовать упорядоченный анализ всех точек x из X на предмет обнаружения пары $(\bar{x}, \bar{u})$. Этот способ описан в следующем разделе.

Отметим еще одно полезное качество метода множителей Лагранжа. Для всякого $u \geqslant 0$ из условий оптимальности следует, что некоторое $\bar{x}$ такое, что

$$L(\bar{u}) = -\bar{u}b + f(\bar{x}) + \bar{u}g(\bar{x}),$$

оптимально в такой задаче (6.3), в которой b заменено на $b + \delta$, где δ неотрицательно, и $\delta_i = 0$, когда $\bar{u}_i > 0$. Следовательно, методы множителей Лагранжа могут быть использованы для приближенного решения задач вида (6.3), когда ограничения $g(x) \leqslant b$ не являются жесткими. Даже если ограничения $g(x) \leqslant b$ — жесткие и получено некоторое недопустимое решение $\bar{x}$ задачи (6.3), для которого $g(\bar{x}) = b + \delta$ и δ достаточно мало, появляется возможность приближенного решения задачи путем преобразования $\bar{x}$ в некоторое допустимое значение x посредством эвристических алгоритмов, учитывающих структуру задачи (6.3).

Имеются два внешне отличных друг от друга, но внутренне тесно связанных подхода к решению двойственной задачи (6.5). Первый из них предполагает использование методов негладкой оптимизации. Хотя функ-

ция $L(u)$ не всюду дифференцируема, для максимизации $L(u)$ могут быть использованы методы, основанные на обобщениях понятия градиента. Вектор γ называется субградиентом L в точке $\bar{u}$, если

$$L(u) \leqslant L(\bar{u}) + (u - \bar{u})\bar{\gamma}$$

для всех u, $0 < u < \infty$. Можно показать, что для каждого субградиента полупространство

$$\{u \mid (u - \bar{u})\bar{\gamma} \geqslant 0\}$$

содержит все решения двойственной задачи с большими значениями L, чем $L(\bar{u})$. Другими словами, субградиент указывает направление неубывания функции $L(u)$ в точке $\bar{u}$. В частности, субградиентом является вектор

$$\bar{\gamma} = g(\bar{x}) - b, \qquad (6.6)$$

где $\bar{x}$ — некоторое решение из X, удовлетворяющее условию $L(\bar{u}) = -\bar{u}b + f(\bar{x}) + \bar{u}g(\bar{x})$. Если имеется единственное $\bar{x} \in X$, минимизирующее L в точке $\bar{u}$, то L дифференцируема в этой точке, а $\bar{\gamma}$ является градиентом.

В задаче (6.5) метод субградиентной оптимизации может состоять в генерации последовательности $\{u^k\}$, $k = 0, 1, 2, \ldots$, неотрицательных решений по способу

$$u_i^{k+1} = \max\{0, u_i^k + \theta_k \gamma_i^k\}, \qquad i = 1, \ldots, m,$$

где γ^k — субградиент, выбранный в соответствии с (6.6), а θ_k — длина шага. Например, если множитель θ_k удовлетворяет условиям $\theta_k \to +0$, $\sum_{k=1}^{\infty} \theta_k \to +\infty$, то можно показать, что $L(u^k)$ стремится к d, оптимальному решению двойственной задачи (6.5). Заметим, что при использовании методов субградиентной оптимизации нет гарантии, что $L(u^{k+1}) > L(u^k)$.

Второй подход к решению задачи (6.5) связан с представлением ее в форме задачи линейного программирования большой размерности. А именно, задача (6.5) эквивалентна следующей задаче линейного программирования: найти

$$d = \max v$$

при условии

$$\begin{aligned} &v \leqslant -ub + f(x^t) + ug(x^t), \qquad t = 1, \ldots, T, \\ &u \geqslant 0. \end{aligned} \qquad (6.7)$$

Задача (6.7) действительно является задачей большой размерности, так как общее число ограничений T нередко имеет экспоненциальный порядок роста. Например, если D — период планирования в задаче составления расписаний, а N — общее число операций, то T — общее число расписаний — имеет порядок N^D.

Рассмотрим задачу линейного программирования, двойственную к (6.7): найти

$$d = \min \sum_{t=1}^{T} f(x^t)\lambda_t \tag{6.8a}$$

при условиях

$$\sum_{t=1}^{T} g(x^t)\lambda_t \leqslant b, \tag{6.8b}$$

$$\sum_{t=1}^{T} \lambda_t = 1, \tag{6.8c}$$

$$\lambda_t \geqslant 0, \quad t = 1, \ldots, T. \tag{6.8d}$$

В декомпозируемых задачах вида (6.3), например в задаче составления расписаний с независимыми цепями операций, двойственная задача в форме (6.8 a—d) имеет ограничение (6.8c) для каждой компоненты лагранжиана (цепи операций). Число таких ограничений — M (количество цепей операций), а число общих ограничений (6.8b) — D (длительность интервала планирования). Если $M > D$, то оптимальное решение задачи (6.8), найденное посредством симплекс-метода, по крайней мере для $(M - D)$ цепей будет иметь чистые стратегии, для которых соответствующая переменная будет равна 1. В случае $M \gg D$ метод множителей Лагранжа даст хорошее приближение, поскольку чистые стратегии будут выбраны почти для всех цепей. Иначе говоря, так называемый разрыв двойственности, т. е. разность оптимальных значений функционалов прямой и двойственной задач, в этом случае будет мал.

Заметим, что если в (6.8 a—d) потребовать, чтобы λ_t были целыми, то задача (6.8 a—d) была бы эквивалентна прямой задаче (6.3). Более того, решение двойственной задачи является решением прямой задачи (6.3), если в (6.8 a—d) найдется ровно одно положительное λ_t. Например, пусть $\lambda_r = 1$, $\lambda_t = 0$ при $t \neq r$. В этом случае $x^r \in X$ является оптимальным решением прямой задачи и оно найдено методом множителей Лагранжа. Обратное,

предположим, что в оптимальном решении задачи (6.8 a—d) более чем одно λ_t положительно, например, $\lambda_1 > 0, \ldots, \lambda_r > 0$; $\lambda_t = 0$, $t \geqslant r+1$. Тогда следует ожидать, что решение $\sum_{t=1}^{r} \lambda_t x^t$ не принадлежит X, поскольку X — дискретное множество и решение двойственной задачи не дает решения прямой. Даже если $y = \sum_{t=1}^{r} \lambda_t x^t$ принадлежит X, нет гарантии, что y оптимально, так как условия оптимальности 2) и 3) могут не выполняться для y.

В задачах (6.7) и (6.8 a—d) годятся обобщенные методы линейного программирования, приспособленные для решения задач большой размерности. Одной из возможностей является метод Данцига — Вульфа. В этом случае координирующая (связывающая) задача оперирует с подмножеством столбцов задачи (6.8 a—d). Потенциально новый столбец для введения в координирующую задачу определяется из $\bar{x} \in X$, удовлетворяющего условию $L(-\bar{\pi}) = +\bar{\pi}b + f(\bar{x}) - \bar{\pi}g(\bar{x})$, где $\bar{\pi} \leqslant 0$ — вектор оптимальных двойственных оценок строк (6.8b) координирующей задачи. Если $L(-\bar{\pi}) < \bar{\pi}b + \bar{\theta}$, где $\bar{\theta}$ — оптимальная двойственная оценка строки (6.8c), то новый столбец $(f(\bar{x}), g(\bar{x}), 1)^T$ добавляется в координирующую задачу с новой переменной λ. Если $L(-\bar{\pi}) = \bar{\pi}b + \bar{\theta}$ (случай $>$ невозможен), то оптимальное решение координирующей задачи является оптимальным в задаче (6.8 a—d).

6.2.2. Использование множителей Лагранжа в методе ветвей и границ. Как следует из предыдущего раздела, метод множителей Лагранжа в общем случае не гарантирует точного решения задачи дискретного программирования (6.3). Для полного решения задачи (6.3) необходимо организовать систематический поиск решений на дискретном множестве X. Такой поиск может быть организован методом ветвей и границ. Эффективность алгоритмов ветвей и границ в основном определяется точностью оценок (границ), используемых для ограничения области поиска. По теореме 6.1 и следствию 6.1, метод множителей Лагранжа указывает способ решения оценочных задач исходной задачи (6.3). Более того, анализ множителей Лагранжа может быть использован для решения ряда других вспомогательных задач, возникающих в алгоритме ветвей и границ, например, выбора переменной для ветвления, получения допустимого ре-

шения из решения оценочной задачи. Обратное, метод ветвей и границ можно представлять как метод последовательного преобразования исходной задачи (6.3), когда метод множителей Лагранжа в чистом виде не дает ее решения.

Применительно к задаче (6.3) опишем алгоритм ветвей и границ в сочетании с методом множителей Лагранжа. Процесс вычислений определяется, как обычно, последовательностью шагов. На каждом шаге процесса известно некоторое допустимое решение $\hat{x} \in X$ такое, что $g(\hat{x}) \leqslant b$, имеющее минимальную стоимость $\hat{z} = f(\hat{x})$. $\hat{x}$ называется рекордом, а $\hat{z}$ — значением рекорда. Алгоритм ветвей и границ генерирует последовательность задач вида: найти

$$v(X^k) = \min f(x) \tag{6.9}$$

при условиях

$$g(x) \leqslant b,$$

$$x \in X^k,$$

где $X^k \subseteq X$. Множества X^k выбираются таким образом, чтобы сохранить специальную структуру X. В соответствии с общим описанием метода ветвей и границ, если найдено оптимальное решение задачи (6.9), все подзадачи вида (6.9) с $X^l \subseteq X^k$ считаются протестированными (проверенными) и не подлежат дальнейшему анализу в процессе счета. Тот же результат имеет место, если возможно установить, что $v(X^k) \geqslant \hat{z}$ без явного вычисления точного значения $v(X^k)$. В обоих случаях считается, что подзадача (6.9) исключена. В противном случае она разбивается на подзадачи вида (6.9) с множествами X^l вместо X^k такими, что

$$\bigcup_{l=1}^{L} X^l = X^k, \quad X^{l1} \cap X^{l2} = \varnothing, \quad l1 \neq l2.$$

Метод множителей Лагранжа используется с целью исключения подзадач вида (6.9) посредством решения двойственных к ним задач: найти

$$d(X^k) = \max L(u, X^k) \tag{6.10}$$

при условии

$$u \geqslant 0,$$

где

$$L(u, X^k) = -ub + \min_{x \in X^k} (f(x) + ug(x)). \tag{6.11}$$

Использование двойственной задачи (6.10) в анализе задачи (6.9) иллюстрируется на рис. 6.1. Обсудим этот процесс по шагам.

Шаги 1 и 2. Начальный список подзадач состоит из единственной подзадачи, соответствующей множеству X.

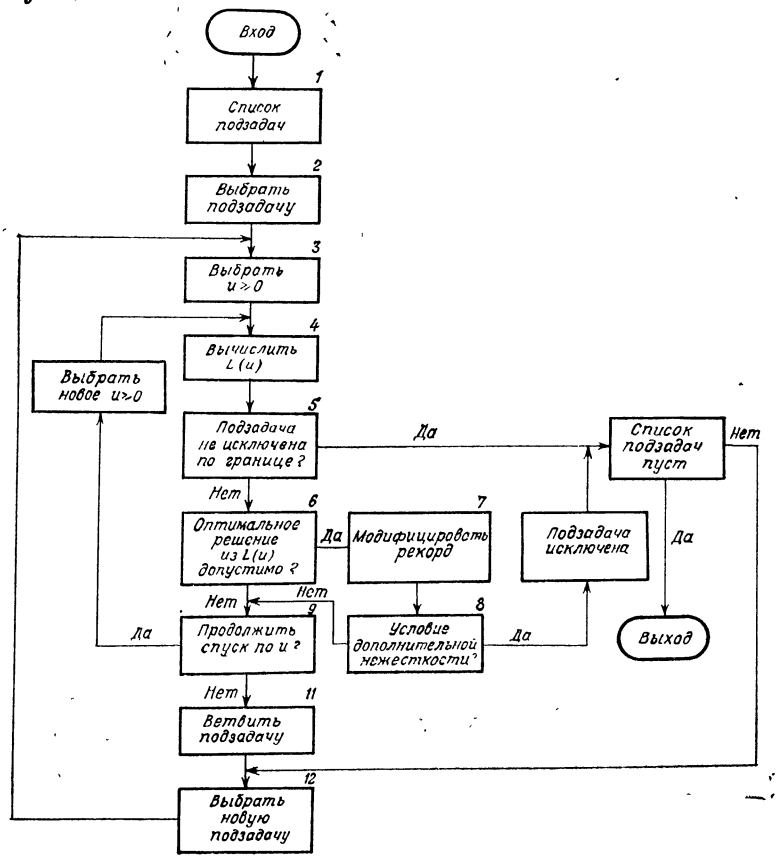

Рис. 6.1.

Шаг 3. Хорошее начальное приближение в двойственной задаче может быть установлено пользователем, исходя из опыта предыдущих вычислений.

Шаг 4. Вычисление собственно лагранжиана может представлять собой решение некоторой вспомогательной оптимизационной задачи, например задачи о максимальном потоке, задачи о кратчайшем пути на графе и т. п.

Шаг 5. Результат шага 4: имеется нижняя граница $L(\bar{u}, X^k)$ величины $v(X^k)$; если $L(\bar{u}, X^k) \geqslant \hat{z}$, задачу (6.9) можно исключить, так как $L(\bar{u}, X^k) \leqslant v(X^k)$.

Шаги 6—8. Пусть $\tilde{x} \in X^k$ — оптимальное решение (6.11), и пусть $\tilde{x}$ допустимо, т. е. $g(\tilde{x}) \leqslant b$. Поскольку задача (6.9) не была исключена (шаг 5), имеем

$$L(\bar{u}, X^k) = f(\tilde{x}) + \bar{u}(g(\tilde{x}) - b) < \hat{z},$$

причем $\bar{u}(g(\tilde{x}) - b) \leqslant 0$. Следовательно, если $f(\tilde{x}) < \hat{z}$, то рекорд $\hat{x}$ должен быть заменен на $\tilde{x}$. Во всяком случае, если $\tilde{x}$ допустимо, из теории двойственности следует, что

$$f(\tilde{x}) + \bar{u}(g(\tilde{x}) - b) \leqslant v(X^k) \leqslant f(\tilde{x})$$

и поэтому $\tilde{x}$ оптимально в задаче (6.9), если $\bar{u}(g(\tilde{x}) - b) = 0$, т. е. выполняется условие нежесткости.

Шаг 9. Это может быть тест на оптимальность решения двойственной задачи при текущем $\bar{u}$ или же тест текущего улучшения нижней границы. Если для решения двойственной задачи используется обобщенное линейное программирование, то оно дает на каждой итерации верхнюю границу $\bar{d}$ величины $d(X^k)$. В этом случае, если $\bar{d} < \hat{z}$, то подзадача (X^k) не может быть исключена.

Шаг 10. Выбор нового значения $\hat{u} \geqslant 0$ зависит от методов решения двойственной задачи.

Шаги 11 и 12. Ветвление данной подзадачи (6.9) может быть осуществлено на основе информации о двойственной задаче.

Отметим, что наибольшая нижняя граница, полученная в результате решения двойственной задачи (6.10), остается нижней границей для подзадач-потомков, т. е. для задач l таких, что $X^l \subseteq X^k$. Еще одна возможность связана с использованием так называемых суррогатных (замещающих) ограничений, имеющих вид [27]

$$f(x) + u(g(x) - b) < \hat{z}, \qquad u \geqslant 0. \tag{6.12}$$

Они могут быть введены в задачу (6.9), поскольку каждое ее допустимое решение со значением целевой функции, меньшим чем $\hat{z}$, удовлетворяет (6.12). Если u оптимально или близко к оптимальному решению двойственной задачи в форме (6.10), ограничение (6.12) существенно влияет на анализ подзадач-потомков задачи (6.9).

6.3. Двойственные оценки в задачах на ациклических сетях операций

В этом разделе методы теории двойственности в оценочных задачах будут распространены на сетевую задачу распределения ресурсов общего вида, в которой отношение предшествования операций представлено множеством дуг ациклического графа. Такая задача относится к задачам типа (6.3), и для нее могут быть выписаны лагранжиан (6.4), двойственная негладкая задача (6.5) и внутренняя задача минимизации в (6.4). Особенность данного случая в том, что внутреннюю задачу (6.4) удается свести к широко известной задаче о максимальном потоке в сети, для решения которой имеются алгоритмы с полиномиальной оценкой (эффективные алгоритмы). Тем самым оказывается, что и метод решения оценочной задачи в данном случае эффективен.

