

Московский государственный университет им. М.В.Ломоносова
Экономический факультет

Левин Марк Имуилович

НЕКОТОРЫЕ ДИСКРЕТНЫЕ ОПТИМИЗАЦИОННЫЕ ЗАДАЧИ
РАСПРЕДЕЛЕНИЯ РЕСУРСОВ В ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

05.13.02 "Теория систем, теория автоматического
регулирования и управления, системный анализ"

Представляется на соискание ученой степени кандидата
технических наук

198/3
Научный руководитель
доктор технических наук, профессор
Юдин Д.Б.

Москва, 1981 г.

ОГЛАВЛЕНИЕ

	стр.
Введение	4
Глава 1. Проблема рационального распределения ресурсов в вычислительных системах и задачи дискретной оптимизации	9
1.1 Экономическая эффективность использования вычислительной техники	9
1.2 Использование дискретных оптимизационных моделей распределения ресурсов в вычисли- тельных системах	12
1.3 Эффективные комбинаторные методы решения дискретных оптимизационных задач	16
Выводы к главе 1	22
Глава 2. Эффективные методы детерминированного плани- рования с учетом ограничений предшествования	23
2.1 Детерминированные задачи составления рас- писаний при организации вычислительного процесса в случае одновременного поступ- ления заявок	25
2.2 Постановка задач, определения, эквивалентные работы	27
2.3 Преобразование задач	34
2.4 Алгоритмы упорядочения	38
2.5 Примеры составления расписаний	41
2.6 Анализ характеристик одного семейства близких по постановке задач	48
Выводы к главе 2	52

Глава 3. Экстремальная модель организации данных при ограничении на общий объем используемой оперативной памяти	54
3.1 Постановка задачи	55
3.2 Простейшие частные случаи	60
3.3 Вспомогательные задачи	65
3.4 Древовидная связь между данными	70
3.5 Некоторые частные случаи компоновки при одинаковых характеристиках модулей	73
3.6 Задача на ациклическом направленном графе	76
Выводы к главе 3	79
Глава 4. Распределение вычислительных ресурсов на примере подсистемы АСУ-Ростгеология	80
4.1 Назначение подсистемы	80
4.2 Структура информационной базы	82
4.3 Состав программного комплекса	85
4.4 Выбор варианта структуры программного комплекса	88
Выводы к главе 4	102
Приложение I	103
Приложение 2	111
Заключение	115
Литература	117

Введение

Одним из важнейших задач, возникающих в условиях широкого применения вычислительной техники, является проблема ее рационального использования. Сложность определения экономического эффекта, связанного с применением вычислительной техники, обуславливает необходимость исследования различных частных задач, решение которых направлено на повышение эффективности использования ЭВМ. Одной из важнейших задач является проблема совершенствования вычислительной техники, программного обеспечения, снижения затрат при эксплуатации вычислительных комплексов [1, с.144]. В рамках этой проблемы в последнее время проводятся исследования по экономному использованию основных ресурсов вычислительных систем: производительности и памяти.

Число публикации, в которых детально рассматриваются различные вопросы распределения ресурсов в вычислительных системах, постоянно растет. Наиболее значительные из них появились в последние годы [2, 31, 33]. Обычно исследуются следующие основные типы задач рационального распределения ресурсов в вычислительных системах [2, 31, 33]:

- управление распределением времени работы компонент вычислительных комплексов;
- организация данных;
- распределение памяти (многоуровневой, одного уровня быстродействия);
- управление обменом с внешними устройствами.

При решении этих задач используются модели теории массового обслуживания, теории вероятностных автоматов и дискретной оптимизации.

зации. Дискретные оптимизационные модели применяются при исследовании следующих задач:

- детерминированные задачи составления расписаний при организации вычислительного процесса;
- организация данных;
- распределение памяти.

Дискретные постановки, например, графовые, являются наглядными и простыми для понимания. Построение таких, даже упрощенных, моделей позволит в значительной степени структуризовать стоящие перед специалистами по разработке и эксплуатации вычислительных комплексов проблемы.

Допустимые затраты на решение задач распределения ресурсов в вычислительных системах не должны превышать эффекта, получаемого от применения соответствующего метода. Это обуславливает требование к эффективности алгоритмов решения дискретных оптимизационных задач распределения ресурсов. Большинство практических задач распределения производительности и памяти ЭВМ являются сложными комбинаторными проблемами. Поэтому является актуальным построение эффективных (трудоемкость которых полиномиально зависит от размерности задачи) точных и приближенных методов решения возникающих проблем; выявление классов задач, допускающих эффективные точные алгоритмы решения, эффективные приближенные методы с гарантированной точностью; разработка и исследование эвристических алгоритмов.

Исследованию проблем построения дискретных оптимизационных моделей распределения ресурсов в вычислительных системах и разработке эффективных методов решения возникающих задач посвящена настоящая работа.

В первой главе рассматриваются подходы к оценке экономической

эффективности использования вычислительной техники, основные типы задач распределения ресурсов в вычислительных системах, исследование которых приводит к дискретным оптимизационным проблемам, и современные подходы к построению эффективных методов решения дискретных оптимизационных задач.

В некоторых системах управления имеют место ситуации, когда основную загрузку вычислительного комплекса составляет конечная совокупность программ и требуется определить порядок их выполнения, обеспечивающий экстремальное значение некоторого критерия, например, минимум суммарного времени выполнения. При этом программы часто бывают информационно связаны, т.е. выходные результаты одних программ являются входными данными для других. Возникает необходимость учета ограничений предшествования на порядок выполнения заявок. Ограничения предшествования значительно усложняют возникающие задачи теории расписаний и для многих таких задач доказано, что они относятся к классу универсальных переборных проблем [69, 74].

Во второй главе рассматривается класс детерминированных задач составления расписаний при организации вычислительного процесса в случае одновременного поступления заявок. Исследование свойств задач этого класса позволило разработать общий эффективный точный метод решения для случая, когда ограничение предшествования представляет собой дерево или параллельно-последовательную сеть. К рассмотренному классу относятся шесть известных задач теории расписаний, в том числе задача минимизации суммарного штрафа в случае одного процессора с линейными или экспоненциальными функциями штрафа и задача Беллмана-Джонсона в случае двух процессоров. В заключении второй главы приведен анализ характеристик семейства близких по постановке задач теории расписаний, который позво-

ляет выявить наиболее перспективные направления возможностей решения задач.

В третьей главе исследуется одна экстремальная задача организации данных на примере задачи выбора структуры программного комплекса (компоновки модулей, входящих в систему). Задача заключается в минимизации среднего числа обменов между оперативной и внешней памятью при ограничении на общий объем оперативной памяти, предоставляемой для работы комплекса. Предполагается, что передача по управлению осуществляется с возвратом в вызывающий модуль и связи между модулями имеют вид дерева. Задача компоновки сводится к задаче выбора стягиваемых дуг во взвешенном ориентированном дереве.

Анализ различных случаев задачи и построение эффективных алгоритмов решения основаны на переходе от простейших постановок ко все более сложным. Даже простейшие частные случаи задачи относятся к классу универсальных переборных проблем, поэтому предлагаются приближенные методы решения с гарантированной относительной потребностью целевой функции. При решении сложных случаев рассматриваемой задачи, включая особый случай древовидной связи, дополнительно допускается гарантированная относительная потребность выполнения ограничения на память. Рассмотренные модели являются обобщением известных задач дискретного программирования о рюкзаке и "лестничном" рюкзаке. Связи между данными часто имеют более сложный вид, чем дерево. В работе показано, что эти случаи на основе эвристических подходов могут быть сведены к рассмотренным.