Рассмотрим следующую задачу. Имеется конечный ориентированный граф (V, E) без петель и контуров, $V = \{1, 2, \ldots, i, \ldots, n\}$, $E \subset V \times V$, с вершинами-операциями. Каждая операция описывается парой (d_i, r_i), $r_i = \{r_i^1, \ldots, r_i^k, \ldots, r_i^K\}$, где d_i — длительность операции i; r_i^k — уровень потребления ресурсов k-го вида в процессе выполнения операции i. Длительности операций — целые числа. Независимая переменная t предполагается целочисленной и изменяется в интервале планирования $[0, T]$. Задан также вектор

$$r(t) = (r^1(t), \ldots, r^k(t), \ldots, r^K(t)),$$

где $r^k(t)$ — кусочно-постоянные, непрерывные справа функции времени, описывающие общие уровни наличия ресурсов во времени. Выполнение операции состоит в предоставлении ей необходимого количества ресурсов каждого вида на время d_i, $i = 1, \ldots, n$. Операции предполагаются неделимыми, т. е. если операция i началась в момент времени t_i, то время ее завершения равно $t_i + d_i$. Расписанием называется вектор

$$S = (t_1(S), \ldots, t_i(S), \ldots, t_n(S)),$$

где $t_i(S)$ — время начала операции i, $i = 1, \ldots, n$. Пусть $V_t(S)$ — множество операций, выполняющихся в момент времени t в соответствии с расписанием S:

$$V_t(S) = \{i \mid t_i(S) \leqslant t \leqslant t_i(S) + d_i\};$$

$r_t^k(S)$ — общий уровень потребления ресурсов k-го вида

в момент времени t:

$$r_t^k(S) = \sum_{i \in V_t(S)} r_i^k, \quad k = 1, \ldots, K;$$

$f_0(S)$ — целевая функция; предполагается, что

$$f_0(S) = \sum_{i=1}^{n} f_i(t_i(S)),$$

где $f_i(t)$ — условные потери, связанные с началом операции i в момент времени t, $i = 1, \ldots, n$.

Задача составления оптимального расписания имеет вид: найти

$$\min f_0(S)$$

при условиях

$$r_t^k(S) \leqslant r^k(t), \quad t = 1, \ldots, T,$$
$$t_i(S) + d_i \leqslant t_j(S), \quad (i, j) \in E,$$
$$t_i(S) \in \{1, \ldots, T\}, \quad i = 1, \ldots, n.$$

Представим ее в форме задачи целочисленного линейного программирования. Введем переменные x_{it}, $i = 1, \ldots, n$; $t = 1, \ldots, T$, такие, что $x_{it} = 1$, если в искомом расписании операция i начинается не позднее t, иначе $x_{it} = 0$. Полагаем $x_{it} = 0$, $t < \underline{t}_i$, и $x_{it} = 1$, $t \geqslant \bar{t}_i$, где $\underline{t}_i$, $\bar{t}_i$ — соответственно наиболее ранний и наиболее поздний сроки начала операции i при условии, что все операции начинаются не ранее $t = 1$ и не позднее $(T - d_i)$. Для вычисления $\underline{t}_i$ и $\bar{t}_i$ могут быть использованы известные методы сетевого планирования. Обозначим $c_{it} = f_i(t)$, $i = 1, \ldots, n$. Задача составления оптимального расписания преобразуется к виду: найти

$$\min \sum_{i=1}^{n} \sum_{t=1}^{T} c_{it} (x_{it} - x_{i(t-1)}) \tag{6.13}$$

при условиях

$$\sum_{i=1}^{n} \sum_{t=1}^{T} r_i^k \left(x_{it} - x_{i(t-d_i)}\right) \leqslant r^k(t), \quad k = 1, \ldots, K, \tag{6.14}$$

$$x_{it} \leqslant x_{i(t+1)}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, (T-1), \tag{6.15}$$

$$x_{jt} \leqslant x_{i(t-d_i)}, \quad (i, j) \in E, \quad t = 1, \ldots, T, \tag{6.16}$$

$$x_{it} = 0 \vee 1, \quad i = 1, \ldots, n, \quad t = 1, \ldots, T, \quad x_{it} = 0, \quad t \leqslant 0. \tag{6.17}$$

Определим выражение для функции Лагранжа. Применительно к задаче (6.13)—(6.17) функция Лагранжа имеет вид

$$L(x, u) = \sum_{i=1}^{n} \sum_{t=1}^{T} c_{it}(x_{it} - x_{i(t-1)}) + \\ + \sum_{k=1}^{K} \sum_{t=1}^{T} u_{kt} \left[\sum_{i=1}^{n} r_i^k \left(x_{it} - x_{i(t-d_i)}\right) - r^k(t) \right] = \\ = \sum_{i=1}^{n} \sum_{t=1}^{T-1} c'_{it}(u)\, x_{it} - \sum_{k=1}^{K} \sum_{t=1}^{T} u_{kt} r^k(t),$$

где

$$c'_{it}(u) = c_{it} - c_{i(t+1)} + \sum_{k=1}^{K} r_i^k \left(u_{kt} - u_{k(t+d_i)}\right),$$

а $u_{k\tau} = 0$ при $\tau > T$. Множество X определяется условиями (6.15)—(6.17). Внутренняя задача оптимизации в (6.4) определяется так:

$$\Phi(u) = \left\{ \min \sum_{i=1}^{n} \sum_{t=1}^{T-1} c'_{it}(u)\, x_{it} \mid x \in X \right\}. \qquad (6.18)$$

Сопоставим задаче (6.18) граф (Y, U) следующим образом. Множество вершин графа взаимно однозначно соответствует множеству переменных задачи. Дуга $(x_{it}, x_{j\tau})$ принадлежит U, если и только если среди ограничений (6.15), (6.16) найдется ограничение $x_{it} \geqslant x_{j\tau}$. Очевидно, (Y, U) также является ориентированным графом без петель и контуров. Структура допустимого решения задачи (6.18) имеет наглядную геометрическую интерпретацию на графе (Y, U). А именно, $x_{j\tau} = 1$ влечет $x_{it} = 1$ для всех x_{it}, для которых в графе (Y, U) найдется путь $x_{it} \to x_{j\tau}$. Обратное, $x_{it} = 0$ влечет $x_{j\tau} = 0$ при том же условии. Заметим, что, поскольку матрица ограничений задачи (6.18) вполне унимодулярна, условие целочисленности в ограничениях (6.17) можно опустить. После очевидной замены переменных задача (6.18) сводится к виду: найти

$$\min \sum_{p=1}^{|Y|} c_p z_p \qquad (6.19)$$

при условиях

$$z_p - z_q \geqslant 0, \quad (p, q) \in U, \qquad (6.20)$$

$$-z_p \geqslant -1, \quad p = 1, \ldots, |Y|, \qquad (6.21)$$

$$z_p \geqslant 0, \quad p = 1, \ldots, |Y|. \qquad (6.22)$$

Переменные задачи (6.19)—(6.22) по-прежнему соответствуют вершинам графа (Y, U), а ограничения (6.20) — его дугам.

Введем в задачу (6.19)—(6.22) фиктивную переменную $z_0 \geqslant 0$, положим $c_0 = 0$, а ограничения (6.21) запишем в виде

$$z_0 - z_p \geqslant -1, \qquad p = 1, \ldots, |Y|.$$

Новую задачу обозначим (6.19′)—(6.22′). Ей соответствует расширенный граф (Y', U'), в котором

$$Y' = Y \cup 0, \qquad U' = U \cup \{(0, y),\ y \in Y\}.$$

Рассмотрим задачу, двойственную к (6.19′)—(6.22′): найти

$$\max\left(-\sum_{p \in Y} w_{0p}\right) \tag{6.23}$$

при условиях

$$\sum_q w_{pq} - \sum_q w_{qp} \leqslant c_p, \qquad p \in Y', \tag{6.24}$$

$$w_{pq} \geqslant 0, \qquad (p, q) \in U'. \tag{6.25}$$

Здесь переменные w_{0p} соответствуют ограничениям (6.21′) прямой задачи, а переменные w_{pq} — ограничениям (6.20′). Задача (6.23)—(6.25) является специальной задачей о спросе и предложении на сети (Y, U), причем верхние границы на пропускные способности дуг отсутствуют. Как известно, задачи о спросе и предложении сводятся к задачам о максимальном потоке на соответствующим образом расширенной сети [24]. Выделим в Y две группы вершин:

$$Y_1 = \{p \mid c_p > 0\}, \qquad Y_2 = \{q \mid c_q < 0\}$$

и дополним сеть (Y, U) парой вершин (s, t), дугами (s, p), $p \in Y_1$, с пропускными способностями c_p и дугами (q, t), $q \in Y_2$, с пропускными способностями $|c_q|$. В расширенной сети $|Y', U'|$ найдем максимальный поток из s в t. Эквивалентная задача о максимальном потоке имеет вид: найти

$$\max \sum_{p \in Y_1} w_{sp} \tag{6.26}$$

при условиях

$$\sum_q w_{pq} - \sum_q w_{qp} = 0, \qquad p \in Y, \tag{6.27}$$

$$0 \leqslant w_{sp} \leqslant c_p, \qquad 0 \leqslant w_{qt} \leqslant |c_q|, \qquad p \in Y_1,\ q \in Y_2, \tag{6.28}$$

$$w_{pq} \geqslant 0, \qquad (p, q) \in U'. \tag{6.29}$$

Пусть $(Y^*, \overline{Y}^*)$ — минимальный разрез в сети (Y', U'). Тогда в оптимальном решении задачи (6.19)—(6.22), двойственной к задаче (6.26)—(6.29),

$$z_p = 0, \quad p \in Y^*, \quad \text{и} \quad z_q = 1, \quad q \in \overline{Y}^*.$$

Соответственно, в эквивалентной задаче (6.18) имеем

$$x^*_{it} = 0, \quad (i, t) \in (Y^* \setminus s),$$

и

$$x^*_{i\tau} = 1, \quad (i, \tau) \in (\overline{Y}^* \setminus t).$$

Обратимся к двойственной задаче (6.5). Как отмечалось в разделе 6.2, для ее решения могут быть использованы методы субградиентной оптимизации. Так, обобщенный градиентный спуск в (6.5) представляется как процесс:

$$u_{k+1} = Q_+ \left[u_k + h_{k+1} \frac{g_\Phi(u_k)}{\| g_\Phi(u_k) \|} \right], \quad k = 0, 1, 2, \ldots, \quad (6.30)$$

где $Q_+[\cdot]$ означает операцию проектирования выражения в скобках на конус $u \geqslant 0$; $u_0 = 0$; $h_0 > 0$ и $h_k = h_0 q^k$, причем $0 \leqslant q < 1$; $g_\Phi(u)$ — обобщенный градиент функции $\Phi(u)$ в точке u, определяемый в нашем случае как вектор размерности KT, kt-я компонента которого равна

$$g^{kt}_\Phi(u) = \sum_i \sum_t r^k_i \left(x^*_{it}(u) - x^*_{i(t-d_i)}(u) - r^k(t), \right.$$

$x^*(u)$ — оптимальное решение задачи (6.18) при фиксированном u.

Таким образом, вычисление оценок снизу для (6.13) в задаче (6.13)—(6.17) сводится к процессу (6.30), на каждом шаге которого решается задача о максимальном потоке (6.26)—(6.29); минимальный разрез задачи (6.26)—(6.29) используется для вычисления обобщенного градиента. Альтернативным способом вычисления искомых оценок является непосредственное решение задачи линейного программирования (6.13)—(6.17).

6.3.1. Иллюстративный пример. Рассмотрим сеть операций (V, E), изображенную на рис. 6.2. Длительности операций таковы: $d_i = 1$, $i = 1, \ldots, 6$. Подлежит распределению один вид ресурса: $K = 1$. Уровни потребления ресурса: $r_i = 1$, $i = 1, \ldots, 6$. Интервал планирования: $T = 4$. Общий уровень наличия ресурса: $r(t) = 2$, $t = 1, \ldots, 4$. Цель планирования: минимизировать $\sum_{i=1}^{6} t_i$.

Легко проверить, что оптимальное расписание, например, следующее:

$$i\colon\ 1,\ 2,\ 3,\ 4,\ 5,\ 6;$$

$$t_i\colon\ 1,\ 2,\ 1,\ 3,\ 2,\ 3;$$

$$\min f_0 = 12.$$

Найдем двойственную оценку для $\min f_0$. Задача (6.18)

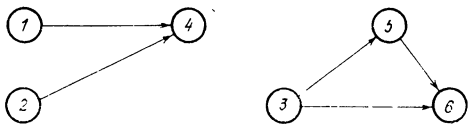

Рис. 6.2.

имеет вид: минимизировать

$$\sum_{i=1}^{6}\sum_{t=1}^{3}(u_t - u_{t+1} - 1)\,x_{it}$$

при условиях

$$x_{11} \leqslant x_{12}, \qquad x_{22} \geqslant x_{43},$$
$$x_{11} \geqslant x_{42}, \qquad x_{31} \geqslant x_{52},$$
$$x_{12} \geqslant x_{43}, \qquad x_{31} \geqslant x_{63},$$
$$x_{21} \leqslant x_{22}, \qquad x_{42} \leqslant x_{43},$$
$$x_{21} \geqslant x_{42}, \qquad x_{52} \geqslant x_{63},$$

$$0 \leqslant x_{it} \leqslant 1, \quad i = 1, \ldots, 6; \quad t = 1, \ldots, 4.$$

Сеть (X, U) изображена на рис. 6.3; в вершинах указаны номера пар it. Данные итерационного процесса (6.29) сведены в табл. 6.1. Принято $h_0 = 16$, $q = 0{,}5$. В послед-

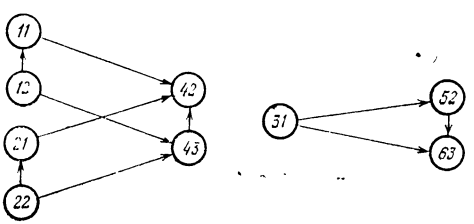

Рис. 6.3.

нем столбце табл. 6.1 приводится значение оценки. Как видно, на третьем шаге достигается $\Phi(u) = \min f_0 = 12$,

Таблица 6.1

	$u(t)$					it									$g_{\Phi(u)}$				$\Phi(u)$
t	1	2	3	4		11	12	21	22	31	42	43	52	63	1	2	3	4	
$k=0$	0	0	0	0	c_{it}	−1	−1	−1	−1	−1	−1	−1	−1	−1	1	0	−1	−2	19
					x_{it}	1	1	1	1	1	1	1	1	1					
$k=1$	6,5	0	0	0	c_{it}	5,5	−1	5,5	−1	5,5	−1	−1	−1	−1	−2	1	0	−1	13
					x_{it}	0	1	0	1	0	0	1	0	0					
$k=2$	0	3,27	0	0	c_{it}	−4,27	2,27	−4,27	2,27	−4,27	2,27	−1	2,27	−1	1	−2	0	−1	16
					x_{it}	1	1	1	1	1	0	1	0	0					
$k=3$	1,63	0	0	0	c_{it}	0,63	−1	0,63	−1	0,63	−1	−1	−1	−1	0	0	0	−2	12
					x_{it}	1	1	0	1	1	0	1	1	1					

т. е. в процессе вычисления оценки получено оптимальное решение задачи составления оптимального расписания.

6.4. Специальные методы решения оценочных задач

6.4.1. Две важные специальные задачи. Как описано в главе I, задачи оптимального распределения ограниченных ресурсов и составления расписаний в общих чертах заключаются в следующем. Задана система операций. Выполнение операций связано с использованием определенных средств (ресурсов). Как правило, оказывается, что общего уровня наличия ресурсов недостаточно для одновременного выполнения всех операций. Требуется упорядочить процесс выполнения операций во времени таким образом, чтобы при соблюдении ряда дополнительных условий минимизировать заданную функцию потерь, определенную на сроках начала или завершения операций. Наиболее часто речь идет о двух родственных задачах. Эта задача минимизации времени завершения всех операций при заданных уровнях наличия ресурсов (задача А) и задача минимизации максимального по времени уровня потребления ресурсов при заданном сроке завершения операций (задача В). Сформулируем их. Предполагается, что задан ориентированный граф (V, E) без петель и контуров,

$$V = \{v_1, \ldots, v_n\}, \quad E \subset V \times V,$$

с вершинами-операциями. Каждая операция описывается парой целых неотрицательных чисел (τ_i, r_i), где τ_i — длительность операции v_i; r_i — уровень потребления ресурсов. Кроме того, задана кусочно-постоянная, непрерывная справа функция времени $r(t)$, описывающая общие уровни наличия ресурсов. Операции предполагаются непрерывными. Расписанием называется вектор

$$S = (t_1(S), \ldots, t_i(S), \ldots, t_n(S)),$$

где $t_i(S)$ — срок завершения операции v_i в расписании S. Все операции начинаются не ранее $t = 0$.

Пусть $V_t(S)$ — множество операций, выполняемых в момент времени t:

$$V_t(S) = \{v_i \in V \mid t_i(S) - \tau_i \leqslant t \leqslant t_i(S)\};$$

$r_t(S)$ — общий уровень потребления ресурсов в момент времени t:

$$r_t(S) = \sum_{\{i|v_i \in V_t(S)\}} r_i.$$

Задача А. Минимизировать функцию

$$\max_i t_i(S) \tag{6.31}$$

при условиях

$$r_t(S) \leqslant r(t),$$
$$t_i(S) \leqslant t_j(S) - \tau_j, \quad (v_i, v_j) \in E,$$
$$t_i(S) \geqslant 0, \quad i = 1, \ldots, n.$$

Пусть теперь задан срок завершения всех операций T.