В четвертой главе рассмотрена подсистема АСУ-Ростология (назначение, структура информационной базы и программное обеспечение).

Проведен анализ функционирования системы на основе предложенного в предыдущей главе подхода к решению задачи компоновки программных модулей.

Результаты исследования автора нашли отражение в статьях [24, 25, 26, 27] ^{1/}.

^{1/} Личный вклад автора диссертации в работы [25] - [27] освещен в приложении II.

Глава I. Проблема рационального распределения ресурсов в вычислительных системах и задачи дискретной оптимизации

В связи с широким использованием ЭВМ во всех отраслях народного хозяйства актуальной является проблема повышения экономической эффективности использования вычислительной техники (ВТ). Сложность определения экономического эффекта вызывает необходимость дифференцированного подхода к оценке экономической эффективности и выделения частных целей, согласованных с проблемой повышения экономической эффективности ВТ в целом. Одной из основных частных целей является рациональное распределение ресурсов в вычислительных системах и исследование возникающих при этом дискретных оптимизационных задач и эффективных методов их решения. В данной главе рассматриваются основы дифференцированного подхода к оценке экономической эффективности использования ВТ, случаи применения дискретных оптимизационных моделей распределения ресурсов в вычислительных системах и основные подходы к построению эффективных комбинаторных методов решения задач дискретной оптимизации. Вопросы, связанные с постановкой некоторых дискретных оптимизационных задач распределения ресурсов в вычислительных системах и построением эффективных методов их решения, рассматриваются в следующих главах.

I.1 Экономическая эффективность использования вычислительной техники.

В самом общем виде экономическую эффективность использования ВТ можно определить как отношение стоимостных результатов использования ВТ к совокупным затратам на их достижение. Основная составляющая результатов использования ВТ (экономического эффекта) —

косвенный экономический эффект, который является следствием повышения экономических показателей функционирования объекта. Определение величины косвенного экономического эффекта обычно затруднено из-за сложности выделения доли повышения экономических показателей, вызываемой применением ВТ [4].

ВТ, с точки зрения ее места в народном хозяйстве, можно рассматривать в двух аспектах. Во-первых, как относительно обособленную систему, и во-вторых, как часть более общей системы - некоторого экономического объекта. Это определяет два подхода к рассмотрению экономической эффективности использования ВТ: эффективность в узком смысле и эффективность в широком смысле. Заметим, что аналогично решается проблема эффективности управления [46].

Экономическая эффективность в узком смысле предполагает рассмотрение ВТ как обособленной системы и определение экономического эффекта в рамках этой системы. Эффект может выражаться, например, в увеличении времени загрузки вычислительного комплекса (ВК) и отдельных его частей, в снижении стоимости использования ВТ. Другим подходом является рассмотрение ВК как составной части экономической системы. В этом случае можно говорить об экономической эффективности в широком смысле и рассматривать результаты использования ВТ у потребителя. Оценкой этого эффекта могут служить различные показатели, например, прирост выпуска продукции, повышение коэффициента использования мощностей [23]. Экономический эффект использования ВТ необходимо также дифференцировать с точки зрения уровня получения его в народном хозяйстве, например, на уровне отрасли, предприятия. Актуальность разделения проектной и фактической экономической эффективности использования ВТ обосновывается в [38]. Определение экономической эффективности применения ВТ существенным образом зависит от конкретного случая использования

БТ. Можно выделить, например, такие варианты применения БТ, как ОАСУ или ВК, используемые для научных исследований и разработок.

Таким образом, приходится к комплексному подходу к проблеме экономической эффективности на основе следующей дифференциации:

- в узком и широком смысле;
- по уровням народного хозяйства,
- проектной и фактической,
- по вариантам использования БТ.

Аналогичный подход к оценке исследований, разработок и внедрения новой техники рассматривается в [45].

В некоторых случаях комплексный подход позволяет проводить оценку экономической эффективности использования БТ, например, для АСУ технологическими процессами. В большинстве случаев исследование экономического эффекта представляет собой сложную задачу, поэтому является целесообразным, в соответствии с методами анализа сложных систем, переход к исследованию частных подзадач, соотносимых с основной целью — повышением экономической эффективности использования БТ.

Одной из важнейших подзадач является проблема снижения затрат, связанных с применением БТ. В этом направлении можно выделить два уровня: глобальный и локальный. Первый заключается в организационных мероприятиях, направленных на централизованное использование технических средств и программного обеспечения. К этому уровню можно отнести промышленную разработку программного обеспечения, создание вычислительных центров коллективного использования и сетей БИМ. Локальный уровень снижения затрат, связанных с использованием БТ, заключается в решении конкретных технических проблем по экономии важнейших ресурсов ВК: производительности и памяти. При этом необходим учет конкретных особенностей проектируе-

ного или эксплуатируемого ВК на основе правильного построения математических моделей. Некоторые методические подходы к рациональному распределению ресурсов в вычислительных системах имеют общий характер. Следует отметить, что особенностью автоматизированных систем на базе ЭВМ является постоянное совершенствование. При проектировании это накладывает особые требования к обеспечению адаптивности систем, например, открытость и модульность построения программного обеспечения.

1.2. Использование дискретных оптимизационных моделей распределения ресурсов в вычислительных системах

В большинстве случаев под задачей распределения ресурсов в вычислительных системах понимается нахождение такого распределения памяти и времени работы компонент ВК между программами, пользователями, процессами, при котором некоторый показатель качества функционирования ВК достигает экстремального значения. При этом предполагается, что структура и компоненты ВК фиксированы [31]. Вычислительные системы представляют собой сложные системы, для которых определение единого критерия качества, как правило, затруднено. В связи с этим рассматриваются частные критерии (время центрального процессора, объем оперативной и внешней памяти, среднее число обменов с внешними устройствами, время пробега магнитных головок чтения-записи в случае использования магнитных дисков), позволяющие оценивать достижение некоторых частных целей, согласованных между собой и не противоречащих основной цели. Обычно исследуются следующие основные типы задач, связанных с распределением ресурсов [2, 31, 33]:

- управление распределением времени работы компонент ВК;
- организация данных;

- распределение памяти многоуровневом, одного уровня быстрой деактивации;
- управление обменом с внешними устройствами.

Особое значение имеют анализ мультипрограммного режима работы ВК, задачи распределения ресурсов в многомашинных и многопроцессорных системах.

Следует заметить, что разбиение задач распределения ресурсов в вычислительных системах условно, так как решение задачи одного класса обычно основано на решении задач другого класса [52].

При решении задач распределения ресурсов в вычислительных системах используются модели теории массового обслуживания, теории вероятностных автоматов и дискретной оптимизации. Выбор типа модели определяется характеристиками поведения программ (пользователей, процессов) [2]. Дискретные оптимизационные модели используются в основном при решении следующих задач:

- детерминированные задачи составления расписаний при организации вычислительного процесса;
- организация данных;
- распределение памяти.

При исследовании задач организации вычислительного процесса имеют место ситуации, например, в автоматизированных системах управления, когда основную нагрузку ВК составляет конечная совокупность программ и требуется определить порядок их выполнения, обеспечивающий экстремальное значение некоторого критерия. При этом заранее известны детерминированные характеристики программ, например: время выполнения, директивные сроки, информационные связи между программами, задающие ограничения на порядок выполнения. В качестве критериев рассматриваются такие, например, как суммарное время выполнения всех программ, средневзвешенная сумма моментов

тов выполнения программ, число программ, невыполненных к директивным срокам [21, 31, 33, 42, 59]. Для анализа подобных задач распределения ресурсов используются дискретные оптимизационные модели теории расписания [21, 42, 58].

Актуальность задачи организации данных значительно возросла в связи с широким применением различных систем обработки информации. От организации хранения данных в значительной степени зависят такие технико-экономические характеристики, как время поиска, стоимость хранения. Решение возникающих при организации данных оптимизационных комбинаторных проблем позволяет существенно повысить эффективность систем обработки информации при незначительных дополнительных затратах. Например, использование вспомогательного поискового массива в информационной системе, рассматриваемой в четвертой главе настоящей работы, позволило сократить время поиска данных с нескольких минут до долей секунды, причем объем дополнительно используемой памяти составил менее 1% от объема памяти, требуемой для хранения основного информационного массива [26]. Классы задач организации данных рассмотрены в [52]. При этом выделены следующие типы задач:

- оптимизация многоступенчатой организации данных;
- оптимизация поисковых деревьев;
- оптимальная организация информационных массивов.

Модели, используемые при решении этих задач, как правило, представляют собой экстремальные проблемы дискретного программирования. Часто оказывается эффективным применение экстремальных постановок на графах. Следует отметить, что в связи со значительной сложностью программного обеспечения актуальны задачи организации данных, в которых в качестве блоков информации рассматриваются программные модули [2, 6, 7, 20, 31, 32, 36].

Задачи распределения памяти вычислительных систем заключаются в размещении программных и информационных модулей в многоуровневой памяти или в памяти одного уровня. В качестве уровней многоуровневой памяти чаще всего рассматриваются оперативная память (ОП), память на магнитных дисках (МД) и на магнитных лентах (МЛ). Задача распределения данных в памяти одного уровня обычно связана с ОП, МД, МЛ. Задачи первого типа заключаются в распределении данных по уровням памяти, а второго - в упорядочении информации, отнесенной к памяти одного уровня.

В общем случае проблема состоит в уменьшении времени обработки данных при использовании ограниченных ресурсов памяти. К наиболее часто используемым критериям можно отнести время поиска, среднее число обращений к внешней памяти, время пробега магнитных головок чтения-записи в случае МД, время перемотки МЛ, число смежных пакетов МД или МЛ.

Так как быстродействия различных уровней памяти отличаются на несколько порядков, можно решать задачи первого типа поэтапно, последовательно выделяя для каждого уровня соответствующие ему объемы информации, начиная с уровня максимального быстродействия [41, 52]. Как правило, при этом получаются формальные постановки целочисленного или линейного программирования [52].

В конце 60-х - начале 70-х годов многие исследования были посвящены размещению информации на МЛ [8, 11]. Полученные при этом результаты являются интересными и полезными как с теоретической, так и с практической точек зрения. Последние годы, в связи с широким использованием МД, усилия большого числа авторов направлены на решение различных задач размещения данных на пакетах МД [2, 14, 15, 26, 31, 32, 37, 39, 44, 62]. Как отмечается в [14], перенос результатов, полученных при решении задач размещения информации на МЛ, на слу-

чая использования МД чаще всего не представляется возможным вследствие специфики пакетов МД.

При решении задач размещения информации на МД часто рассматриваются упрощенные модели, в рамках которых проводится упорядочение равных по объему блоков данных в пределах одного пакета [8,14,44]. Даже в этих случаях формальные постановки представляют собой сложные комбинаторные проблемы, например, в [14]. Простейшие задачи этого типа иногда сводятся к моделям теории расписаний [2,8,11]. В реальных вычислительных системах, как правило, массивы имеют различные размеры и одновременно обрабатываются данные, размещенные на нескольких пакетах [2], и это усложняет модели. Сложность задач размещения данных растет также при построении моделей, более адекватно учитывающих стохастический характер обращения к информации и возможность совместной обработки массивов [1,14,62]. Как и при решении задач организации данных, при размещении информации на МД рационально использовать экстремальные постановки на графах [14].

Постановка дискретных оптимизационных задач распределения ресурсов в вычислительных системах полезна при проектировании программных комплексов, систем обработки данных. Дискретные постановки, например, графовые, являются наглядными и простыми для понимания. Поэтому построение, даже упрощенных, моделей позволяет в значительной степени структуризовать стоящие перед разработчиками проблемы.

1.3 Эффективные комбинаторные методы решения дискретных оптимизационных задач

Допустимые затраты на решение задач распределения ресурсов в вычислительных системах не должны превышать эффекта, получаемого от применения соответствующего метода. Это обуславливает требования к эффективности алгоритмов решения дискретных оптимизационных

задач распределения ресурсов. Эффективными называют алгоритмы, трудоемкость и память которых является полиномом от размерности задачи [3, 29, 40, 43]. (Размерность задачи - это длина двоичной записи всех исходных данных.) Исследования, проводимые в рамках теории сложности, показали, что большое число трудно решаемых дискретных задач, для которых не были найдены эффективные алгоритмы, эквивалентны с точки зрения их вычислительной сложности. Эти задачи получили название универсальных или полиномиально-полных проблем [3, 29, 40, 43]. В неформальных терминах это означает, что либо все эти задачи неразрешимы за полиномиальное время, либо, если хотя бы одну из них можно разрешить за полиномиальное время, то любая универсальная задача может быть решена за полиномиальное время. Широкое распространение имеет гипотеза, что задачи неразрешимы.

Большое теоретическое и практическое значение имеет перевод определенных классов задач на более низкий уровень сложности иерархии. В [40] такое направление определено как глобальный уровень повышения эффективности методов решения дискретных задач. В рамках этого направления можно выделить следующие два подхода. Во-первых, построение эффективных точных методов решения, если это возможно. Например, задача Беллмана-Джонсона в случае двух машин при ограничении предшествования на порядок выполнения работ в виде дерева. некоторыми исследователями относилась к классу полиномиально-полных проблем [57, 68]. В настоящей работе [27] для этой задачи построен алгоритм с трудоемкостью $O(n \log n)$, где n - число работ, причем достаточные условия применимости метода позволили распространить его на целый класс задач теории расписаний [24] (см. главу 2). Во-вторых, для некоторых задач, принадлежащих к классу универсальных, возможно построение эффективных алгоритмов (эффективных приближенных), позволяющих находить приближенное реше-

ние с определенным образом гарантированной точностью [9,28,29,47, 48,61,71].

Другим важным направлением построения эффективных алгоритмов является построение методов решения дискретных задач, позволяющих находить оптимальное или "почти оптимальное" решение в подавляющем большинстве случаев [10]. Такие алгоритмы называются статистически эффективными [40,47,48].

Рассмотрим подробнее эффективные приближенные алгоритмы.

Определение I.1 [29]. Алгоритм называется ε -приближенным (или соответственно ε_a -приближенным) для массовой экстремальной задачи P , если он для любой индивидуальной задачи I_P вырабатывает решение $\bar{x}$ такое, что $|f(I_P, \bar{x}) - f^*(I_P)| \leq \varepsilon f^*(I_P)$ или соответственно $|f(I_P, \bar{x}) - f^*(I_P)| \leq \varepsilon_a$, где $f^*(I_P)$ - оптимальное значение целевой функции, $f(I_P, \bar{x})$ - значение целевой функции, вырабатываемое алгоритмом.