Задача В. Минимизировать функцию

$$\max_{0 \leqslant t \leqslant T} r_t(S) \tag{6.32}$$

при условиях

$$0 \leqslant t_i(S) \leqslant T,$$

$$t_i(S) \leqslant t_j(S) - \tau_j, \quad (v_i, v_j) \in E.$$

Обе задачи являются весьма сложными в вычислительном отношении комбинаторными проблемами, и для их решения может быть использован метод ветвей и границ. В алгоритмах ветвей и границ необходимо решать оценочные задачи. В принципе задачи А и В могут быть представлены как задачи целочисленного линейного программирования, и в этом случае в качестве оценочной задачи можно рассматривать соответствующую задачу линейного программирования (без условий целочисленности). Альтернативой являются двойственные оценки, описанные в разделе 6.2. Ниже будет рассмотрен еще один путь для решения оценочных задач в задачах А и В. Он основан на использовании структурных свойств задач и специфики функционалов (6.31) и (6.32). Средством для описания специальных методов решения оценочных задач являются понятия интегральных графиков потребностей ресурсов и интервалов концентрации (потребностей ресурсов).

6.4.2. Оценки на основе интегральных графиков потребностей ресурсов. Пусть заданы интервал планирования $[t_0, T_0]$, сеть операций (V, E) и константа m — об-

щий уровень наличия ресурсов на отрезке $[t_0, T_0]$. Как показано в главе II, каждую задачу оптимизации можно представить в виде некоторой задачи существования. Пусть для сети (V, E) существует расписание с длительностью $(T_0 - t_0)$ и максимальным общим уровнем потребления ресурсов r_0; тогда в задаче А существует расписание с длительностью, не превосходящей $(T_0 - t_0)$, а в задаче В — расписание с максимальным общим уровнем, не превосходящим r_0.

Пусть t_i — наиболее ранний, а $\bar{t}_i$ — наиболее поздний сроки начала операции v_i. Величины $\underline{t}_i$, $\bar{t}_i$, $i = 1, \ldots, n$, определяются при условиях, что все операции сети (V, E) начинаются не ранее t_0 и завершаются не позднее T_0. Пусть также τ_0 обозначает длительность критического пути в сети (V, E), т. е. наибольшего пути вида $v_i \to v_j$, где v_i — мажоранта, а v_j — миноранта частично упорядоченного множества, порождаемого графом (V, E). Далее предполагается, что $t_0 \geqslant 0$, $T_0 - t_0 \geqslant \tau_0$.

Определим ряд терминов, в которых выражаются условия существования расписаний в интервале времени $[t_0, T_0]$. Графиками потребностей операции i в ресурсах соответственно по ранним и по поздним срокам называются следующие функции переменной t:

$$r_i(t \mid t_0) = \begin{cases} r_i, & \underline{t}_i \leqslant t \leqslant \underline{t}_i + \tau_i, \\ 0 & \text{в остальных случаях;} \end{cases}$$

$$r_i(t \mid T_0) = \begin{cases} r_i, & \bar{t}_i \leqslant t \leqslant \bar{t}_i + \tau_i, \\ 0 & \text{в остальных случаях.} \end{cases}$$

Графиками потребностей в ресурсах по ранним и поздним срокам называются соответственно функции

$$r(t \mid t_0) = \sum_{i=1}^{n} r_i(t \mid t_0), \quad r(t \mid T_0) = \sum_{i=1}^{n} r_i(t \mid T_0).$$

Величина $m_1 = \max\limits_{t_0 \leqslant t \leqslant T_0} r(t \mid t_0)$, $m_2 = \max\limits_{t_0 \leqslant t \leqslant T_0} r(t \mid T_0)$ называется шириной расписания по ранним (поздним) срокам. Интегральными графиками потребления ресурсов по ранним и поздним срокам называются функции

$$R(t \mid t_0) = \sum_{\tau=t_0}^{t} r(\tau \mid t_0), \quad R(t \mid T_0) = \sum_{\tau=t_0}^{t} r(\tau \mid T_0).$$

Как и ранее, параметр m определяет общий уровень наличия ресурсов в интервале планирования $[t_0, T_0]$.

Сглаженными (или m-сглаженными) интегральными графиками потребления ресурсов по ранним и поздним срокам называются функции

$$R_m(t \mid t_0) = \\ = \min\{R(t \mid t_0), \min_{t_0 \leqslant \tau \leqslant t} (R(\tau \mid t_0) + m(t-\tau))\}, \quad t \geqslant t_0,$$

и

$$R_m(t \mid T_0) = \\ = \max\{R(t \mid T_0), \max_{t \leqslant \tau \leqslant T_0} (R(\tau \mid T_0) - m(\tau - t))\}, \quad t \leqslant T_0.$$

Отметим некоторые свойства интегральных графиков, непосредственно следующие из их определений.

1. $R_m(t|t_0) \leqslant R(t|t_0) = R_{m_1}(t|t_0)$, $R_m(t|T_0) \geqslant R(t|T_0) = R_{m_2}(t|T_0)$. Вообще, из $m \leqslant M$ следует, что

$$R_m(t|t_0) \leqslant R_M(t|t_0),$$
$$R_m(t|T_0) \geqslant R_M(t|T_0).$$

2. Сглаженные интегральные графики $R_m(t|t_0)$ и $R_m(t|T_0)$, равно как и несглаженные интегральные графики $R(t|t_0)$ и $R(t|T_0)$, являются неубывающими кусочно-линейными функциями переменной t. Кроме того,

$$R(t \mid t_0) = \begin{cases} 0, & t \leqslant t_0, \\ R_0, & t \geqslant t_0 + \tau_0; \end{cases} \quad R(t \mid T_0) = \begin{cases} 0, & t \leqslant T_0 - \tau_0, \\ R_0, & t \geqslant T_0; \end{cases}$$

$$R_m(t \mid t_0) = \begin{cases} 0, & t < t_0, \\ R_0, & t \geqslant b; \end{cases} \quad R_m(t \mid T_0) = \begin{cases} 0, & t \leqslant a, \\ R_0, & t \geqslant T_0, \end{cases}$$

где использованы следующие обозначения:

$$R_0 = \sum_{i=1}^{n} r_i \tau_i,$$
$$a = \max\{t \mid R_m(t \mid T_0) = 0\},$$
$$b = \min\{t \mid R_m(t \mid t_0) = R_0\}.$$

3. Пусть задан сглаженный интегральный график по ранним срокам $R_m(t|t_0)$. На временной оси найдутся такие точки t_λ, $\lambda = 1, 2, \ldots$, что внутри каждого отрезка $[t_\lambda, t_{\lambda+1}]$ имеет место один из двух случаев А1, В1:

А1. $R_m(t|t_0) = R(t|t_0)$.

В1. $R_m(t|t_0) < R(t|t_0)$, $R_m(t|t_0) = R(t_\lambda|t_0) + m(t - t_\lambda)$.

Для сглаженного интегрального графика по поздним срокам выполняются аналогичные свойства:

А2. $R_m(t|T_0) = R(t|T_0)$.

В2. $R_m(t|T_0) > R(t|T_0)$, $R_m(t|T_0) = R(t_\lambda|T_0) + m(t - t_\lambda)$. В случаях А1 и А2 будем говорить, что имеем отрезки первого рода, а в случаях В1 и В2 — отрезки второго рода. Отметим, что отрезки второго рода сглаженных интегральных графиков являются отрезками прямой в плоскости (t, R) и эта прямая наклонена к оси t под углом $\operatorname{arctg} m$.

4. Пусть точка (t^1, R^1) принадлежит отрезку второго рода графика $R_m(t|t_0)$ или $R_m(t|T_0)$; $R = mt + c$ — прямая в плоскости (t, R), содержащая данный отрезок. Тогда

$$mt + c \leqslant R_m(t|t_0), \quad t \leqslant t^1,$$
$$mt + c \geqslant R_m(t|T_0), \quad t \geqslant t^1.$$

Определенные выше графики потребностей в ресурсах неявно зависят от сроков (ранних или поздних) выполнения операций. Для заданного расписания $S = (t_1(S), t_2(S), \ldots, t_n(S))$ также могут быть определены соответствующие графики. А именно, свяжем с S графики потребностей операций в ресурсах:

$$r_i(t|S) = \begin{cases} r_i, & t_i(S) \leqslant t \leqslant t_i(S) + \tau_i, \\ 0 & \text{в противном случае,} \end{cases} \quad i = 1, \ldots, n,$$

график потребностей в ресурсах:

$$r(t|S) = \sum_{i=1}^{n} r_i(t|S)$$

и интегральный график потребностей в ресурсах:

$$R(t|S) = \int_{t_0}^{t} r(\tau|S)\, d\tau.$$

Заметим, что функция $R(t|S)$ также является неубывающей кусочно-линейной функцией времени (переменной t).

Как следует из определений, сглаженный интегральный график по ранним срокам указывает для каждого момента времени верхние границы объемов работ (ресурсозатрат) в расписаниях при условии, что выполнение совокупности всех операций сети начинается не ранее t_0. Аналогично, сглаженный интегральный график по поздним срокам указывает для каждого момента времени нижние границы объемов работ в расписаниях при условии, что выполнение совокупности всех операций сети завершается не позднее T_0. Для практического использования интегральных графиков и сглаженных интеграль-

ных графиков удобно переопределить их в терминах, обратных по отношению к ним функций. Пусть переменная R принадлежит $[0, R_0]$, а соответствующие обратные функции

$$T(R|t_0), \quad T(R|T_0), \quad T_m(R|t_0), \quad T_m(R|T_0)$$

определим следующим образом:

$$T(R\,|\,t_0) = \begin{cases} 0, & R = 0, \\ \max\{t\,|\,R(t\,|\,t_0) = R\}, & 0 < R < R_0, \\ t_0 + \tau_0, & R = R_0; \end{cases}$$

$$T(R\,|\,T_0) = \begin{cases} 0, & R = 0, \\ \min\{t\,|\,R(t\,|\,t_0) = R\}, & 0 < R < R_0, \\ T_0 - \tau_0, & R = R_0; \end{cases}$$

$$T_m(R\,|\,t_0) = \begin{cases} 0, & R = 0, \\ \max\{t\,|\,R_m(t\,|\,t_0) = R\}, & 0 < R < R_0, \\ b, & R = R_0; \end{cases}$$

$$T_m(R\,|\,T_0) = \begin{cases} 0, & R = 0, \\ \min\{t\,|\,R_m(t\,|\,T_0) = R\}, & 0 < R < R_0, \\ a, & R = R_0. \end{cases}$$

Посредством интегральных графиков и соответствующих им обратных функций могут быть определены необходимые условия существования расписаний с заданными свойствами, из которых следуют оценки функционалов в задачах A и B. Пусть заданы интервал планирования $[t_0, T_0]$, общий уровень наличия ресурсов во времени $m(t) = \text{const} = m$, расписание S. Переменная $T_{\min}$ обозначает длительность кратчайшего расписания, а $\underline{T}^{(\ldots)}_{\min}$ — нижнюю оценку для нее. Справедливы следующие утверждения.

Лемма 6.1. *Пусть в расписании S все операции начинаются не ранее t_0. Тогда*

$$R(t|S) \leqslant R_m(t|t_0), \qquad t_0 \leqslant t \leqslant b.$$

Следствие 6.3.

$$T_{\min} \geqslant \underline{T}^{(1)}_{\min} = b - t_0.$$

Лемма 6.2. *Пусть в расписании S все операции завершаются не позднее T_0. Тогда*

$$R_m(t|T_0) \leqslant R(t|S), \qquad a \leqslant t \leqslant T_0.$$

Следствие 6.4.

$$T_{\min} \geqslant \underline{T}^{(2)}_{\min} = T_0 - a.$$

Лемма 6.3. *Пусть на отрезке* $[t_0, T_0]$ *существует некоторое расписание. Тогда*

$$R_m(t|T_0) \leqslant R_m(t|t_0), \quad a \leqslant t \leqslant b. \tag{6.33}$$

Из (6.33) и свойства 1 следует, что для несглаженных интегральных графиков выполняется аналогичное условие

$$R(t|T_0) \leqslant R(t|t_0). \tag{6.34}$$

В терминах обратных функций необходимые условия существования расписаний (6.33) и (6.34) принимают вид

$$T_m(R|t_0) \leqslant T_m(R|T_0), \quad T(R|t_0) \leqslant T(R|T_0)$$

или

$$\delta_m(t_0, T_0) \leqslant 0, \quad \delta(t_0, T_0) \leqslant 0, \tag{6.35}$$

где обозначено

$$\delta_m(t_0, T_0) = \max_{0 \leqslant R \leqslant R_0} (T_m(R|t_0) - T_m(R|T_0)),$$

$$\delta(t_0, T_0) = \max_{0 \leqslant R \leqslant R_0} (T(R|t_0) - T(R|T_0)).$$

Следствие 6.5.

$$T_{\min} \geqslant \underline{T}^{(3)}_{\min} = T_0 - t_0 - \delta_m(t_0, T_0). \tag{6.36}$$

Лемма 6.4.

$$\underline{T}^{(1)}_{\min} \leqslant \underline{T}^{(3)}_{\min}, \quad \underline{T}^{(2)}_{\min} \leqslant \underline{T}^{(3)}_{\min}.$$

Доказательство. Из (6.33) следует аналогичное неравенство и для обратных функций:

$$T_m(R|t_0) \leqslant T_m(R|T_0).$$

Так как $\delta = \delta(t_0, T_0) \leqslant 0$, последнее неравенство остается справедливым, если заменить в нем T_0 на $T_0' = T_0 - \delta$, т. е. имеет место $T_m(R|t_0) \leqslant T_m(R|T_0')$ для всех R, $0 \leqslant R \leqslant R_0$. В частности, для $R = 0$ и $R = R_0$ выполнено

$$t_0 = T_m(0|t_0) \leqslant T_m(0|T_0') = a - \delta,$$

$$b \leqslant T_m(R_0|t_0) \leqslant T_m(0|T_0') = T_0 - \delta.$$

Отсюда получаем

$$\underline{T}^{(3)}_{\min} - \underline{T}^{(1)}_{\min} \leqslant T_0 - \delta - b \geqslant 0,$$

$$\underline{T}^{(3)}_{\min} - \underline{T}^{(2)}_{\min} \leqslant a - \delta - t_0 \geqslant 0,$$

что и требовалось доказать.

В лемме 6.4 установлено, что оценка $\underline{T}^{(3)}_{\min}$ не слабее $\underline{T}^{(1)}_{\min}$ и $\underline{T}^{(2)}_{\min}$. Следующий пример показывает, что разница этих оценок может строго возрастать с ростом числа операций сети. Сеть (рис. 6.4, *а*) состоит из $n(n+2)$ операций единичной длительности. При этом $\tau_i = 1$, $\tau_0 = 2n+1$, $m = n$, $R_0 = n^2 + 2n$. На рис. 6.4, *б* представлены интегральные графики по ранним срокам (ломаная

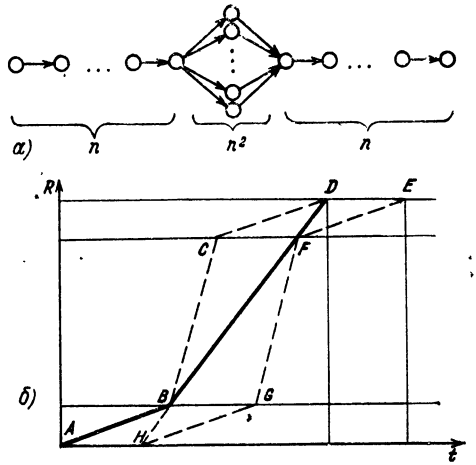

Рис. 6.4.

ABCD) и по поздним срокам (ломаная *HGFE*). Соответствующие сглаженные интегральные графики изображены ломаными *ABD* и *HBFE*. В данном случае имеем

$$\underline{T}^{(1)}_{\min} = \underline{T}^{(2)}_{\min} = 2n+1, \quad T^{(3)}_{\min} = 3n,$$

$$\delta T_{\min} = \underline{T}^{(3)}_{\min} - \underline{T}^{(1)}_{\min} = n-1,$$

т. е. разность оценок пропорциональна n.

Таким образом, вычисление оценок длительности кратчайших расписаний в задаче А сводится к определению расстояния между парой сглаженных интегральных графиков (по ранним и поздним срокам), т. е. величины $\delta_m(t_0, T_0)$, и использованию выражения (6.36) для $\underline{T}^{(3)}_{\min}$. Покажем теперь, каким образом необходимые условия существования расписаний используются для вычисления оценки в задаче 2 — нижней оценки максимального уровня использования ресурсов при условии, что сроки выполнения операций ограничены снизу и сверху.