Определение I.2 [37,53]. Алгоритм называется ε_r -приближенным для массовой экстремальной задачи P , если он для любой заданной индивидуальной задачи I_P вырабатывает решение $\bar{x}$ такое, что

$$\frac{f(I_P, \bar{x}) - f^*(I_P)}{\max_{x \in X} f(I_P, x) - \min_{x \in X} f(I_P, x)} \leq \varepsilon_r.$$

Определение I.3 [29]. Приближенный алгоритм называется полиномиальным или эффективным, если при фиксированном ε его трудоемкость и память оцениваются сверху степенными функциями от размерности задачи.

Определение I.4 [29]. Полиномиальный приближенный алгоритм называется быстрым, если его трудоемкость и память оцениваются сверху полиномом относительно размерности задачи и величины $1/\varepsilon$.

Определения I.3 и I.4 очевидным образом распространяются на случай, когда точность решения задается при помощи ε_a или ε_r .

Приближенные эффективные алгоритмы (в том числе и быстрые) для решения различных дискретных оптимизационных задач рассмотрены в [29]. В третьей главе настоящей работы предложены быстрые эффективные алгоритмы для решения некоторых вариантов задачи компоновки программных модулей.

Следует отметить, что содержащаяся в приведенных определениях 1.1 и 1.2 плата (точность целевой функции) может оказаться недостаточной для получения алгоритма, имеющего полиномиальную сложность по размерности задачи и по $\frac{1}{\varepsilon} \left(\frac{1}{\varepsilon_a} \cdot \frac{1}{\varepsilon_v} \right)$. Рассмотрим условную задачу максимизации некоторой функции $\varphi(x)$, когда одно из условий имеет вид $\varphi(x) \leq \varphi$, где $\varphi(x)$ - функция от x , φ - константа.

Определение 1.5 Алгоритм называется (ε, δ) приближенным для массовой экстремальной задачи P , если для любой индивидуальной задачи I_p он вырабатывает решение $\bar{x}$ такое, что $\alpha(I_p, \bar{x}) \leq \varepsilon$ и $\beta(I_p, \bar{x}) \leq \delta$, где

$$\alpha(I_p, \bar{x}) = \begin{cases} 0, & \text{если } f(I_p, \bar{x}) \geq f^*(I_p), \\ \frac{f^*(I_p) - f(I_p, \bar{x})}{f^*(I_p)}, & \text{если } f(I_p, \bar{x}) < f^*(I_p), \end{cases}$$

$$\beta(I_p, \bar{x}) = \begin{cases} 0, & \text{если } \varphi(I_p, \bar{x}) \leq \varphi(I_p), \\ \frac{\varphi(I_p) - \varphi(I_p, \bar{x})}{\varphi(I_p)}, & \text{если } \varphi(I_p, \bar{x}) > \varphi(I_p), \end{cases}$$

$$f^*(I_p) \neq 0, \varphi(I_p) \neq 0.$$

$f(I_p)$ - правая часть условия в задаче I_p , $\varphi(I_p, \bar{x})$ - левая часть условия при решении, вырабатываемом алгоритмом.

Очевидно, что определение 1.5 может быть модифицировано для других случаев, например: целевая функция минимизируется, ограничение имеет обратный знак, точность решения задается другим способом (при помощи $\varepsilon_a, \varepsilon_v$). Из определения 1.5 следует, что получаемое решение является ε - приближенным относительно оптималь-

ного решения при правой части условия ψ . Но в подавляющем большинстве случаев найденное решение является ε - приближенным и относительно оптимального при правой части условия, равной $(1+\delta)\psi$. Алгоритм, соответствующий определению 1.5 предлагается в главе 3 для решения некоторых вариантов одной экстремальной задачи организации данных. Оценка погрешности решения из определения 1.5 соответствует введенному в [37] вектору относительных погрешностей.

В [40] выделен также локальный уровень повышения эффективности методов решения дискретных задач. При этом под локальным уровнем понимаются "любые приемы, позволяющие расширить имеющиеся на сегодняшний день возможности фактического решения задач" [40, с.295]. Вследствие сложности моделей, возникающих при решении задач распределения ресурсов в вычислительных системах, часто применяются методы, опирающиеся на некоторые правдоподобные, формально не обоснованные соображения, связанные с конкретной особенностью структуры задач - эвристические алгоритмы [22, 40, 47]. Одному из классов эвристических, так называемым "пожирающим" алгоритмам (greedy), последние годы уделяется большое внимание. Например, в задаче коммивояжера такой алгоритм действует по правилу "идти к ближайшему". Оценки трудоемкости и памяти "пожирающих" алгоритмов полиномиальны.

При решении задач распределения ресурсов в вычислительных системах применяются различные методы решения дискретных оптимизационных задач. Например, в [51] предложен приближенный метод организации многоступенчатой памяти, в [14] - метод локальной оптимизации для решения задачи размещения блоков данных на пакете МД, в [39] - метод случайного поиска для аналогичной задачи. Использование методов неявного перебора применительно к задачам организации памяти содержится в [35]. Широкое практическое использование имеют

различные эвристические методы при решении задач распределения времени центрального процессора, организации обмена с внешними устройствами, организации вычислительного процесса при мультипрограммировании, в многопроцессорных и многомашинных ВК [2, 31, 33].

В настоящее время представляются важными исследования, основанные на современных подходах к проблемам дискретной оптимизации, для решения задач распределения ресурсов в вычислительных системах по следующим направлениям:

- построение эффективных точных методов,
- построение эффективных приближенных (особенно быстрых) алгоритмов,
- построение и анализ эвристических методов.

Выводы к главе I.

1. Трудности, возникающие при определении косвенного экономического эффекта, обуславливают актуальность исследования проблемы снижения затрат, связанных с использованием ВТ, и, в частности, затрат на разработку и эксплуатацию программного обеспечения. Уменьшение этих затрат связано с решением задач распределения основных ресурсов вычислительных комплексов: памяти и производительности.

2. При исследовании задач организации вычислительного процесса, организации данных и распределения памяти возникают дискретные оптимизационные модели.

3. При решении дискретных оптимизационных задач распределения ресурсов в вычислительных системах актуальным является построение эффективных методов:

- точных,
- приближенных, в том числе быстрых (полиномиальных от размерности и величин, обратно пропорциональных точности),

- 22 -

- эвристических (и анализ этих алгоритмов).

Глава 2. Эффективные методы детерминированного планирования с учетом ограничений предшествования

При исследовании задач распределения времени работы компонент ВК системы организации вычислительного процесса классифицируются с точки зрения неопределенности априорной информации об основных временных характеристиках потоков заявок и их обслуживания [33]. В работе [33, с.143] выделены следующие три типа систем:

- полностью стохастические, когда неизвестны априорные характеристики потоков, позволяющие их классифицировать и разделить;
- использующие априорную информацию о стохастических характеристиках различных потоков;
- детерминированные или квазидетерминированные, использующие достоверную информацию о каждой заявке.

В данной главе рассматриваются методы построения расписания обслуживания заявок для некоторых систем третьего типа при наличии ограничения предшествования на порядок выполнения заданий. Простые правила составления оптимальных расписаний для случая детерминированного распределения производительности вычислительных систем могут быть применены даже при выполнении не всех ограничений, при которых они оптимальны, в качестве первого приближения [31].