Заметим, что при заданных t_0 и T_0 расстояние (6.35) между парой сглаженных интегральных графиков $T_m(R|t_0)$ и $T_m(R|T_0)$ является функцией параметра m. В таком случае задача оценки состоит в определении максимального значения m, при котором имеет место

$$\delta_m(t_0,\ T_0) = 0. \tag{6.37}$$

Искомое значение m обозначим $\underline{m}$. Перейдем к решению этой вспомогательной задачи. Поскольку сглаженные интегральные графики по определению сами зависят от m, прямой способ получения $\underline{m}$ мог бы заключаться в варьировании значения m (при фиксированных t_0 и T_0) в некотором интервале с целью решения уравнения (6.37). Покажем, однако, что существует более простой способ решения этой вспомогательной задачи на основе несглаженных интегральных графиков.

Рассмотрим несглаженные интегральные графики $R(t|t_0)$ и $R(t|T_0)$. Как отмечалось, они являются неубывающими кусочно-линейными функциями t. Множество пар координат $\{(t_p,\ a_p),\ p = 1,\ 2,\ \ldots\}$ и $\{(t_q,\ a_q),\ q = 1,\ 2,\ \ldots\}$ обозначают точки излома графиков $R(t|t_0)$ и $R(t|T_0)$ в плоскости $(t,\ R)$.

Теорема 6.2. *Пусть для пары интегральных графиков необходимое условие* (6.34) *выполняется строго, т. е.* $\delta(t_0,\ T_0) < 0$, *а в* (6.35) *имеет место* $\delta_m(t_0,\ T_0) = 0$ *для некоторого* m. *Тогда*

$$\underline{m} = \max_{(p,q):\, t_p < t_q} \frac{a_q - a_p}{t_q - t_p}. \tag{6.38}$$

Доказательство. Пусть $\delta_m(t_0,\ T_0) = 0$. Это означает, что обратные функции $T_m(R|t_0)$ и $T_m(R|T_0)$ имеют по крайней мере одну общую точку. Пусть это будет $(\overline{T},\ \overline{R})$. Назовем ее точкой касания интегральных графиков $R_m(t|t_0)$ и $R(t|T_0)$. Убедимся в том, что $(\overline{T},\ \overline{R})$ принадлежит отрезкам второго рода этих графиков, т. е. в $(\overline{T},\ \overline{R})$ для $R(t|t_0)$ выполняется В1, а для $R(t|T_0)$ выполняется В2. Действительно, предположим обратное: $(\overline{T},\ \overline{R})$ принадлежит отрезкам первого рода. Но таковые по условиям А1 и А2 представляют собой отрезки несглаженных интегральных графиков $R(t|t_0)$ и $R(t|T_0)$, которые по условию леммы не пересекаются. Остается предположить, что $(\overline{T},\ \overline{R})$ принадлежит отрезкам второго рода. Последние по определению являются отрезками прямых в плоскости $(t,\ R)$ с одним и тем же углом наклона $\operatorname{arctg} m$.

Поскольку эти прямые имеют общую точку $(\overline{T}, \overline{R})$, то они совпадают. Итак, отрезки второго рода графиков $R(t|t_0)$ и $R(t|T_0)$, содержащие точку касания $(\overline{T}, \overline{R})$, лежат на одной и той же прямой $R = mt + c$ в плоскости (t, R). Рассмотрим, каким условиям должна удовлетворять эта прямая. Согласно свойствам 4 и 1 имеем

$$mt + c \leqslant R(t|t_0), \quad t < \overline{T}, \tag{6.39}$$

$$mt + c \geqslant R(t|T_0), \quad t > \overline{T}. \tag{6.40}$$

Далее, согласно определениям сглаженных интегральных графиков левые границы отрезков второго рода графиков $R_m(t|t_0)$ принадлежат $R(t|t_0)$, а правые границы отрезков второго рода графиков $R_m(t|T_0)$ принадлежат графикам $R(t|T_0)$. Это означает, что искомая прямая должна иметь общие точки с графиками $R(t|t_0)$ и $R(t|T_0)$. Назовем их точками касания прямой $R = mt + c$ с графиками $R(t|t_0)$ и $R(t|T_0)$. Поскольку $R(t|t_0)$ и $R(t|T_0)$ — кусочно-линейные функции, касание имеет место в точках излома. Пусть искомая прямая касается графиков $R(t|t_0)$ и $R(t|T_0)$ в точках (t_p, a_p) и (t_q, a_q). Тогда угол ее наклона равен

$$\operatorname{arctg} \frac{a_q - a_p}{t_q - t_p}.$$

Для соблюдения условий (6.39) и (6.40) выберем пару (p, q) с максимальным углом наклона. Теорема доказана.

Таким образом, для существования расписаний длины $(T_0 - t_0)$ и ширины m необходимо, чтобы число m удовлетворяло условию (6.38). Это необходимое условие можно использовать для оценивания снизу достаточных уровней ресурсов каждого вида при заданном ограничении на длину расписаний. Легко подобрать сеть, для которой оценка (6.38) совпадает с шириной расписания длины $(T_0 - t_0)$. Другими словами, оценки (6.38) являются точными.

В сети на рис. 6.5, *а* $\tau_i = 1$, $i = 1, \ldots, 4$. Потребности в ресурсах r_i^k записаны в таблице (рис. 6.5, *б*). В случае $r_0^1 = r_0^2 = 1$ длина кратчайшего расписания равна 3 (график Ганта на рис. 6.5, *в*). Пусть теперь $t_0 = 1$, $T_0 = 3$. Оценим величины r_0^k в соответствии с предлагаемой методикой. Отрезки AB, CD и $A'B'$, $C'D'$ (рис. 6.5, *г*, 6.5, *д*) изображают интегральные графики для $k = 1, 2$. Искомые пары точек таковы: AD и $A'D'$, что означает $m^1 = m^2 = 1$.

6.4.3. Оценки на основе интервалов концентрации потребностей ресурсов. Пусть для каждой операции v_i, $i = 1, \ldots, n$, величины $\underline{t}_i$ и $\overline{t}_i$ соответственно обозначают

наиболее ранний и наиболее поздний сроки начала операции при условии, что выполнение системы операций начинается не ранее $t_0 \geqslant 0$ и завершается не позднее $T_0 \geqslant \tau_0$, t_1 и t_2 — целые числа из $[t_0, T_0]$,

$$0 \leqslant t_1 < t_2 \leqslant T_0,$$

$$M_i(t_1, t_2) =$$

$$= r_i \min\{|[t_1, t_2] \cap [\underline{t_i}, \underline{t_i} + \tau_i]|, |[t_1, t_2] \cap [\bar{t_i}, \bar{t_i} + \tau_i]|\},$$

$$M(t_1, t_2) = \sum_{i=1}^{n} M_i(t_1, t_2).$$

Как следует из определений, функции $M_i(t_1, t_2)$ указывают необходимые объемы потребностей операций в ресурсах для каждого подынтервала $[t_1, t_2]$ из $[t_0, T_0]$, в то время как $M(t_1, t_2)$ — общие объемы потребностей в ресурсах

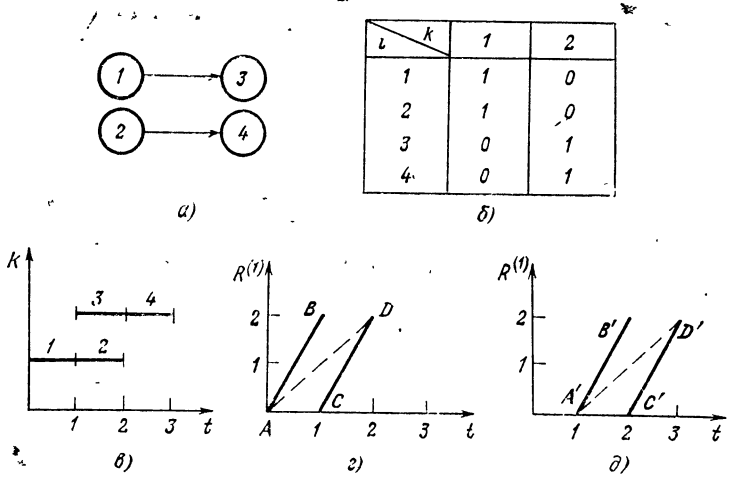

Рис. 6.5.

совокупности всех операций. Так же, как и интегральные графики, функции $M(t_1, t_2)$ могут быть использованы для решения оценочных задач в задачах оптимизации А и В. А именно, можно показать, что

$$\underline{T}^{(4)}_{\min} = \tau_0 + \max_{0 < t_1 < t_2 < T_0} \left[\frac{1}{m} M(t_1, t_2) - (t_2 - t_1)\right]$$

является нижней границей длительности кратчайшего

расписания, т. е. оценкой в задаче А, а

$$\underline{m}^{(4)} = \max_{0 \leqslant t_1 < t_2 \leqslant T_0} \frac{M(t_1, t_2)}{t_2 - t_1}$$

является нижней границей максимального уровня использования ресурсов, т. е. оценкой в задаче В.

Отрезок $[t_1, t_2]$ назовем интервалом концентрации, если $M(t_1, t_2) - m(t_2 - t_1) \geqslant 0$. Имеет место определенное соответствие между интегральными графиками и интервалами концентрации в заданном интервале планирования системы операций (V, E).

Лемма 6.5. *Пусть на плоскости* (t, R) *существует пара точек* (t_1, R_1) *и* (t_2, R_2) *таких, что выполняются условия*:

1) $(t_1, R_1) \in R(t|t_0)$, $(t_2, R_2) \in R(t|T_0)$, $t_1 < t_2$;

2) $R_2 - R_1 \geqslant m(t_2 - t_1)$.

Тогда $[t_1, t_2]$ — *интервал концентрации.*

Доказательство. Сопоставим каждой операции ранний или поздний срок ее начала t_i', $i = 1, \ldots, n$, таким образом, что совокупность чисел t_i' образует расписание $S' = \{t_i', i = 1, \ldots, n\}$. Пусть, как и ранее, множество $V_t(S')$ состоит из операций, выполняющихся в момент времени t, а $r'(t)$ — общие потребности в ресурсах операций из $V_t(S')$. Построим интегральный график использования ресурсов $R'(t)$ по правилу

$$R'(t) = \int_{t_0}^{t} r'(\tau)\, d\tau.$$

Из лемм 6.1 и 6.2 следует, что

$$R(t|t_0) \geqslant R'(t) \geqslant R(t|T_0).$$

В частности, имеем

$$R'(t_1) \leqslant R(t_1|t_0) = R_1,$$
$$R'(t_2) \geqslant R(t_2|T_0) = R_2.$$

Эти неравенства совместно с условием 2) леммы влекут

$$R'(t_2) - R'(t_1) \geqslant m(t_2 - t_1),$$

что и требовалось доказать.

Сравним оценки $\underline{T}^{(3)}_{\min}$ и $\underline{T}^{(4)}_{\min}$.

Лемма 6.6. *Пусть для пары сглаженных интегральных графиков* $\Delta t_{\min}$ — *минимальное значение сдвига* Δt,

при котором выполняется необходимое условие существования расписаний (6.33), *т. е.*

$$R_m(t|t_0) \leqslant R_m(t|T_0 = \tau_0 + t_0 + \Delta t_{\min}).$$

Тогда найдется такой интервал концентрации $[t_1, t_2]$, $t_0 \leqslant t_1 < t_2 \leqslant t_0 + \tau_0$, *что* $M(t_1, t_2) \geqslant m(t_2 - t_1)$.

Доказательство. Случай $\Delta t_{\min} = 0$ тривиален: в качестве искомого интервала концентрации достаточно взять

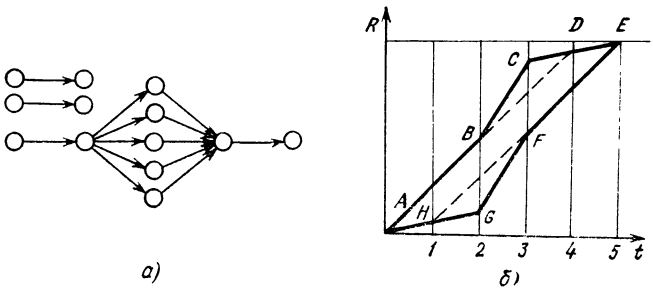

Рис. 6.6.

точку t_0 или $t_0 + \tau_0$. Пусть $\Delta t_{\min} > 0$. В этом случае несглаженные интегральные графики $R(t|t_0)$ и $R(t|T = t_0 + \tau_0 + \Delta t_{\min})$ не имеют общих точек. Однако по условию леммы множество общих точек сглаженных интегральных графиков $R_m(t|t_0)$ и $R_m(t|T_0)$ непусто. Следовательно, общие точки относятся к сглаженному отрезку, принадлежащему графикам $R_m(t|t_0)$ и $R_m(t|T_0)$. Такой отрезок по определению является отрезком прямой, образующей с осью t угол $\operatorname{arctg} m$. Кроме того, на его границах выполнены условия $(t_1, R_1) \in R(t|t_0)$ и $(t_2, R_2) \in R(t|T_0)$, поэтому справедливо $R_2 - R_1 \geqslant m(t_2 - t_1)$. Отсюда в силу леммы 6.5 $M(t_1, t_2) \geqslant m(t_2 - t_1)$.

Из лемм 6.5 и 6.6 следует, что оценки длительности кратчайших расписаний, полученные исходя из понятия интервала концентрации, не слабее оценок на основе интегральных графиков, а именно: $\underline{T}^{(3)}_{\min} \leqslant \underline{T}^{(4)}_{\min}$. На рис. 6.6, *а* показан пример сети, в которой последнее неравенство выполняется строго. Длительность каждой операции равна единице. На рис. 6.6, *б* изображены интегральные графики (*ABCDE* и *AHGFE*) и сглаженные интегральные графики при $m = 3$ (*ABDE* и *AHFE*). В данном слу-

чае имеем $\underline{T}^{(3)}_{\min} = 5 < 5\frac{2}{3} = \underline{T}^{(4)}_{\min}$. Оценка $\underline{T}^{(4)}_{\min}$ достигается на интервале концентрации [2; 3].

6.4.4. Сравнение оценок на основе интервалов концентрации и двойственных оценок. Рассмотрим задачу составления кратчайшего расписания заданной сети операций (V, E) (задача А). Без ограничения общности можно считать, что в графе (V, E) имеется только одна вершина v_n, которая не предшествует никакой другой вершине, и только одна вершина v_1, которая не следует ни за какой другой вершиной. Иначе всегда можно таким образом добавить две фиктивные вершины, чтобы полученный граф удовлетворял указанному условию. Очевидно, что в таком графе с начальной вершиной v_1 и конечной вершиной v_n для каждой вершины v_i всегда существует путь из v_1 в v_n, проходящий через v_i. Длиной пути $v_i \to v_j$ будем называть сумму длительностей операций, находящихся на этом пути.

Линейная целочисленная формулировка задачи составления кратчайшего расписания имеет вид: найти

$$T_{\min} = \min\left(\sum_{t=1}^{T_0} t x_n^t + \tau_n\right) \tag{6.41}$$

при условиях

$$\sum_{t=1}^{T_0} x_j^t = 1, \quad j = 1, \ldots, n, \tag{6.42}$$

$$\sum_{j=1}^{n} \sum_{\tau=\max(0, t-\tau_j+1)}^{t} r_j x_j^\tau \leqslant m, \quad t = 1, \ldots, T_0, \tag{6.43}$$

$$\sum_{t=1}^{T_0} t x_i^t + \tau_i \leqslant \sum_{t=1}^{T_0} t x_j^t, \quad (i, j) \in E, \tag{6.44}$$

$$x_j^t = 0 \vee 1, \quad j = 1, \ldots, n; \quad t = 1, \ldots, T_0. \tag{6.45}$$

Здесь подразумевается, что $[0, T_0]$ — интервал планирования, $T_{\min} \leqslant T_0$,

$$x_j^t = \begin{cases} 1, & \text{если операция } v_j \text{ начинается в момент} \\ & \text{времени } t, \\ 0 & \text{в противном случае.} \end{cases}$$

Набор $x = \{x_j^t,\ j = 1, \ldots, n;\ t = 1, \ldots, T_0\}$ назовем псевдорешением задачи (6.41)—(6.45), если он удовлет-

воряет ограничениям (6.42), (6.44), (6.45). Множество псевдорешений обозначим через X. Пусть элементы множества X перенумерованы. Каждому $x_q \in X$, $q = 1, \ldots, Q$, соответствуют некоторое значение функционала $c_q = \sum_{t=1}^{T_0} t x_{nq}^t + \tau_n$ и некоторые уровни потребления ресурсов

$$\alpha_q^t = \sum_{j=1}^{n} \sum_{\tau=\max(0, t-\tau_j+1)}^{t} r_j x_{jq}^{\tau}, \quad t = 1, \ldots, T_0.$$

Задачу (6.41)—(6.45) можно записать в следующем виде: найти

$$T_{\min} = \min \sum_{q=1}^{Q} c_q y_q \tag{6.46}$$

при условиях

$$\sum_{q=1}^{Q} y_q = 1, \tag{6.47}$$

$$\sum_{q=1}^{Q} \alpha_q^t y_q \leqslant m, \quad t = 1, \ldots, T_0, \tag{6.48}$$

$$y_q = 0 \vee 1, \quad q = 1, \ldots, Q, \tag{6.49}$$

где

$$y_q = \begin{cases} 1, \text{ если выбрано псевдорешение } x_q \in X, \\ 0 \text{ в противном случае.} \end{cases}$$

Пусть заданы некоторые $u_t \geqslant 0$, $t = 1, \ldots, T_0$. Рассмотрим следующую задачу: найти

$$w(u) = \min_{y_q} \left\{ \sum_{q=1}^{Q} c_q y_q + \sum_{t=1}^{T_0} u_t \left(\sum_{q=1}^{Q} \alpha_q^t y_q - m \right) \right\},$$

где минимум берется по множеству

$$\sum_{q=1}^{Q} y_q = 1,$$

$$y_q = 0 \vee 1, \quad q = 1, \ldots, Q.$$

По следствию 6.1 $w(u) \leqslant T_{\min}$ для любого $u \geqslant 0$.

Для получения более точной оценки необходимо решить задачу: найти

$$w^* = \max_{u \geqslant 0} w(u). \tag{6.50}$$

Обозначая $\pi_q = c_q + \sum_{t=1}^{\tau_0} \alpha_q^t u_t$, получаем

$$w(u) = \pi - \sum_{t=1}^{T_0} u_t m, \tag{6.51}$$

где $\pi = \min\limits_{1 \leqslant q \leqslant Q} \pi_q$.

С учетом (6.51) задачу (6.50) можно сформулировать как задачу линейного программирования: найти

$$w^* = \max \left\{ \pi - \sum_{t=1}^{T_0} u_t m \right\} \tag{6.52}$$

при условиях

$$\pi \leqslant c_q + \sum_{t=1}^{T_0} \alpha_q^t u_t, \quad q = 1, \ldots, Q, \tag{6.53}$$

$$u_t \geqslant 0, \quad t = 1, \ldots, T_0. \tag{6.54}$$

Нетрудно увидеть, что задача (6.52)—(6.54) является двойственной к задаче (6.46)—(6.49) с отброшенным условием целочисленности.

Сравним оценки $\underline{T}_{\min}^{(4)}$ и w^* для следующей задачи:

1) граф (V, E) изображен на рис. 6.7;

2) $\tau_j = 1$, $j = 1, \ldots, n$;

3) наличное количество ресурса равно m;

4) $r_j \leqslant m$, $j = 1, \ldots, n$; $\sum_{j=2}^{n-1} r_j = lm$, где $l > 1$, l — целое число.

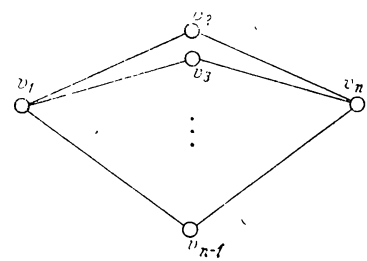

Рис. 6.7.

Вычислим $\underline{T}_{\min}^{(4)}$. Имеем $\tau_0 = 3$, экстремальный интервал концентрации

$$[t_1^*, t_2^*] = [1, 2],$$

$$\frac{1}{m} M(t_1^*, t_2^*) - (t_2^* - t_1^*) = l - 1.$$

Отсюда $\underline{T}_{\min}^{(4)} = l + 2$.

Оценим w^* сверху. Пусть $\overline{w}$ — значение функционала на некотором решении задачи (6.46)—(6.48) (условие целочисленности отброшено). Очевидно, что $w^* \leqslant \overline{w}$. Построим решение задачи (6.46)—(6.48).

Пусть $x_q = \{x_j^t, j = 1, \ldots, n; t = 1, \ldots, T_0\}$ — некоторое псевдорешение. Псевдорасписанием S_q назовем набор $\{t_j^q, j = 1, \ldots, n\}$ такой, что $t_j^q = \sum_{t=1}^{T_0} t x_{jq}^t$. Соответствие между S_q и x_q однозначно.

Рассмотрим следующие псевдорасписания S_q и соответствующие им наборы α_q^t, $t = 1, \ldots, T_0$; $q = 1, \ldots, l$:

$$S_q = \{t_1^q = 0; t_j^q = q, j = 2, \ldots, n-1; t_n^q = q+1\}, q = 1, \ldots, l;$$

$$\alpha_q^t = \begin{cases} r_1 \text{ при } t = 0, \\ \sum_{j=2}^{n-1} r = lm \text{ при } t = q, \\ r_n \text{ при } t = q+1, \\ 0 \text{ в остальных случаях.} \end{cases}$$

Пусть

$$y_q = \begin{cases} \frac{1}{l}, & q = 1, \ldots, l, \\ 0, & q = l+1, \ldots, Q. \end{cases}$$

Легко увидеть, что y_q, $q = 1, \ldots, Q$, есть допустимое решение задачи (6.46)—(6.48). Вычислим $\bar{w}$:

$$\bar{w} = \sum_{q=1}^{Q} c_q y_q = \sum_{q=1}^{l} (t_n^q + 1) \cdot \frac{1}{l} = \frac{1}{l} \sum_{q=1}^{l} (q+2) = \frac{l+5}{2}.$$

Отсюда $w^* \leqslant \bar{w} < \underline{T}_{\min}^{(4)}$.

Таким образом, несмотря на большую трудоемкость вычисления оценки w^*, оказывается, что по крайней мере для некоторых задач сравнительно простая оценка $\underline{T}_{\min}^{(4)}$ лучше, чем w^*.

6.5. Практические алгоритмы ветвей и границ

6.5.1. Две специальные задачи составления расписаний. Рассмотрим некоторые алгоритмы для решения двух известных задач: задачи минимизации длительности расписания при заданном общем уровне использования ресурсов (задача А) и задачи минимизации максимального общего уровня использования ресурсов при заданной длительности расписания (задача В). Следуя терминологии раздела 6.1, опишем эти алгоритмы с помощью нескольких ключевых терминов: подзадача, ветвление, стратегия, оценка, тест. Начнем с задачи В.

Пусть вершинам заданной сети операций (V, E) присвоены номера $i = 1, \ldots, n$ таким образом, что из условия $(v_i, v_j) \in E$ следует условие $i < j$ для всех $v_i, v_j \in V$. Подзадачи z^α, $\alpha = 0, 1, 2, \ldots$ (z^0 — исходная задача В), определяются как n-наборы пар чисел $(\underset{\_}{t}_i^\alpha, \bar{t}_i^\alpha)$, где $\underset{\_}{t}_i^\alpha$ — наиболее ранний, а $\bar{t}_i^\alpha$ — наиболее поздний сроки начала операции v_i. Величины $\underset{\_}{t}_j^\alpha$ и $\bar{t}_j^\alpha$ удовлетворяют условиям

$$\begin{aligned} \underset{\_}{t}_j^\alpha &= \max_{\{i \mid (v_i, v_j) \in E\}} \left(\underset{\_}{t}_i^\alpha + \tau_i\right), \\ \bar{t}_j^\alpha &= \min_{\{h \mid (v_j, v_h) \in E\}} \left(\bar{t}_h^\alpha - \tau_j\right) \end{aligned} \tag{6.55}$$

и могут быть найдены с помощью известных методов сетевого планирования.

Ветвление в задаче z^α определяется как последовательность шагов 1—3:

1. Выбрать операцию v_{i_0} такую, что

$$\bar{t}_{i_0}^\alpha - \underset{\_}{t}_{i_0}^\alpha = \max_{1 \leqslant i \leqslant n} \left(\bar{t}_i^\alpha - \underset{\_}{t}_i^\alpha\right) > 1.$$

2. Положить для операции v_{i_0}:

$$\underset{\_}{t}_{i_0}^{\alpha 1} = \underset{\_}{t}_{i_0}^\alpha, \quad \underset{\_}{t}_{i_0}^{\alpha 2} = \left\lceil \frac{1}{2} \left(\bar{t}_{i_0}^\alpha - \underset{\_}{t}_{i_0}^\alpha\right) \right\rceil,$$

$$\bar{t}_{i_0}^{\alpha 1} = \left\lceil \frac{1}{2} \left(\bar{t}_{i_0}^\alpha - \underset{\_}{t}_{i_0}^\alpha\right) \right\rceil, \quad \bar{t}_{i_0}^{\alpha 2} = \bar{t}_{i_0}^\alpha.$$

3. Образовать подзадачи $z^{\alpha 1}$ и $z^{\alpha 2}$, в которых ранний и поздний сроки операции v_{i_0} определены на шаге 2, а ранние и поздние сроки начала остальных операций определены из условий (6.55).

Подзадачи $z^{\alpha 1}$ и $z^{\alpha 2}$ называются (непосредственными) потомками z^α; подзадача z^α называется (непосредственным предшественником подзадач $z^{\alpha 1}$ и $z^{\alpha 2}$. Ветвление определяет разбиение множества допустимых решений подзадачи на непересекающиеся классы, образующие множества решений подзадач $z^{\alpha 1}$ и $z^{\alpha 2}$. В свою очередь, подзадачи $z^{\alpha 1}$ и $z^{\alpha 2}$ допускают ветвление, если в них найдется операция v_{i_0}, удовлетворяющая условию шага 1 ветвления. Дерево подзадач определяется как ориентированный граф (Z, U), вершинами которого являются подзадачи z^α, $\alpha = 0, 1, 2, \ldots$, а дуги $(z^\alpha, z^\beta) \in U$ тогда и только тогда, когда z^β является непосредственным потомком z^α, причем z^0 — корневая вершина.

Пару вершин-подзадач назовем независимой, если в дереве подзадач отсутствуют пути, соединяющие вершины z^α и z^β. Множество вершин-подзадач $I \subset Z$ назовем независимым, если каждая пара его вершин независима. Стратегией будем называть функцию $h(I)$, которая каждому независимому множеству подзадач I сопоставляет некоторую подзадачу из I.

Множество висячих вершин дерева подзадач, т. е. таких подзадач z^α, что $\underline{t}_i^\alpha = \bar{t}_i^\alpha$ для всех $i = 1, \ldots, n$, взаимно однозначно соответствует множеству всех допустимых расписаний в исходной задаче z^0. Просматривая висячие вершины в некотором порядке, в принципе можно найти оптимальное решение. Такой способ решения задачи В может оказаться практически неприемлемым из-за экспоненциального роста общего числа висячих вершин (как функции n). Сокращения объема вычислений во многих случаях можно достичь за счет более детального анализа подзадач в процессе их порождения.

Нижнюю оценку подзадачи z^α определим как

$$\underline{m}^\alpha = \max_{1 \leqslant p < q \leqslant T} \frac{M(p, q)}{q - p},$$

где

$$M(p, q) = \sum_{i=1}^{n} r_i \min\left(\left| (\underline{t}_i^\alpha, \underline{t}_i^\alpha + \tau_i) \cap (p, q) \right|, \right.$$
$$\left. \left| (\bar{t}_i^\alpha, \bar{t}_i^\alpha + \tau_i) \cap (p, q) \right| \right).$$

Верхнюю оценку подзадачи z^α определим как

$$\bar{m}^\alpha = \min\left(\max_{1 \leqslant t \leqslant T} r_{\underline{S}}(t), \max_{1 \leqslant t \leqslant T} r_{\bar{S}}(t) \right), \qquad (6.56)$$

где

$$\underline{S} = (\underline{t}_1^\alpha, \ldots, \underline{t}_n^\alpha), \quad \bar{S} = (\bar{t}_1^\alpha, \ldots, \bar{t}_n^\alpha)$$

— расписания, в которых операции выполняются соответственно по ранним и поздним срокам. Очевидно, оптимальное значение функционала в задаче В ограничено сверху величиной $\bar{m}^\alpha$. Пусть $\bar{S}^{(\alpha)}$ — расписание, для которого достигается минимум в (6.56). Множество подзадач $Y \subseteq Z$ назовем правильным, если вместе с каждой подзадачей z^β оно содержит всех ее предшественников. В дальнейшем условимся рассматривать только правильные

множества подзадач. Если Y — правильное множество подзадач, то независимое множество $I(Y)$ по определению состоит из всех таких и только таких подзадач, у которых в Y не найдется предшественников.

Пусть дано множество подзадач $Y \subseteq Z$. Верхний рекорд определим как

$$\overline{m}^*(Y) = \min_{z^\alpha \in Y} \overline{m}^\alpha. \tag{6.57}$$

Через $S^*(Y)$ обозначим расписание в подзадаче, для которой достигается минимум в (6.57). Нижний рекорд определим как

$$\underline{m}^*(Y) = \min_{z^\alpha \in I(Y)} \underline{m}^\alpha.$$

Тестом подзадачи $z^\alpha \in Y$ назовем операцию, заключающуюся в проверке условия $\underline{m}^\alpha \geqslant \overline{m}^*(Y)$.

Множество тестированных подзадач, которое также будем считать правильным, обозначим $\overline{Y}$. Пусть определено некоторое множество $\overline{Y}$. Решением задачи B назовем пару $(\overline{S}^*(\overline{Y}), \overline{m}^*(\overline{Y}))$.

Алгоритм ветвей и границ в задаче B может быть описан следующим образом:

M0 (начало). Положить $Y = I = \{z^0\}$, $\overline{Y} = \varnothing$, $\overline{m}^* = \underline{m}^* = \infty$, $S^* = \varnothing$.

M1 (проверка останова). Если $|I| = 0$, то перейти к M6.

M2 (стратегия). Выбрать подзадачу $z^\alpha = h(I)$.

M3 (оценки). Определить $\underline{m}^\alpha$, $\overline{m}^\alpha$, $\overline{S}^\alpha$.

M4 (тест). Если $\underline{m}^\alpha \geqslant \overline{m}^*(Y)$, то $I := I \backslash \{z^\alpha\}$; перейти к M1. В противном случае модифицировать $\overline{m}^*$, $\underline{m}^*$; $\overline{Y} \leftarrow \overline{Y} \cup \{z^\alpha\}$.

M5 (ветвление). Сформировать $z^{\alpha 1}$ и $z^{\alpha 2}$ и положить $I \leftarrow (I \backslash \{z^\alpha\}) \cup \{z^{\alpha 1}, z^{\alpha 2}\}$. Перейти к M1.

M6 (останов). $\overline{m}^*$, $\overline{S}^*$ — решение задачи.

Метод решения задачи A при условии $r(t) = m = \text{const}$ состоит в решении последовательности задачи B при фиксированных значениях срока завершения операций $\overline{T}$:

Шаг 1. Зафиксировать $\overline{T}$, $\Delta \geqslant 0$, $k = 0$.

Шаг 2. $k := k + 1$.

Шаг 3. Решить задачу B при $T = \overline{T}$.

Шаг 4. Если $m^*(\overline{T}) > m$, то положить $\overline{T} := \overline{T} + \Delta$ и перейти к шагу 2; иначе перейти к шагу 5.

Шаг 5. Закончить; $\overline{T}$ — решение задачи A при уровне наличия ресурса m.

Отметим, что нахождение оптимального решения для практических задач большой размерности может оказаться крайне трудоемким. Кроме того, в ряде случаев доказательство оптимальности полученного решения занимает намного больше времени, чем его поиск. Это подтверждается и приведенными ниже результатами счета. Поэтому при решении задач A и B целесообразно использовать и другие виды останова (кроме останова по пустому множеству I):

— по достижении заданного значения погрешности

$$\varepsilon = (\overline{m}^*(Y) - \underline{m}^*(Y))/\underline{m}^*(Y);$$

— по максимальному количеству подзадач в I (определяет потребности в оперативной памяти ЭВМ для хранения подзадач);

— по количеству анализируемых подзадач (определяет время счета).

Для экспериментальной оценки эффективности алгоритмов Б. Е. Поляченко разработана программа на ПЛ/1 версии 6.1 ОС ЕС, использующая указанные виды останова. С ее помощью было решено 7 практических задач планирования вычислительного процесса в АСУ. Отношение порядка на множестве V определялось информационными связями между программными модулями АСУ. Под уровнем потребления ресурса понимался объем оперативной памяти, требуемой для загрузочного модуля каждой программы. Некоторые результаты машинных экспериментов по решению практических задач на ЭВМ ЕС-1050 приведены в табл. 6.2 и 6.3. При решении задач 1—5 использовалась стратегия одностороннего обхода *LIFO*,

Таблица 6.2

Задача	$\|V\|$	$\|E\|$	Общее процессорное время счета, мин	Относительная погрешность	Кол-во проанализированных подзадач	Вид останова алгоритма	Используемый объем оперативной памяти ЭВМ, байт
1	6	7	13,22	0	968	по достижении оптимума	36
2	12	16	19,00	0,17	363	по времени	68
3	31	34	7,51	0,28	20	по числу подзадач	80
4	40	51	16,34	0,4	142	по ε	88
5	61	82	12,03	0,58	50	по числу подзадач	106
6	61	82	20,00	0,48	126	по времени	166

что позволяет решать задачи большой размерности. Задачи 5 и 6 — одна и та же задача, решенная с использованием двух различных стратегий: *LIFO* (задача 5) и по минимальной сумме верхней и нижних оценок в списке активных подзадач (задача 6). Как видно из табл. 6.2,

Таблица 6.3

Изменение значений погрешности на различных шагах алгоритма для задачи 4 из табл. 6.2

Номер подзадачи	1	5	10	26	93	112	132	142
Достигнутая относительная погрешность	9,82	0,70	0,62	0,58	0,51	0,46	0,42	0,4

изменение стратегии не привело к существенному снижению гарантированной погрешности при ограниченном времени счета. В табл. 6.3 приведено изменение значений погрешности на различных шагах алгоритма для задачи В.