При составлении расписаний в ВК типичной является невысокая размерность задач, когда упорядочению подлежат 30-50 программ [31, 33]. Использование методов явного перебора, даже в таких случаях, может привести к значительным затратам на упорядочение. Поэтому создание эвристических методов упорядочивания программ в процессе составления оптимального расписания и точных решающих правил оптимизации является одной из основных проблем, возникаю-

щих при анализе и синтезе систем организации вычислительного процесса методами детерминированного планирования [33].

В тех случаях, когда программы информационно связаны, т.е. выходные результаты одних программ являются входными данными для других, возникает необходимость учета ограничений предшествования на порядок выполнения заявок. Такие ограничения значительно усложняют задачи теории расписаний и для многих из них доказано, что они относятся к классу универсальных переборных задач [69,74]. Поэтому весьма важным является выделение классов задач детерминированного распределения производительности вычислительных комплексов с учетом ограничений предшествования, для которых возможно построение эффективных точных методов решения. В данной главе приведены условия, при выполнении которых для целого класса задач теории расписаний построен общий полиномиальный точный алгоритм (при ограничении предшествования в виде совокупности деревьев или параллельно-последовательной сети). Приведены примеры решения различных задач. Продемонстрировано, что предложенный подход позволяет решать задачи рассматриваемого класса и на некоторых сетях, более сложных, чем параллельно-последовательные. Следует заметить, что для детерминированной задачи теории расписаний в случае одного процессора и линейных функций штрафа эффективный точный алгоритм был построен в 1972 году Хорном [60]. В 1973 году этот результат был распространен Гордоном и Танаевым на случай экспоненциальных функций штрафа [13]. В последние годы (в основном 1977-80 гг.) многие исследователи (Адольфсон, Лоулер и Сивазлиан, Монма и Сидней, автор) независимо распространили результат Хорна на случаи параллельно-последовательных сетей [24,55,67,70]. Как уже было сказано, автором предложен общий подход к решению целого класса задач, включающего, кроме упомянутых, задачу Беллмана-Джонсона в случае двух про-

процессоров при одинаковом ограничении предшествования на обоих процессорах и три задачи для одного процессора с функциями штрафа, имеющими несколько более сложный вид, чем в задачах, рассмотренных Хорнсом и Гордоном, Танаевым. Заметим, что этот результат автора также в некоторой степени пересекается с результатами других исследователей. Например, указанный вариант задачи Беллмана-Джонсона изучал Курису [63-65] и построенные им алгоритмы, как показал их анализ, позволяют эффективно получать оптимальное расписание при двувидном ограничении предшествования. Монма и Сидней распространили этот результат на случай параллельно-последовательных сетей [70].

В заключении главы проведен анализ характеристик одного семейства близких по постановке задач, который позволяет выявлять актуальные проблемы для дальнейших исследований.

2.1. Детерминированные задачи составления расписаний при организации вычислительного процесса в случае одновременного поступления заявок

Исследование детерминированных задач составления расписаний при организации вычислительного процесса основывается на предположении, что точно известны состав подлежащих выполнению заданий, длительности их выполнения и другие характеристики. Такие ситуации имеют место в некоторых управляющих системах [2, 34]. При этом задача состоит в определении порядка выполнения программ на одном или нескольких процессорах, который минимизирует некоторые критерии качества.

Рассмотрим некоторые модели теории расписаний для случая, когда все заявки поступают одновременно [21, 42, 58, 68]. Пусть имеется

множество заданий $R = \{I, \dots, n\}$ и множество идентичных процессоров $P = \{P_1, \dots, P_m\}$. Каждое задание $i \in R$ характеризуется временем выполнения t_i ; директивным сроком d_i , к которому необходимо выполнить задание; моментом окончания выполнения C_i ; коэффициентом важности q_i ; функцией штрафа $\phi_i(t)$. В качестве аргумента функции штрафа часто используется C_i . Функция штрафа обычно имеет линейный или экспоненциальный вид, однако возможны и более сложные виды зависимости [21, 31, 33, 42, 58, 68]. Рассмотрим следующие два типа критериев качества составления расписания [68]: $f_1 = \max_{i \in R} \phi_i(C_i)$ и $f_2 = \sum_{i \in R} \phi_i(C_i)$. В первом случае минимизируется максимальное значение функции штрафа, во втором — суммарное (или среднее). В качестве критериев далее в главе 2 будут рассматриваться $C_{\max} = \max_{i \in R} C_i$, $\sum_{i \in R} C_i$, $\sum_{i \in R} q_i C_i$, $\sum_{i \in R} q_i \phi_i(C_i)$ и некоторые другие. Перечисленные критерии соответствуют задачам минимизации следующих величин: общего времени выполнения заданий; суммы (или среднего значения) моментов окончания выполнения заданий; взвешенной суммы (или средневзвешенного значения) моментов окончания выполнения заданий; взвешенной суммы (или средневзвешенного значения) величин, экспоненциально зависящих от моментов окончания выполнения заданий.

Следует заметить, что число различных моделей теории расписаний, используемых при детерминированном планировании вычислительных процессов, велико. Часто рассматриваются модели, в которых критерий качества зависит от величин запаздывания выполнения заявок, например: $\max_{i \in R} \{C_i - d_i\}$, $\sum_{i \in R} q_i (C_i - d_i)$, $\sum_{i \in R} [\max(0, C_i - d_i)]$, $\sum_{i \in R} q_i \bar{C}_i$, где $\bar{C}_i$ равно 0 при $C_i \leq d_i$ и 1 в противном случае. В практических ситуациях между упорядочиваемыми программами часто существуют информационные связи, что приводит к необходимости учета ограничений предшествования при обслуживании заявок. Подобным моделям уделяется большое внимание [12, 13, 21, 42, 55-60, 63-70, 71, 74]. В последние

годы возрос интерес к моделям организации вычислительного процесса, в которых учитываются дополнительно требуемый при выполнении заявки ресурс, например, память [58]. Приемлемых общих методов решения возникающих задач теории расписаний нет. Например, учет ограничений предшествования значительно усложняет модели и часто переводит их в класс универсальных переборных проблем [69,74]. Таким образом, наряду с построением и анализом эвристических процедур детерминированного планирования вычислительного процесса, актуальным является анализ сложности задач, выделение классов задач, для которых возможно построение эконсмных точных (или хотя бы приближенных) методов решения [31].

2.2 Постановка задач, определения, эквивалентные работы

В рассматриваемых задачах имеется множество заданий (работ) $R = \{1, 2, \dots, n\}$ и ограничение предшествования, заданное направленным ациклическим графом $G = (R, \Gamma)$. Требуется установить оптимальный порядок S^* выполнения работ (расписание), который является допустимым (т.е. удовлетворяет заданным ограничениям предшествования) и на котором значение функционала минимально: $f(S^*) = \min_S f(S)$.

Задача 1. Имеется одна машина. Каждая работа $i \in R$ характеризуется временем выполнения t_i и функцией штрафа $\varphi_i(t) = \alpha_i t + \beta_i$. Требуется минимизировать суммарный штраф $f(S) = \sum_{i \in R} \varphi_i(C_i)$, где C_i - момент окончания выполнения работы i (все работы поступают в момент времени $t = 0$).

Задача 2. Отличается от задачи 1 тем, что функции штрафа имеют вид: $\varphi_i(t) = \alpha_i \exp(\lambda t) + \beta_i$.

Задача 3. Это задача Беллмана-Джонсона в случае двух машин. Каждая работа $i \in R$ должна выполняться сначала на первой машине, затем

на второй о временах выполнения α_i, β_i . Требуется минимизировать суммарное время выполнения всех работ.

Задача 4. Имеется одна машина. Каждая работа $i \in K$ характеризуется временем выполнения τ_i и функцией штрафа $\varphi_i(t) = \alpha_i t + \beta_i$. Требуется минимизировать суммарный штраф $\varphi(s) = \sum_{i \in K} \varphi_i(t_i)$, где $t_i = \tau_i \cdot s[i]$, $s[i]$ - номер работы, находящейся в расписании на i -м месте.

Задача 5. Отличается от задачи 4 тем, что функция штрафа имеет вид: $\varphi_i(t) = \alpha_i \ln \lambda t + \beta_i$.

Задача 6. Имеется одна машина. Каждая работа $i \in K$ характеризуется стоимостью выполнения ρ_i и вероятностью срыва выполнения работы p_i . Требуется минимизировать среднюю стоимость выполнения всех работ $\varphi(s) = \sum_{i \in K} \frac{\rho_i s[i]}{D s[i]}$, где $D s[i] = \prod_{k=1}^i (1 - p_k)$.

Последняя задача заменой переменных сводится к задаче I [43], но ее можно рассматривать и независимо.

Для задачи 3 просто доказывается, как и в [16], что оптимальное расписание можно искать на множестве одномаршрутных расписаний.

Таким образом, расписание s - это перестановка n чисел $1, 2, \dots, n$: $s = \langle s[1], s[2], \dots, s[n] \rangle$.

Указанные задачи, несмотря на различие в постановках, обладают общими свойствами. Таким образом можно рассматривать некоторую обобщенную задачу, которая характеризуется следующим. В случае отсутствия ограничений предшествования она может быть решена упорядочением работ по неубыванию одностепенных функции - приоритетов $\omega(i)$, вычисляемых для каждой работы $i \in K$ (на основе перестановочного приема [42]). Приоритеты для рассматриваемых задач [30, 42, 72, 73] имеют

$$\omega(i) = \frac{\tau_i}{\alpha_i};$$

$$\omega(i) = a_i \exp(\lambda \bar{c}_i) [1 - \exp(\lambda \bar{c}_i)]^{-1};$$

$$\omega(i) = \text{sign}(a_i - \theta_i) [M - \min(a_i, \theta_i)],$$

$$M = \sum_{k \in R} (a_k + \theta_k);$$

$$\omega(i) = \frac{a_i \bar{c}_i}{1 - \bar{c}_i};$$

$$\omega(i) = \frac{a_i \ln \lambda \bar{c}_i}{1 - \ln \lambda \bar{c}_i};$$

$$\omega(i) = \frac{p_i}{p_i'}.$$

Определение 2.1 Допустимые расписания s, s' называются сопряженными по работам i, j , если $s = \langle \dots i, j \dots \rangle, s' = \langle \dots j, i \dots \rangle$ и $s[\ell] = s'[\ell]$ при $s[\ell] \neq i, j$.

Приоритеты обладают следующими свойствами [16, 42], если $\omega(i) \leq \omega(j)$, то

$$j(s) \leq j(s'). \quad (2.1)$$

Другим свойством обобщенной задачи является то, что для нее выполняется условие "склеивания" работ [24, 27], которое заключается в следующем.

Пусть $N(i, j)$ - множество допустимых расписаний работ R , в которых две произвольные фиксированные работы $i, j \in R$ стоят рядом и работа i предшествует работе j . Рассмотрим произвольное расписание $s \in N(i, j)$, в котором $i = s[\ell], j = s[\ell + 1]$ ($1 \leq \ell < n$). Пусть имеется работа $J \in R$. Определим расписание $\bar{s}$ множества работ $(R \setminus \{i, j\}) \cup J$ следующим образом:

$$\bar{s}[\tau] = \begin{cases} s[\tau], & \text{при } \tau = \overline{i, \ell-1}, \\ J, & \text{при } \tau = \ell, \\ s[\tau+1], & \text{при } \tau = \overline{\ell+1, n-1}. \end{cases} \quad (2.2)$$

Определение 2.2 Работу $J = J(i, j)$ назовем эквивалентной рабо-

там i, j , если выполняются следующие два условия:

1) параметры работы J могут быть в явном виде выражены через параметры работ i и j ,

2) для любого расписания $s \in N(i, j)$ выполняется соотношение

$$f(s) = f(s) + c_{ij}, \quad (2.3)$$

где c_{ij} - константа, независимая от s .

Определение 2.3 Расписание S , полученное из $s \in N(i, j)$ по формуле (2.2), называется эквивалентным расписанием S , если J - работа, эквивалентная работам i, j .

Параметры эквивалентной работы $J(i, j)$ и константа c_{ij} выражаются через параметры работ i и j следующим образом:

для задачи 1:

$$a_j = a_i + a_j, \quad \tilde{a}_j = \tilde{a}_i + \tilde{a}_j, \\ b_j = b_i + b_j - a_i \tilde{a}_j, \quad c_{ij} = 0;$$

для задачи 2:

$$a_j = a_i \exp(-\lambda \tilde{a}_j) + a_j, \quad \tilde{a}_j = \tilde{a}_i + \tilde{a}_j, \\ b_j = b_i + b_j, \quad c_{ij} = 0;$$

для задачи 3:

$$a_j = a_i + a_j - \min(a_j, b_i), \quad \tilde{a}_j = b_i + b_j - \min(a_j, b_i), \\ c_{ij} = \min(a_j, b_i);$$

для задачи 4:

$$a_j = \frac{a_i}{\tilde{a}_i} + a_j, \quad \tilde{a}_j = \tilde{a}_i \tilde{a}_j, \\ b_j = b_i + b_j, \quad c_{ij} = 0;$$

для задачи 5:

$$a_j = a_i + a_j, \quad \tilde{a}_j = \tilde{a}_i \tilde{a}_j, \\ b_j = b_i + b_j - a_i \ln \tilde{a}_j, \quad c_{ij} = 0;$$

для задачи 6:

$$p_j = p_i(1-p_j) + p_j, \quad p_j = 1 - (1-p_i)(1-p_j), \\ c_{ij} = 0.$$

Можно ввести склеивание конечного числа работ. Пусть работы множества $Q \subset R$ ($|Q| = k$) в исходном расписании s стоят рядом, начиная с ℓ -го места слева. Тогда эквивалентное расписание $\bar{s}$ определяется формулой

$$\bar{s}[\tau] = \begin{cases} s[\tau], & \text{при } \tau = \overline{1, \ell-1}, \\ J(Q), & \text{при } \tau = \ell, \\ s[\tau+k-1], & \text{при } \tau = \overline{\ell+\ell, n-k+1}. \end{cases}$$

При этом

$$f(s) = f(\bar{s}) + c_Q.$$

Пусть $Q = \{q_1, \dots, q_k\}$ и q_i предшествует q_{i+1} ($1 \leq i < k$).