6.5.2. Задача с независимыми цепями операций. Рассмотрим следующую задачу. Требуется выполнить совокупность из m работ. Работа i, $i = 1, \ldots, m$, определена как последовательность из n_i операций, имеющих номера с 1 по n_i. Длительности операций являются целыми числами τ_{ij} (где ij обозначает j-ю операцию работы i). Имеется K различных видов ресурсов, причем $r_{ij}^k \geqslant 0$ — количество k-го ресурса, используемого в процессе выполнения операции ij; R_{kt} — общее количество k-го ресурса в период времени t; время t измеряется целыми числами, $1 \leqslant t \leqslant T$. Начатые операции не разрешается прерывать. Выполнение операции состоит в предоставлении ей необходимого количества ресурсов на время τ_{ij}. Пусть t_{ij} обозначает время начала выполнения операции ij.

Для каждой операции ij заданы ранние $\underline{t}_{ij}$ и поздние $\bar{t}_{ij}$ сроки начала выполнения, т. е. для t_{ij} должно выполняться условие

$$\underline{t}_{ij} \leqslant t_{ij} \leqslant \bar{t}_{ij}.$$

Положим, $I_t = \{ij \mid t_{ij} \leqslant t < t_{ij} + \tau_{ij}\}$ — множество операций, выполняющихся в момент времени t; пусть, далее, $f_{ij} = t_{ij} + \tau_{ij}$ и $f_i = \max\limits_{1 \leqslant j \leqslant n_i} f_{ij} = f_{in_i}$ — время завершения работы ij. Технологические ограничения следования опера-

ций выражаются условиями вида

$$t_{ij} \geqslant t_{il} + \tau_{il},$$

если в последовательности операций работы i операция ij непосредственно следует за il.

В каждый момент времени t должны выполняться ограничения по ресурсам:

$$\sum_{ij \in I_t} r_{ij}^k \leqslant R_{kt}, \quad k = 1, \ldots, K, \quad 1 \leqslant t \leqslant T.$$

Набор чисел t_{ij}, удовлетворяющий указанным условиям, определяет план выполнения работ — расписание. На множестве расписаний определена целевая функция

$$z = \sum_{i=1}^{m} g_i(f_i),$$

где $g_i(f_i)$ — неубывающие функции. Требуется найти расписание с минимальным значением z.

Описанную экстремальную задачу будем решать методом ветвей и границ, причем в качестве нижних границ используются решения двойственной задачи. Построим дальнейшее изложение по следующему плану: рассмотрим соответствующую задачу целочисленного линейного программирования и ее лагранжеву релаксацию, метод решения вспомогательной дискретной задачи и организацию спуска по множителям Лагранжа, некоторые особенности организации алгоритма ветвей и границ, план и результаты численных экспериментов.

Выпишем линейную целочисленную формулировку задачи. Положим

$$x_{ij}^t = \begin{cases} 1, & \text{если операция } ij \text{ начинается в} \\ & \text{момент времени } t. \\ 0 & \text{в противном случае;} \end{cases}$$

$$c_i^t = g_i\left(t + \tau_{in_i}\right). \quad .$$

Пусть $T < \infty$ — общая длительность интервала планирования. Тогда экстремальная задача построения расписания состоит в том, чтобы найти

$$\min \sum_{i=1}^{m} \sum_{t=1}^{T} c_i^t x_{in_i}^t \tag{6.58}$$

при условиях

$$\sum_{t=\underline{t}_{ij}}^{\bar{t}_{ij}} x_{ij}^t = 1, \quad i = 1, \ldots, m; \quad t = 1, \ldots, T, \tag{6.59}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{\tau=\max(0, t-\tau_{ij}+1)}^{t} r_{ij}^k x_{ij}^\tau \leqslant R_{kt},$$

$$k = 1, \ldots, K; \quad t = 1, \ldots, T, \tag{6.60}$$

$$\sum_{t=1}^{T} t x_{ij}^t + \tau_{ij} \leqslant \sum_{t=1}^{T} t x_{i,j+1}^t,$$

$$i = 1, \ldots, m; \quad j = 1, \ldots, n_i - 1, \tag{6.61}$$

$$x_{ij}^t = 0 \vee 1,$$

$$i = 1, \ldots, m; \quad j = 1, \ldots, n_i; \quad t = 1, \ldots, T. \tag{6.62}$$

Набор $x = \{x_{ij}^t\}$ назовем псевдорешением задачи (6.58)—(6.62), если он удовлетворяет ограничениям (6.59), (6.61), (6.62). Множество всех псевдорешений обозначим X.

Рассмотрим функцию Лагранжа

$$L(x, u) = \sum_{i=1}^{m} \sum_{t=1}^{T} c_i^t x_{in_i}^t + \sum_{t=1}^{T} \sum_{k=1}^{K} u_{kt} \left(\sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{\tau=\max(0, t-\tau_{ij}+1)}^{t} r_{ij}^k x_{ij}^\tau - R_{kt} \right).$$

Определим следующую экстремальную задачу: найти

$$w(u) = \min_{x \in X} L(x, u). \tag{6.63}$$

Так как X — конечное множество, то $w(u)$ — кусочно-линейная вогнутая функция. Положим

$$w^* = \max_{u \geqslant 0} w(u). \tag{6.64}$$

Как известно, если v^* — оптимальное значение задачи (6.58)—(6.62), то $w^* \leqslant v^*$.

Рассмотрим задачу (6.63). Набор $x_i = \{x_{ij}^t\}$, $j = 1, \ldots, n_i$; $t = 1, \ldots, T$, назовем расписанием i-й последовательности операций, если он удовлетворяет ограничениям (6.59), (6.61), (6.62). Множество всех таких наборов для i-й последовательности обозначим $\overline{X}_i$. Нетрудно видеть, что $X = \overline{X}_1 \times \ldots \times \overline{X}_m$.

Определим

$$\pi_i(x_i, u) = \sum_{t=1}^{T} c_i^t x_{in_i}^t + \\ + \sum_{t=1}^{T} \sum_{k=1}^{K} u_{kt} \sum_{j=1}^{n_i} \sum_{\tau=\max(0, t-\tau_{ij}+1)} r_{ij}^k x_{ij}^\tau, \quad (6.65)$$

где $x_i \in \overline{X}_i$, $i = 1, \ldots, m$.

С учетом (6.65) задачу (6.63) можно представить в следующем виде:

$$w(u) = \min_{x_1, \ldots, x_{n_i}} \left\{ \sum_{i=1}^{m} \pi_i(x_i, u) - \sum_{t=1}^{T} \sum_{k=1}^{K} u_{kt} R_{kt} \right\} = \\ = \sum_{i=1}^{m} \min_{x_i \in \overline{X}_i} \pi_i(x_i, u) - \sum_{t=1}^{T} \sum_{k=1}^{K} u_{kt} R_{kt}. \quad (6.66)$$

Таким образом, для решения задачи (6.63) достаточно для каждой последовательности операций найти $\pi_i(u) = \min_{x_i \in \overline{X}_i} \pi_i(x_i, u)$ или, более подробно, решить задачу (индекс i опущен): найти

$$\pi(u) = \min \left\{ \sum_{t=1}^{T} c^t x_n^t + \right. \\ \left. + \sum_{t=1}^{T} \sum_{k=1}^{K} u_{kt} \sum_{j=1}^{n} \sum_{\tau=\max(0, t-\tau_{ij}+1)}^{t} r_{ij}^k x_{ij}^\tau \right\}, \quad (6.67)$$

$$\sum_{t=t_j}^{\bar{t}_j} x_j^t = 1, \quad j = 1, \ldots, n, \quad (6.68)$$

$$\sum_{t=1}^{T} t x_j^t + \tau_j \leqslant \sum_{t=1}^{T} t x_{j+1}^t, \quad j = 1, \ldots, n-1, \quad (6.69)$$

$$x_j^t = 0 \vee 1, \quad j = 1, \ldots, n; \quad t = 1, \ldots, T. \quad (6.70)$$

Опишем метод динамического программирования для решения задачи (6.67)—(6.70). Множеству решений задачи поставим в соответствие сеть (V, E, H), вершины которой соответствуют различным моментам начала выполнения операций. Сеть (V, E, H) состоит из $(n+2)$ слоев и удовлетворяет условиям 1—3:

1. Множество вершин: $V = \left\{0, \bigcup_{j=1}^{n} V_j, 1\right\}$. Множество V_j, $j = 1, \ldots, n$, состоит из всех вершин v_j^t таких, что $\underline{t}_j \leqslant t \leqslant \bar{t}_j$, t — целое.

2. Дугами связаны только те вершины, которые принадлежат соседним слоям V_j и V_{j+1}, $j = 0, 1, \ldots, n$;

а) вершина 0 (слой 0) связана дугами $(0, v_1^t)$ со всеми вершинами первого слоя, $v_1^t \in V_1$, $t = \underline{t}_1, \ldots, \bar{t}_1$;

б) пара вершин $v_j^{t_1} \in V_j$ и $v_{j+1}^{t_2} \in V_{j+1}$ из соседних слоев связана дугой $\left(v_j^{t_1}, v_{j+1}^{t_2}\right)$, если $t_1 + \tau_j \leqslant t_2$;

в) вершина 1 (слой $n+1$) связана дугами $(v_n^t, 1)$ со всеми вершинами n-го слоя, $v_n^t \in V_n$, $t = \underline{t}_n, \ldots, \bar{t}_n$.

3. Веса, сопоставленные вершинам (множество H), подсчитываются следующим образом:

$$h(0) = h(1) = 0,$$

$$h\left(v_j^t\right) = \sum_{\tau=t}^{t+\tau_j-1} \sum_{k=1}^{K} r_j^k u_{k\tau}, \quad j = 1, \ldots, (n-1),$$

$$h\left(v_n^t\right) = \sum_{\tau=t}^{t+\tau_n-1} \sum_{k=1}^{K} r_n^k u_{kt} + c^t.$$

Нетрудно видеть, что каждому пути $\left(0, v_1^{t_1}, \ldots, v_n^{t_n}, 1\right)$ из вершины 0 в вершину 1 в сети (V, E, H), если он существует, соответствует допустимое решение $\tilde{x} = \{\tilde{x}_j^t\}$ задачи (6.67)—(6.70):

$$\tilde{x}_j^t = \begin{cases} 1, & \text{если} \quad t = \tilde{t}_j, \\ 0 & \text{в противном случае,} \end{cases}$$

причем $\pi(\tilde{x}, u) = \sum_{j=1}^{n} h\left(v_j^{\tilde{t}_j}\right)$. Задача отыскания $\min_{x \in X} \pi(x, u)$ сводится, таким образом, к нахождению в сети (V, E, H) кратчайшего пути, соединяющего вершины 0 и 1. Учитывая простую структуру сети, алгоритм динамического программирования для этой задачи можно организовать таким образом, что его трудоемкость будет $\sim nT$.

Рассмотрим некоторые особенности организации алгоритма ветвей и границ, реализованные в соответствующей программе ЕС ЭВМ.

Генерирование новых подзадач. Как и ранее, $\underline{t}_{ij}$, $\bar{t}_{ij}$ обозначают соответственно ранние и поздние

сроки начала выполнения операции ij (для исходной задачи — $\underline{t}^0_{ij}$, $\bar{t}^0_{ij}$). Будем говорить, что сроки $\underline{t}_{ij}$, $\bar{t}_{ij}$ скорректированы, если

$$\underline{t}_{i1} \geqslant 1, \quad \bar{t}_{in_i} + \tau_{in_i} \leqslant T, \quad i = 1, \ldots, m,$$

$$t_{i(j-1)} + \tau_{i(j-1)} \leqslant \underline{t}_{ij} \leqslant \bar{t}_{ij} \leqslant t_{i(j+1)} - \tau_{ij},$$

$$i = 1, \ldots, m; \quad j = 2, \ldots, n_i - 1.$$

Каждая вершина дерева ветвления (подзадачи) идентифицируется набором $\{\underline{t}_{ij}, \bar{t}_{ij}\}$, $i = 1, \ldots, m$; $j = 1, \ldots, n_i$. Резервом времени операции ij назовем разность $W_{ij} = \bar{t}_{ij} - \underline{t}_{ij}$.

Пусть для ветвления выбрана подзадача $\{\underline{t}_{ij}, \bar{t}_{ij}\}$ и операция $i'j'$ имеет максимальный резерв времени $W_{i'j'}$. Полагая $\underline{t}^1_{i'j'} = \underline{t}_{i'j'}$, $\bar{t}^1_{i'j'} = \left\lfloor \dfrac{\underline{t}_{i'j'} + \bar{t}_{i'j'}}{2} \right\rfloor$, $\underline{t}^2_{i'j'} = \left\lfloor \dfrac{\underline{t}_{i'j'} + \bar{t}_{i'j'}}{2} \right\rfloor + 1$, $\bar{t}^2_{i'j'} = \bar{t}_{i'j'}$ и корректируя ранние и поздние сроки остальных операций, получаем две новые подзадачи $\{\underline{t}^1_{ij}, \bar{t}^1_{ij}\}$ и $\{\underline{t}^2_{ij}, \bar{t}^2_{ij}\}$. Здесь $\lceil x \rceil$ — наибольшее целое, не превышающее x; $\lfloor x \rfloor$ — наименьшее целое, большее x.

Вычисление оценок. Для решения задачи (6.64) используется метод обобщенного градиентного спуска с растяжением пространства. Обобщенный градиент, как легко видеть, равен

$$G_{kt}(u) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{\tau=\max(0, t-\tau_{ij}+1)}^{t} r^k_{ij} x^{\tau}_{ij}(u) - R_{kt},$$

где $\{x^{\tau}_{ij}(u)\}$ — решение задачи (6.63) при заданном u. Обозначим через u_0 начальные значения множителей Лагранжа. Тогда для исходной задачи $u_0 = 0$; в дальнейшем при вычислении оценки порожденной подзадачи полагаем $u_0 = \bar{u}$, где $\bar{u}$ — наилучшее значение множителей для порождающей подзадачи.

Число итераций градиентного спуска выбирается сравнительно небольшим (~ 50) ввиду большой трудоемкости вычислений значений функции $w(u)$ и градиента $G_{kt}(u)$.

Пусть вершина $\{\underline{t}'_{ij}, \bar{t}'_{ij}\}$ — непосредственный потомок вершины $\{\underline{t}_{ij}, \bar{t}_{ij}\}$, полученный при уменьшении резерва

времени операции ij. Рассмотрим решение $\{x^t_{ij}(\bar{u})\}$ ($\{\tilde{x}^t_{ij}(u)\}$) задачи (6.64), соответствующее вершине $\{t_{ij}, \bar{t}_{ij}\}$ (соответственно $\{t'_{ij}, \bar{t}'_{ij}\}$), при $u=\bar{u}$. Очевидно, $t'_{ij}=t_{ij}$, $\bar{t}'_{ij}=\bar{t}_{ij}$, если $i\neq i'$, откуда следует $\tilde{x}^t_{ij}(\bar{u})=x^t_{ij}(\bar{u})$ при $i\neq i'$, т. е. при вычислении оценки $w(\bar{u})$ для новой вершины $\{t'_{ij}, \bar{t}'_{ij}\}$ достаточно найти решение задачи (6.63) только для той последовательности операций, для которой изменились ранние и поздние сроки.

Таким образом, для уменьшения трудоемкости алгоритма в целом представляется целесообразным вычислять оценку $w(u)$ для некоторой части вершин при фиксированных значениях множителей Лагранжа.

Пересчет множителей (решение задачи (6.64)) делается для всех вершин дерева ветвления, ранг которых (расстояние от корня дерева) равен pn, $n=0, 1, \ldots$, где p — параметр алгоритма.

Построение допустимого расписания. Сделаем одно замечание нематематического характера. Поскольку исследуемая нами задача является *NP*-полной в смысле Кука — Карпа, следует ожидать, что поиск точного решения задачи потребует экспоненциального объема вычислений. Синтезируя алгоритм ветвей и границ, будем стремиться поэтому к получению наилучшего по функционалу решения в заданное время счета. Технически эта идея может быть реализована посредством организации внутри общей схемы алгоритма ветвей и границ специальной процедуры пересчета верхних границ. Верхнюю границу в исследуемой нами задаче дает любой эвристический алгоритм ее решения. Описываемый ниже алгоритм оперирует информацией, содержащейся в псевдорешениях подзадачи, т. е. в результатах вычисления нижних границ. Опишем соответствующий алгоритм построения допустимых расписаний.

Расписание $\{t_{ij}\}$ будем называть активным, если не существует другого расписания $\{t'_{ij}\}$ такого, что $t'_{ij}\neq t_{ij}$ по крайней мере для одной операции и $t'_{ij}\leqslant t_{ij}$ для всех ij. Очевидно, что в множестве активных расписаний содержится хотя бы одно оптимальное.