Тогда параметры эквивалентной работы $J(Q)$ и константа c_Q выражаются через параметры работ множества Q следующим образом:

для задачи 1

$$\begin{aligned} a_{J(Q)} &= \sum_{i=1}^k a_{q_i}, \\ \hat{c}_{J(Q)} &= \sum_{i=1}^k \hat{c}_{q_i}, \\ b_{J(Q)} &= \sum_{i=1}^k b_{q_i} - \sum_{i=2}^k \left(\hat{c}_{q_{i-1}} \sum_{j=1}^{\ell-1} a_{q_j} \right), \\ c_Q &= 0; \end{aligned}$$

для задачи 2

$$\begin{aligned} a_{J(Q)} &= a_{q_k} + \sum_{\ell=1}^{k-1} a_{q_\ell} \exp(-\lambda \sum_{j=\ell+1}^k \hat{c}_{q_j}), \\ \hat{c}_{J(Q)} &= \sum_{i=1}^k \hat{c}_{q_i}, \\ b_{J(Q)} &= \sum_{i=1}^k b_{q_i}, \\ c_Q &= 0; \end{aligned}$$

для задачи 3

$$\begin{aligned} a_{J(Q)} &= \sum_{i=1}^k a_{q_i} - \min \left(\sum_{i=2}^k a_{q_i}, \sum_{i=1}^{k-1} b_{q_i} \right), \\ b_{J(Q)} &= \sum_{i=1}^k b_{q_i} - \min \left(\sum_{i=2}^k a_{q_i}, \sum_{i=1}^{k-1} b_{q_i} \right), \\ c_Q &= \min \left(\sum_{i=2}^k a_{q_i}, \sum_{i=1}^{k-1} b_{q_i} \right); \end{aligned}$$

для задачи 4

$$\begin{aligned} a_{J(Q)} &= a_{q_k} + \frac{k-1}{\ell+1} \frac{a_{q_\ell}}{\prod_{j=\ell+1}^k \hat{r}_{q_j}}, \\ \hat{r}_{J(Q)} &= \sum_{i=1}^k \hat{r}_{q_i}, \\ b_{J(Q)} &= \sum_{i=1}^k b_{q_i}, \\ c_Q &= 0; \end{aligned}$$

для задачи 5

$$\begin{aligned} a_{J(Q)} &= \sum_{i=1}^k a_{q_i}, \\ \hat{r}_{J(Q)} &= \sum_{i=1}^k \hat{r}_{q_i}, \\ b_{J(Q)} &= \sum_{i=1}^k b_{q_i} - \sum_{\ell=1}^{k-1} a_{q_\ell} \ln \left(\prod_{j=\ell}^k \hat{r}_{q_j} \right), \\ c_Q &= 0; \end{aligned}$$

для задачи 6

$$\begin{aligned} p_{J(Q)} &= \sum_{\ell=1}^{k-1} p_{q_\ell} \prod_{j=\ell+1}^k (1-p_{q_j}) + p_{q_k}, \\ p_{J(Q)} &= 1 - \prod_{i=1}^k (1-p_{q_i}), \quad c_Q = 0. \end{aligned}$$

Очевидно, что параметры эквивалентных работ могут быть вычислены и другими способами.

Введем определения.

Определение 2.4. Пусть $x, y \in K$ и $y \in \Gamma x$. Тогда вершина y называется последователем x .

Определение 2.5. Работы $i, j \in K$ называются независимыми, если допустимость произвольного расписания не зависит от порядка следования элементов i и j , т.е. существует допустимое расписание, в котором i предшествует j и допустимое расписание, в котором j предшествует i .

Определение 2.6. Будем говорить, что задача имеет нормальный вид, если для любых $i, j \in R$ таких, что $j \in \Gamma i$ выполняется: $\omega(i) < \omega(j)$.

Очевидно, что если задача имеет нормальный вид, то для получения оптимального расписания достаточно упорядочить множество работ по убыванию их приоритетов, т.к. полученное упорядочение будет допустимым.

Определение 2.7. Множество $R' \subseteq R$ назовем модулем, если для любого $i \in R'$ выполняется одно из трех условий:

- 1) любая работа $j \in R'$ образует с работой i независимую пару,
- 2) работа i должна предшествовать любой работе $j \in R'$,
- 3) любая работа $j \in R'$ должна предшествовать работе i .

Определение 2.8. Будем говорить, что модуль R' имеет нормальный вид, если для любых двух работ $x_1, x_2 \in R'$ таких, что $x_2 \in \Gamma x_1$, выполняется $\omega(x_1) < \omega(x_2)$.

Определение 2.9. Поддерево T_x с корнем x будем называть нормальным, если:

- 1) в множестве вершин поддерева T_x кроме x входят все вершины из R , которым предшествует x ,
- 2) для любых вершин y, z из T_x таких, что $z \in \Gamma y$, выполняется условие $\omega(y) < \omega(z)$.

Если две фиктивные работы, всегда можно перейти от общего слу-

чая ограничения предшествования к двухполюсной сети [17, 54]. Далее будем рассматривать случаи, когда граф $G = (R, \Gamma)$ (сеть Γ) представляет собой дерево или двухполюсную сеть (в которой каждая вершина достижима из входного полюса, а входной полюс достижим из каждой вершины).

Определение 2.10. Сеть, полученная из односторонних сетей в результате конечного числа параллельных и последовательных соединений, называется параллельно-последовательной сетью (П-сетью) [17, 54].

2.3 Преобразование задач

Пусть сеть Γ задает ограничение предшествования. Обозначим через Γ сеть, полученную из Γ "склеивкой" i и j (т.е. в Γ вместо работ i и j входит эквивалентная работа $\bar{j}$ и $\bar{\Gamma} = (\Gamma \cup \{j\}) \setminus i$). Обозначим через $N(i, j)$ множество расписаний работ $(R \setminus \{i, j\}) \cup \{\bar{j}\}$ допустимых на сети Γ , т.е. $N(i, j)$ — это множество всех перестановок элементов из $(R \setminus \{i, j\}) \cup \{\bar{j}\}$, которые удовлетворяют ограничениям предшествования, заданным сетью Γ . Очевидно, что $\bar{s} \in N(i, j)$. отображение $N(i, j) \rightarrow N(i, j)$ по формуле (2.2) обозначим через Φ . В случае выполнения условия "склеивания" работ просто доказывается следующая лемма:

Лемма 2.1. Для того, чтобы расписание $s_0 \in N(i, j)$ было оптимально на $N(i, j)$, т.е. для любого $s \in N(i, j)$ выполнялось $f(s_0) \leq f(s)$ необходимо и достаточно, чтобы эквивалентное расписание $\bar{s}_0 = \Phi(s_0) \in N(i, j)$ было оптимально на $N(i, j)$, т.е. для любого $\bar{s} \in N(i, j)$ выполнялось $f(\bar{s}_0) \leq f(\bar{s})$.

Доказательство леммы 2.1 приводится в приложении I.

Пусть известно, что оптимальное расписание s_0 исходной задачи P_0 можно искать на множестве $N(i, j)$. Тогда в силу леммы 2.1 доста-

точно найти решение $\bar{s}_1$ задачи P_1 на "склеенной" сети Γ , а затем по $\bar{s}_0$ восстановить оптимальное расписание s_0 : $s_0 = \Phi^{-1}(\bar{s}_0)$. Будем говорить, что задачи P_0 и P_1 эквивалентны. Далее, пусть известно, что оптимальное расписание в задаче P_1 можно искать на некотором множестве расписаний $N_1(x, y)$, в которых $x, y \in (R \setminus \{i, j\}) \cup J$, и x и y стоят рядом и $x \in \Gamma_y$. В силу леммы 2.1 для нахождения $\bar{s}_0$ достаточно найти решение $\bar{s}_1$ задачи P_2 , сформулированной на графе Γ_1 , где Γ_1 получается из Γ "склеивкой" вершин x и y .