Пусть задан набор приоритетов операций $\{\sigma_{ij}\}$ такой, что $\sigma_{ij}\neq\sigma_{i'j'}$, $i'j'\neq ij$, $\sigma_{ij}<\sigma_{i'j'}$, если операция ij предшествует операции $i'j'$. По набору $\{\sigma_{ij}\}$ нетрудно построить некоторое активное расписание. Для этого достаточно последовательно «наращивать» начальный отрезок рас-

писания, выбирая операции в порядке возрастания значений σ_{ij} и «сдвигая» их максимально влево с учетом ранних сроков исходной задачи и ограничений (6.60), (6.61). Такая процедура имеет трудоемкость $\sim T\sum_{i=1}^{n} n_i$.

Допустимое решение строится всякий раз, когда пересчитываются множители Лагранжа (после приближенного решения задачи (6.64)). Пусть наилучшая оценка $w(u)$ вершины $\{\underline{t}_{ij}, \bar{t}_{ij}\}$ получена при $u=\bar{u}$; $\{x_{ij}^t(u)\}$ — соответствующее псевдорешение. Тогда в качестве набора приоритетов операций выбирается набор $\{\sigma_{ij}\}$, удовлетворяющий следующим условиям:

1) $\sigma_{ij} \neq \sigma_{i'j'}$, если $ij \neq i'j'$;

2) $\sigma_{ij} > \sigma_{i'j'}$, если $\sum_{t=1}^{T} t x_{ij}^t(\bar{u}) > \sum_{t=1}^{T} t x_{i'j'}^t(\bar{u})$;

3) в случае $\sum_{t=1}^{T} t x_{ij}'(\bar{u}) = \sum_{t=1}^{T} t x_{i'j'}(\bar{u})$ полагаем $\sigma_{ij} > \sigma_{i'j'}$, если $\sum_{t=1}^{T} c_i^t x_{in_i}^t(\bar{u}) < \sum_{t=1}^{T} c_i^t x_{i'n_{i'}}(\bar{u})$.

Для описания стратегии выбора вершины для ветвления определим приоритет W вершины $\{\underline{t}_{ij}, \bar{t}_{ij}\}$:

$$W = w + \alpha \sum_{i=1}^{m} \sum_{j=1}^{n_i} (\bar{t}_{ij} - \underline{t}_{ij}),$$

где w — оценка вершины $\{\underline{t}_{ij}, t_{ij}\}$; α — параметр стратегии.

Каждый раз для ветвления выбирается вершина, имеющая наименьший приоритет. Если в некоторый момент окажется, что $\bar{W} > \bar{v}^0$, где $\bar{W}$ — приоритет наилучшей вершины, $\bar{v}^0$ — текущий рекорд, полагаем $\alpha := \alpha q$, $0 \leqslant q \leqslant 1$.

Вычислительный процесс оканчивается, если истекло время, выделенное для решения задачи, или получено оптимальное решение. Нетрудно видеть, что если $q=1$ и α — достаточно большое число, то такая стратегия ветвления эквивалентна одностороннему обходу дерева ветвления; при $\alpha=0$ получаем поиск по наилучшей оценке.

Алгоритм ветвей и границ был реализован Ю. П. Лаптиным на языке ПЛ/1 для ОС ЕС ЭВМ. Для запоминания дерева ветвления (информации о подзадачах) использовалась внешняя память (диски). Вычислительные эксперименты проводились на ЭВМ ЕС-1040 для серии из шести тестовых задач. Каждая задача просчитывалась

Таблица 6.4

Характеристики задач

№ задачи	1	2	3	4	5	6
Типы последовательностей	1	1	2	2	1-я, 2-я, 3-я—тип 2 4-я, 5-я — тип 1	
Количество наличного ресурса	9	8	9	8	9	8

Таблица 6.5

Результаты счета

		Задача 1				Задача 2				Задача 3			
p		1	2	3	4	1	2	3	4	1	2	3	4
N		54	110	160	200	54	110	160	200	57	105	140	200
Односторонний обход дерева ветвления	f	39	39	38	39	53	57	57	51	60	58	60	54
	N^*	39	39	60	70	44	41	44	60	6	71	4	116
Поиск по наилучшей оценке	f	39	39	39	39	62	58	51	51	66	61	59	59
	N^*	42	42	46	46	44	47	121	110	2	102	95	112
$\alpha=1$	f	39	39	38	38	58	51	51	51	57	57	57	57
$q=0{,}8$	N^*	42	54	109	108	6	68	72	94	43	95	113	118

		Задача 4				Задача 5				Задача 6			
p		1	2	3	4	1	2	3	4	1	2	3	4
N		57	105	140	200	54	105	180	240	54	105	180	240
Односторонний обход дерева ветвления	f	65	61	61	61	51	51	50	51	58	56	56	56
	N^*	34	78	98	96	6	10	65	12	6	54	39	63
Поиск по наилучшей оценке	f	61	61	61	61	56	52	51	50	56	56	56	56
	N^*	50	50	63	96	1	77	176	234	34	34	26	28
$\alpha=1$	f	61	61	61	61	51	51	50	50	56	56	55	56
$q=0{,}8$	N^*	38	38	41	38	6	10	135	73	42	51	18	36

при различных значениях параметров алгоритма. Время счета одного варианта — 10 мин. Тестовые задачи составлялись из последовательностей операций двух типов (в каждой последовательности 4 операции, все операции потребляют один вид ресурса).

Тип 1: $\tau_i = 3, 1, 1, 2$; $r_i = 3, 4, 6, 3$;

Тип 2: $\tau_i = 3, 1, 3, 2$; $r_i = 4, 6, 2, 3$.

Каждая задача состояла из пяти последовательностей; функции $g_i(f_i)$, $i = 1, \ldots, 5$, и интервал планирования T для всех задач одинаковы: $T = 30$, $g_i(f_i) = \max(0, d_i - f_i)$, где $d_i = 1 + 2 \cdot i$, $i = 1, \ldots, 5$. Таким образом, каждая задача определяется набором последовательностей и количеством наличного ресурса ($R_t = R$ — постоянная).

Характеристики задач приведены в табл. 6.4.

Результаты счета приведены в табл. 6.5. Обозначения: f — наилучший рекорд, N — число построенных подзадач, N^* — число построенных подзадач к моменту последнего уточнения рекорда, α, q, p — параметры алгоритма.

Из результатов эксперимента следует, что для решения рассмотренных задач наиболее целесообразно применять смешанную стратегию $\alpha = 1$, $q = 0{,}8$ с параметром $p > 1$. Необходимо отметить также, что при $p > 2$ все стратегии дают приблизительно одинаковые решения.

Анализ работы алгоритма показывает, что в ходе вычислений оценка снизу оптимального решения растет сравнительно медленно и по окончании работы алгоритма имеет близкие значения для всех вариантов счета одной задачи. Это можно объяснить относительно малым числом построенных подзадач. Приведем оценки оптимальных решений (w_i — наилучшая оценка i-й задачи):

i:	1;	2;	3;	4;	5;	6;
w_i:	32,3;	36,4;	46,5;	51,1;	40,3;	44,3.

Библиографический комментарий

Метод ветвей и границ вкладывается в общую схему метода последовательного анализа вариантов, являясь его специальным случаем. Согласно общей схеме условие задачи, подлежащей решению, описывается в виде множества вариантов и совокупности опытов, тестов и т. п., с результатами которых связаны правила исключения (или сохранения) вариантов. Процедура принятия решения в общих чертах такова. Исходное множество вариантов разбивается на определенные подмножества путем введения дополнительных высказываний, характеризующих каждое подмножество. Часть подмножеств стараются исключить, используя условия несовместимости (логической противоречивости) высказываний, относящихся

к элементам подмножества, и высказываний, характеризующих требования к искомому решению задачи. С оставшимися неисключенными подмножествами производится аналогичная операция разбиения и исключения и т. д. Процесс решения задачи является многоступенчатым, последовательным. Каждая ступень связана с проверкой тех или иных свойств у соответствующих ей подмножеств вариантов и ведет либо к непосредственному сокращению исходного множества, либо подготавливает возможность такого сокращения на следующих этапах [29, 30]. В рамках указанной общей схемы метод ветвей и границ характеризуется: а) определенным способом разбиения множества допустимых решений на подмножества; б) определенным набором опытов; главный из них состоит в решении оценочной задачи и сравнении оценки и значения лучшего из найденных к данному этапу решений; в) определенной стратегией, порядком производства опытов.

Структура алгоритмов ветвей и границ описана на основе обзора [1] (на русском языке основные идеи из [1] повторены в обзоре [2]). Использование множителей Лагранжа в структуре метода ветвей и границ изложено на основе [3]. Применение множителей Лагранжа для решения задач теории расписаний, по-видимому, впервые описано в [4], затем развито в [5]. Метод использования множителей Лагранжа в целочисленном программировании предложен в [6]. Переменные, использованные в формулировке задачи (6.13)—(6.17), по-видимому, впервые использованы в [7]. В данной главе приведены результаты, относящиеся к сетевым задачам общего вида [8] и к задачам на цепях операций [9]. Близкий к [9] результат, но применительно к несетевой задаче одного станка, получен в [10]. В большинстве практических сетевых задач составления расписаний речь идет или о поиске кратчайшего расписания или о минимизации максимального уровня потребления ресурсов (задачи А и В). В этих случаях имеется возможность решения оценочных задач более эффективными специальными методами, чем методы множителей Лагранжа. Речь идет об оценках на основе интегральных графиков потребностей в ресурсах. Оценки на основе интегральных графиков введены в [11] и развиты в [12 — 14]. Оценки на основе интервалов концентрации введены в [15] и развиты в [16, 17]. Сравнение оценок обоих типов произведено в [18]. Сравнение оценки кратчайшего расписания на основе интервалов концентрации и с использованием множителей Лагранжа произведено в [19]. Алгоритм ветвей и границ с использованием оценок на основе интервалов концентрации описан в [20], а с использованием множителей Лагранжа в задаче с независимыми цепями операций — в [9].

Упомянутые выше методы решения оценочных задач можно отнести к уточненным методам в том смысле, что получаемые оценки являются практически достаточно точными. Их имеет смысл использовать в алгоритмах ветвей и границ, заранее ориентированных на анализ небольшого числа подзадач и получение приближенных с апостериорной оценкой погрешности решений. В других случаях могут быть использованы более простые методы получения оценок. Ряд упрощенных методов решения оценочных задач приводится в [21—23].

К главе I

1. Основные положения по разработке и применению систем сетевого планирования и управления.— М.: Статистика, 1974.— 216 с.
2. I Всесоюзная конференция по математическим вопросам сетевого планирования и управления (тезисы и доклады).— Киев: ИК АН УССР, 1967.— 249 с.
3. Голенко Д. И. Статистические методы сетевого планирования и управления.— М.: Наука, 1968.— 400 с.
4. Бурков В. Н. и др. Сетевые модели и задачи управления.— М.: Сов. радио, 1967.— 163 с.
5. Зимин И. Н., Иванилов Ю. П. Решение задач сетевого планирования сведением их к задачам оптимального управления.— ЖВМ и МФ, 1971, 3, № 3, с. 632—641.
6. Зимин И. Н. Алгоритм расчета сетей при переменных интенсивностях выполнения операций.— Известия АН СССР. Технич. кибернетика, 1973, № 6, с. 17—23.
7. Голиков А. И., Деменчук В. И. Об оптимальном распределении ресурсов в планировании комплексов операций.— Автоматика и телемеханика, 1969, 12, № 2, с. 60—70.
8. Разумихин Б. С. Задача об оптимальном распределении ресурсов.— Автоматика и телемеханика, 1967, № 1, с. 62—74.
9. Петрушинин Е. П. Метод множителей Лагранжа и задачи оптимального распределения ресурсов в системах сетевого планирования.— Автоматика и телемеханика, 1966, № 12, с. 115—125.
10. Рубинштейн М. И., Черкашин А. М. Дискретные задачи оптимального распределения ресурсов в сетевом комплексе операций.— Автоматика и телемеханика, 1980, № 1, с. 140—152.
11. Козлов М. К., Шафранский В. В. Календарное планирование выполнения комплексов работ при заданной динамике поступления складируемых ресурсов.— Известия АН СССР. Технич. кибернетика, 1977, № 4, с. 75—81; 1977, № 51, с. 38—43.
12. Тизен Арне (Thesen Arne). Heuristic Scheduling of Activities under Resource and Precedence Restrictions.— Managemente Science, 1976, 23, № 4, p. 412—422.
13. Бенингтон Г. Е., Мак-Гинис Л. Ф. (Beninghton G. E., Mc. Ginis L. F.) A Critique of Project Planning with constrained Resources.— Lect. Notes Econ. and Math. Syst., 1973, 86, p. 1—28.
14. Корнилова Л. Е. Оптимальное распределение ресурса в сетевом планировании методом оптимального потока.— Кибернетика, 1970, № 6, с. 145—148.

15. Патерсон Дж. Г., Рот Г. В. (Patterson J. H., Roth G. W.) Scheduling a Project under multiple Resource Constraints: a 0—1 Programming Approach.— AIIE Trans., 1976, 8, № 4, p. 449—455.
16. Плещинский Л. С. ε-подход к оптимизации сетевой модели.— Экономика и матем. методы, 1976, XII, № 6, с. 1212—1215.
17. Иванилов Ю. П., Моисеев Н. Н., Петров А. А. Некоторые математические вопросы программного управления экономической системой.— В сб.: Кибернетику на службу коммунизму.— М.: Энергия, 1971, 6, с. 9—22.
18. Поспелов Г. С., Ириков В. А. Программно-целевое планирование и управленпе (введение).— М.: Сов. радио, 1976.— 440 с.
19. Иванилов Ю. П. Программный метод управления и связанные с ним задачи.— М.: МФТИ, 1975.— 255 с.
20. Модели и алгоритмы программного метода планирования сложных систем/Под ред. А. П. Петрова.— М.: ВЦ АН СССР, 1977.— 107 с.
21. Барский А. Б. Планирование параллельных вычислительных процессов.— М.: Машиностроение, 1980.— 192 с.
22. Поспелов Д. А. Введение в теорию вычислительных систем.— М.: Сов. радио, 1972.— 280 с.
23. Липаев В. В. Распределение ресурсов в вычислительных системах.— М.: Статистика, 1979.— 247 с.
24. Евреинов Э. В., Хорошевский В. Г. Однородные вычислительные системы.— Новосибирск: Наука, 1978.— 319 с.
25. Трахтенгерц Э. А. Программное обеспечение автоматизированных систем управления.— М.: Статистика, 1974.— 288 с.
26. Головкин Б. А. Параллельные вычислительные системы.— М.: Наука, 1980.— 520 с.
27. Кофман Е. Г., Денинг Р. Дж. (Coffman E. G. Denning P. J.) Operating System Theory.— Englewood Cliffs: Prentice-Hall, 1973, p. 323.
28. ЭВМ и теория расписаний. (Computer and Job Shop Scheduling Theory/Coffman G. F., ed.) — New York: John Wiley, 1976.
29. Гонзалес М. Дж. (Gonzalez M. J.) Deterministic Processor Scheduling.— ACM Computing Surveys, 1977, 9, № 3, p. 173—204.
30. Танаев В. С., Шкурба В. В. Введение в теорию расписаний.— М.: Наука, 1975.— 256 с.

К главе II

1. Кук С. А. Сложность процедур вывода теорем.— Кибернетический сборник. Новая серия, 1975, вып. 12 с. 5—15.
2. Карп Р. М. Сводимость комбинаторных проблем.— Кибернетический сборник. Новая серия, 1975, вып. 12, с. 16—38.
3. Кнут Д. Искусство программирования для ЭВМ. 1. Основные алгоритмы.— М.: Мир, 1976.— 735 с.
4. Вайд В. (Weide B.) A Survey of Analysis Technique for discrete Algorithms.— ACM Computing Surveys, Dec. 1977, 9, № 4, p. 291—314.
5. Тарьян Р. Е. (Tarjan R. E.) Complexity of combinatorial Algorithms.— SIAM Review, 1978, № 3, p. 457—491.
6. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов.— М.: Мир, 1979.— 535 с.

7. Ульман Дж. Д. (Ullman J. D.) Complexity of sequencing Problems.— Computer and Job Shop Theory/Coffman G. F., ed.— New York: John Wiley, 1976, p. 139—164.
8. Ульман Дж. Д. (Ullman J. D.) Polynomial complete Scheduling Algorithms.— J. of Computer and System Science, Jan. 1975, **10**, № 3, p. 384—393.
9. Гэри М. Р., Джонсон Р. С., Сети Р. (Garey M. R., Johnson R. S., Sethi Ravi). The complexity of Flow-Shop and Job-Shop Problem.— Math. Oper. Res., 1976, № 2, p. 117—129.
10. Гэри М. Р., Джонсон Р. С. (Garey M. R., Johnson R. S.) Complexity Results for Multiprocessor Scheduling under Resourse Constraints.— SIAM Computing, Dec. 1975, 4, № 4, p. 397—411.
11. Узбеков М. У. Некоторые универсальные задачи перебора.— Матем. программирование и смежные вопросы. Труды 7-й Зимней школы, Дрогобыч, 1974.— М., 1976, с. 145—150.
12. Гэри М. Р., Джонсон Д. С. Вычислительные машины и труднорешаемые задачи.— М.: Мир, 1982.— 416 с.
13. Брукер П. и др. (Brucker P., Lenstra J. K., Rinnooy Kan A. H. G.) Complexity of Machine Sequencing Problems.— Math. Cent. Afd. Math. Beslisk. B. W., 1975, № 43, p. 1—29.

К главе III

1. Гэри М. Р., Джонсон Д. С. Вычислительные машины и труднорешаемые задачи.— М.: Мир, 1982.— 416 с.
2. ЭВМ и теория расписаний (Computers and Job. Shop Scheduling Theory/Coffman E. G., ed.).— New Pork: John Wiley, 1976.
3. Джонсон Д. С. (Johnson D. S.) Approximation Algorithms for Combinatorial Problems.— J. of Computer and System Sciences, 1974, № 3, p. 256—278.
4. Гимади Э. Х., Глебов Н. И., Перепелица В. А. Алгоритмы с оценками для задач дискретной оптимизации.— Проблемы кибернетики, 1975, вып. 31, с. 35—42.
5. Гэри М. Р., Джонсон Д. С. (Garey M. R., Johnson D. S.) Approximation Algorithms for combinatorial Problems: an Annotated Bibliography, in: Algorithms and Complexity: New Directions and Recent Results/J. F. Traub, ed.— New York: Academic Press, 1976, p. 41—52.
6. Гэри М. Р., Грэхем Р. Л., Джонсон Д. С. (Garey M. R., Graham R. L., Johnson D. S.) Performance Guarantees for Scheduling Algorithms.— Operations Research, Jan., Febr. 1978, 26, № 1, p. 3—21.
7. Грэхем Р. Л. (Graham R. L.) Bounds for certain Multiprocessing Anomalies.— Bell. Syst. Tech. J., Nov. 1966, p. 1563—1581.
8. Маначер Г. К. (Manacher G. K.). Production and Stabilisation of Real — Time Task Schedules.— Journ. of ACM, July 1967, 14, № 3, p. 429—465.
9. Грэхем Р. Л. (Graham R. L.) Bounds on Multiprocessing Timing Anomalies.— SIAM J. Appl. Math., 1969. 17, p. 263—269.
10. Трусов С. С. Об эффективности простых диспетчеров.— Известия АН СССР. Технич. кибернетика, 1973, № 4, с. 150—160.
11. Кофман М. Т. (Kaufman M. T.). An Almost-Optimal Algorithm for the Assembly Line Scheduling Problem.— IEEE Trans., 1977, **C-23**, № 11, p. 1169—1174.

12. Гэри М. Р. и др. (Garey M. R., Graham R. L., Johnson D. S., Yao A. C.-C.) Resource Constrained Scheduling as Generalized Bin Packing — J. Combinatorial Theory (Series A), 1976, 21, p. 257—298.
13. Кукса А. И. Верхняя оценка длительности кратчайших расписаний в одном классе задач Т. С. Ху.— Кибернетика, 1979, № 3, с. 107—108.
14. Гэри М. Р., Грэхем Р. Л. (Garey M. R., Graham R. L.) Bounds for Multiprocessor Scheduling with Resource Constraints.— SIAM J. Comput., 1975, 4, p. 187—200.
15. Яо А. (Yao A. C.-C.) Scheduling Unit-Time Tasks with Limited Resources.— Lecture Notes Comput. Sci., 1975, 24, p. 17—36.
16. Гимади Э. Х., Глебов Н. И., Перепелица В. А. Исследования по теории расписаний. Управляемые системы.— Новосибирск: ИМ СО АН СССР, вып. 12, 1974, с. 3—10.
17. Лившиц Э. М. Исследование некоторых алгоритмов оптимизации сетевых моделей.— Экономико-математические модели. М., 1968, т. IV, вып. 5, с. 768—775.
18. Португал В. М. Исследование асимптотического поведения решений задачи Джонсона.— Кибернетика, 1971, № 1, с. 105—107.
19. Гимади Э. Х., Пузынина Н. М., Севастьянов С. В. О некоторых экстремальных задачах реализации крупных проектов типа БАМ.— Экономика и математические методы, 1979, т. XV, № 5, с. 1017—1020.
20. Белов И. С., Столин Я. Н. Алгоритм в одномаршрутной задаче календарного планирования.— Математическая экономика и функциональный анализ. М.: Наука, 1974, с. 248—257.
21. Севастьянов С. В. Об асимптотическом подходе к некоторым задачам теории расписаний.— Управляемые системы. Новосибирск: ИМ СО АН СССР, 1975, вып. 14, с. 40—51.
22. Белов И. С., Столин Я. Н. Об алгоритмической сложности леммы Штейница.— Численные методы нелинейного программирования (Тезисы II Всесоюзного семинара, 31 мая — 3 июня 1976 г.). Харьков, 1976, с. 74—77.
23. Кадец М. И. Об одном семействе векторных ломаных в пространстве n измерений.— УМН, 1953, 8, № 1, с. 139—143.
24. Гринберг В. С., Севастьянов С. В. О величине константы Штейница.— Функц. анализ и его приложения, 1980, 14, вып. 2, с. 56—57.

К главе IV

1. Дэвис Э. В., Хидорн Г. Е. (Davis E. D., Heidorn G. E.) An algorithm for Optimal Projekt Scheduling under Multiple Resource Constraints.— Manag. Sci. Application, Aug. 1971, 17, № 12, p. 803—816.
2. Иванин В. М., Кукса А. И. Верхняя оценка числа правильных подмножеств частично-упорядоченного множества.— Оптимизация стохастических систем. Киев: ИК АН УССР, препринт 76-34, 1976, с. 50—53.
3. Иванин В. М., Кукса А. И. Вычисление математического ожидания числа правильных подмножеств в ациклическом ориентированном графе со случайными дугами.— Теория оптимальных решений. Киев: ИК АН УССР, 1977, с. 41—45.

4. Гутьяр А. Л., Немхаузер Г. Л. (Gutjahr A. L., Nemhauser G. L.) An Algorithm for the Line-Balancing Problem.— Manag. Sci., Nov. 1964, **11**, № 2, p. 308—315.
5. Хелд М., Карп М. (Held M., Karp M.) Assembly Line Balancing — Dinamic Programming with Precedince Constraints.— Operations Research, 1963, **12**, № 3, p. 442—460.
6. Такаши Кобаяши (Tokashi Kobayahi). A Method for Counting the Number of Feasible Subsets of a Partially ordered finite Set.— Journ. of the Oper. Res. Soc. of Japan, 1966, 8, № 4, p. 155—171.
7. Кукса А. И., Лаптин Ю. П. Динамическое программирование в сетевой задаче теории расписаний.— Кибернетика, Киев, 1978, № 1, с. 111—113.
8. Бэйкер К., Шрэйдж Л. Е. (Baker K., Schrage L. E.) Einding of Optimal Sequence by Dinamic Programming: an Extention to Precedence related Tasks.— Oper. Res., 1978, 26, № 1, p. 111—120.
9. Риордан Дж. Введение в комбинаторный анализ.— М.: Наука, 1959.— 182 с.

К главе V

1. Михалевич В. С., Шор Н. З. Метод последовательного анализа вариантов при решении вариационных задач управления, планирования и проектирования.— Доклады на IV Всесоюзном математ. съезде, Л., 1961, с. 91.
2. Михалевич В. С., Шор Н. З. Численное решение многовариантных задач по методу последовательного анализа вариантов. Научно-метод. материалы экон.-матем. семинара.— М.: ЛЭММ АН СССР, 1962, вып. 1, с. 15—42.
3. Михалевич В. С. Последовательные алгоритмы оптимизации и их применение. I, II.— Кибернетика, 1965, № 1, с. 45—56; № 2, с. 85—89.
4. Шор Н. З. О структуре алгоритмов численного решения задач оптимального планирования и проектирования: Автореф. дисс. канд.— Киев: ИК АН УССР, 1964.— 135 с.
5. Вычислительные методы выбора оптимальных проектных решений/Под ред. В. С. Михалевича.— Киев: Наукова думка, 1977.— 178 с.
6. Михалевич В. С., Шкурба В. В. Последовательные схемы оптимизации в задачах упорядочения выполнения работ.— Кибернетика, 1966, № 2, с. 34—40.
7. Шкурба В. В. и др. Задачи календарного планирования и методы их решения.— Киев: Наукова думка, 1966.— 155 с.
8. Кукса А. И. Задачи упорядочения на графах.— В кн.: Алгоритмизация производственных процессов.— Киев: ИК АН УССР, 1968, вып. 1, с. 56—71.
9. Голиков А. И., Деменчук В. М. Оптимальное распределение ресурсов в планировании комплексов операций.— Автоматика и телеханика, 1969, № 2, с. 60—70.
10. Рамамурти С. В., Чэнди К. М., Гонзалес Ж. Дж. (Ramamorthy C. V., Chandy K.M., Gonzales M.J.) Optimal Scheduling Strategies in a Multiprocessor System.— IEEE Comput., 1972, **C-21**, p. 137—146.
11. Шрэйдж Л. (Schrage L.) Solving Resource Constrained Network Problems by implicit Enumeration: a nonpreemtive Case.— Oper. Res., 1970, **18**, № 2, p. 262—278.

12. Сахни С. (Sahni S.) Algorithms for Scheduling independent Tasks.— Journ. of ACM, 1976, 23, № 1, p. 116—127.
13. Кукса А. И., Шор Н. З. О методе оценки количества условно-оптимальных вариантов дискретного сепарабельного динамического программирования.— Кибернетика, 1972, № 6, с. 37—44.
14. Иванин В. М. Асимптотическая оценка математического ожидания числа элементов множества Парето.— Кибернетика, 1975, № 1, с. 97—101.
15. Градштейн И. С., Рыжик И. М. Таблицы интегралов, сумм, рядов и произведений.— М.: Наука, 1971.— 1108 с.
16. Емеличев В. А., Комлик В. И. Метод последовательности планов для решения задач дискретной оптимизации.— М.: Наука, 1981.— 208 с.
17. Уздемир А. П. Динамическое размещение производств. I, II.— Автоматика и телемеханика, 1979, № 11, с. 142—151, № 12, с. 146—158.

К главе VI

1. Джеоффрион А. М., Марстен Р. Е. (Geoffrion A. M., Marsten R. E.) Integer Programming Algorithms: A Framework and State of of Art Survey.— Manag. Sci., 1972, 18, № 9, p. 465—491.
2. Корбут А. А., Сигал И. Х., Финкельшейн Ю. Ю. Метод ветвей и границ (обзор теории, алгоритмов, программ и приложений).— Math. Operationsforsch. Statist., Ser. Optimization, 1977, 8, № 2, s. 253—280.
3. Шапиро Дж. Ф. (Schapiro J. F.) A Survay of Lagrangian Techniques for Discrete Optimization.— Annals of Discrete Mathematics 5. Discrete Optimization II.— North-Holland Publ. Company, 1979, p. 113—138.
4. Фишер М. Л. (Fischer M. L.) Optimal Solution of Scheduling Problems using Lagrange Multipliers, Part I, II.— Oper. Res., 1973, v. 21, № 5, p. 1114—1127; Lect. Notes Econ. and Math Syst., 1973, 86, p. 294—318.
5. Демин В. К., Чеботарев А. С. Оптимальное обслуживание детерминированного потока требований, I, II.— Известия АН СССР. Техническая кибернетика, 1976, № 4, с. 193—199; № 5, с. 65—70.
6. Лебедев С. С. Целочисленное программирование и множители Лагранжа.— Экономика и математические методы, 1974, № 3, с. 512.
7. Уздемир А. П., Шмелев В. В. Целочисленные динамические задачи экономического планирования с сетевыми ограничениями.— Автоматика и телемеханика, 1978, № 7, с. 106—115; № 9, с. 110—120.
8. Кукса А. И. К использованию теорий двойственности для решения сетевых задач теории расписаний.— ЖВМ и МФ, 1982, № 5.
9. Кукса А. И., Лаптин Ю. П. К использованию теории двойственности в задачах календарного планирования.— Известия АН СССР. Техническая кибернетика, 1981, № 6.
10. Фишер М. Л. (Fisher M. L.) A dual Algorithm for one Machine Scheduling Problem.— Mathematical Programming, 1976, 11, № 3, p. 229—251.

11. Шор Н. З. Об одном подходе к задачам многотемного планирования.— I Всесоюзная конф. по математическим вопросам сетевого планирования и управления. Тезисы докладов. Киев: ИК АН УССР, 1967, с. 94—101.
12. Залум В. (Zaloom V.) On Resource Constrained Projekt Scheduling Problem.— AIEE Trans., 1971, 3, № 4.
13. Кукса А. И. Оценки длительности кратчайших расписаний в задаче сетевого планирования с нескладируемыми ресурсами.— Теория оптимальных решений. Киев: ИК АН УССР, 1975, с. 24—34.
14. Кукса А. И., Сафронов И. А. Об одном методе оценки уровня ресурсов, необходимого для выполнения частично-упорядоченной совокупности операций в заданное время.— Кибернетика, 1978, № 3, с. 126—130.
15. Барский А. Б. Минимизация числа вычислителей при реализации вычислительного процесса в заданное время.— Известия АН СССР. Технич. кибернетика, 1968, № 6, с. 69—74.
16. Фернандес Е. Б., Бассел Б. (Fernandez E. B., Bussel B.) Bounds on the Number of Processor and Time for Multiprocessor Optimal Schedules.— IEEE Trans., 1973, C-22, № 8, p. 745—751.
17. Ланг Т., Фернандес Е. Б. (Lang T. Fernandez E. B.) Improving the Computation of Lower Bounds for Optimal Schedules.— IBM of Research and Development, May 1977, 21, № 3, p. 273—280.
18. Кукса А. И. Сравнение нижних границ длительностей детерминированных мультипроцессорных расписаний.— Кибернетика, 1979, № 5, с. 87—90.
19. Лаптин Ю. П. Сравнение некоторых методов вычисления нижних оценок длительности кратчайших расписаний.— Кибернетика, 1978, № 5, с. 74—78.
20. Кукса А. И., Поляченко Б. Е. Математические методы решения сетевых задач календарного планирования.— Кибернетика, 1981, № 6.
21. Раевич С. К. Применение метода ветвей и границ для решения одной задачи распределения ресурсов.— Математические методы решения экономических задач. М.: Наука, 1969, I, с. 114—126.
22. Шахбазян К. В., Тушкина Т. А., Метод ветвей и границ для задачи параллельного упорядочения.— Записки научн. семин. ЛО МИ АН СССР, 1973, 35, с. 146—155.
23. Плещинский А. С. ε-подход к оптимизации сетевой модели.— Экономика и математические методы, 1976, XII, № 6, с. 1212—1215.
24. Форд Л., Фалкерсон Д. Потоки в сетях.— М.: Мир, 1966.— 274 с.
25. Демьянов В. Ф., Малоземов В. Н. Введение в минимакс.— М.: Наука, 1972, 368 с.
26. Шор Н. З. Методы минимизации недифференцируемых функций и их приложения.— К.: Наукова думка, 1979, с. 200.
27. Джеоффрион А. М. (Geoffrion A. M.) An improved implicit enumeration approach for integer programming.— Operations Res., 1969, 17, p. 437—454.
28. Косарев Н. Г., Уздемир А. П. Динамическая задача планирования научных исследований и разработок и ме-

тод ее решения.— Автоматика и телемеханика, 1977, № 1, с. 62—73.
29. Михалевич В. С., Шор Н. З. Метод последовательного анализа вариантов при решении вариационных задач управления, планирования и проектирования.— Докл. на Всесоюзном матем. съезде. Л., 1961, с. 91.
30. Михалевич В. С., Шор Н. З. Численное решение многовариантных задач по методу последовательного анализа вариантов.— Науч.-метод. материалы экон.-матем. семинара. М.: ЛЭММ АН СССР, 1962, вып. 1, с. 15—42.

Владимир Сергеевич Михалевич,
Анатолий Иванович Кукса

МЕТОДЫ ПОСЛЕДОВАТЕЛЬНОЙ ОПТИМИЗАЦИИ В ДИСКРЕТНЫХ СЕТЕВЫХ ЗАДАЧАХ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ

(Серия: «Экономико-математическая библиотека»)

Редактор А. Д. Вайнштейн
Техн. редактор Е. В. Морозова
Корректоры Г. В. Подвольская, Л. С. Сомова

ИБ № 11746

Сдано в набор 21.05.82. Подписано к печати 11.01.83. Т-02908. Формат 84×108$^{1}/_{32}$. Бумага тип. № 2. Обыкновенная гарнитура. Высокая печать. Услови. печ. л. 10,92. Уч.-изд. л. 11,63. Тираж 7500 экз. Заказ № 191. Цена 1 р. 40 к.

Издательство «Наука»
Главная редакция физико-математической литературы
117071, Москва, В-71, Ленинский проспект, 15

4-я типография издательства «Наука»
630077, Новосибирск, 77, Станиславского, 25