Обозначим через J_1 работу, эквивалентную x и y , через $N_1(x, y)$ - множество расписаний работ $((R \setminus \{i, j\}) \cup J) \setminus \{x, y\} \cup J_1$, допустимых на сети Γ_1 . Отображение $N(i, j) \rightarrow N_1(x, y)$ обозначим через φ_1 . Тогда $\bar{s}_0 = \varphi_1^{-1}(\bar{s}_1)$ и $s_0 = \Phi^{-1}(\bar{s}_0)$. Будем говорить, что задача P_2 эквивалентна задачам P_1 и P_0 . Процесс "склеивания" вершин и получения эквивалентных задач может быть продолжен. Переход к эквивалентным задачам на основе склеивания работ, стоящих рядом число которых больше двух, аналогичен.

За счет "склеивания" работ и перехода к эквивалентным задачам возможно приведение задач к нормальному виду.

Лемма 2.2 Пусть $G = (K, \Gamma)$ - дерево, $K \subseteq R$, M - множество последователей, причем для любого $y \in M$ поддерево с корнем y является нормальным. Пусть $y_0 \in M$ такой, что для любого $y \in M$

$$\omega(y_0) \leq \omega(y)$$

и

$$\omega(x) \geq \omega(y_0).$$

Тогда от любого допустимого расписания работ множества K можно без ухудшения критерия перейти к некоторому допустимому расписанию, в котором работы x и y находятся рядом.

Доказательство леммы 2.2 приводится в приложении I.

Лемма 2.2 позволяет любую исходную задачу на дереве с помощью

"склеивание" работ приводит к эквивалентной задаче нормального вида. Рассмотрим этот процесс. Висячие вершины составляют множество M^0 . Каждый элемент этого множества представляет собой нормальное поддерево. Рассмотрим M^1 - множество вершин, все последователи которых принадлежат M^0 . Пусть x - произвольный элемент множества M^1 и $M_x = \{i \in R \mid i \in P(x)\}$. Пусть элемент $i_0 \in M_x$ такой, что $\omega(i_0) \leq \omega(i)$ для всех $i \in M_x$. Если $\omega(x) < \omega(i_0)$, то поддерево с корнем x является нормальным. Если $\omega(x) \geq \omega(i_0)$, то по лемме 2.2 в любом допустимом расписании, в том числе и в оптимальном, перестановками, не нарушающими допустимость и не ухудшающими значение критерия, можно перейти к расписанию, в котором работы x и i_0 находятся рядом. Таким образом, можно искать оптимальное расписание на множестве тех расписаний $N(x, i_0)$, в которых работы x и i_0 находятся рядом. По лемме 2.1 можно перейти к эквивалентной задаче, т.е. искать оптимальное расписание на множестве эквивалентных расписаний $\bar{N}(x, i_0)$, в которых вместо работ x и i_0 рассматривается эквивалентная работа $\bar{j}(x, i_0)$.

Проводя описанную выше процедуру для эквивалентной задачи и осуществляя, в случае необходимости "склеивание" и переход к эквивалентным задачам, за конечное число шагов (множество конечно) получим эквивалентную задачу, в котором все вершины множества M^1 являются корнями нормальных поддеревьев.

Затем перейдем к множеству вершин M^2 , все последователи которых принадлежат множеству M^1 и т.д. Продолжая этот процесс получим эквивалентную задачу нормального вида. После составления оптимального расписания эквивалентной нормальной задачи для получения оптимального расписания исходной задачи S_0 нужно замкнуть все эквивалентные работы на исходные.

Если ограничения предоставления в задаче заданы в виде дорези-

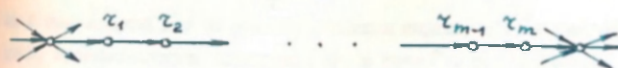


Рис. 1

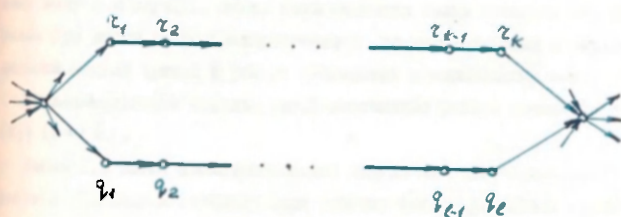


Рис. 2

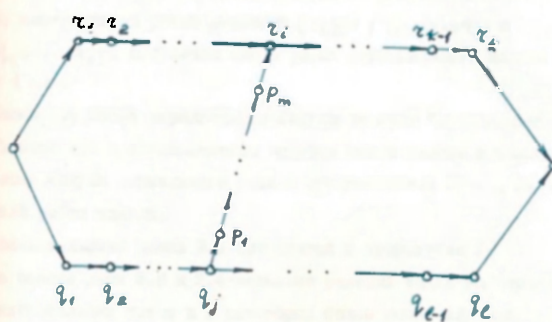


Рис. 3

ев с ориентацией дуг от височных вершин к корням, т.е. с ориентацией, противоположной ориентации дуг в сети Γ , все приведенные выше результаты сохраняют силу с точностью до симметрии. В частности, "склеивание" работ x, y производится в том случае, если $\mu(y) < \mu(x)$.

За счет "склеивания" работ возможно приведение задач к нормальному виду и в случаях, когда сеть является более сложной, чем дерево. При любом графе предшествования последовательным и параллельным модулям (рис.1 и рис.2) приводится к нормальному виду.

Последовательный модуль (рис.1) составляют работы множества $\{r_i\} (i = \overline{1, k})$.

Лемма 2.3 Любому последовательному модулю за счет "склеивания" работ и перехода к эквивалентным задачам можно привести к нормальному виду с трудоемкостью $O(k)$, где k - число работ модуля.

Доказательство леммы 2.3 приводится в приложении I.

Параллельный модуль (рис.2) составляют два последовательных модуля, состоящих из работ множеств $\{r_i\} = \{r_1, \dots, r_k\}$ и $\{r_j\} = \{r_1, \dots, r_l\}$. Обозначим число работ параллельного модуля через $m = k + l$.

Лемма 2.4 Любому параллельному модулю за счет "склеивания" работ и перехода к эквивалентным задачам можно свести к последовательному модулю нормального вида с трудоемкостью $O(m)$, где m - число работ модуля.

Доказательство леммы 2.4 приводится в приложении I.

На основе лемм 2.3 и 2.4 возможно решение задач на параллельно-последовательных сетях и в некоторых более общих случаях.

2.4 Алгоритмы упорядочения

Приведем два эффективных метода решения рассматриваемых задач.

В случае, когда ограничение предшествования имеет вид дерева, алгоритм имеет следующий вид.

Алгоритм 2.1

I. Приведение исходной задачи к эквивалентной задаче нормального вида.

0 шаг. Выделение всех исходных вершин (множество M^0).

1 шаг. Выделение множества M^1 (множество вершин, все последователи которых принадлежат множеству M^0). Для каждой вершины $x \in M^1$ сравнение $\omega(x)$ и $\omega(i_0)$, где i_0 такова, что $\omega(i_0) \leq \omega(i)$ для всех $i \in x$. Если $\omega(x) \geq \omega(i_0)$, то "склеивание" работ x и i_0 и переход к эквивалентной задаче. (Для эквивалентной задачи проводится аналогичное сравнение эквивалентной работы J с ее последователями.)

* шаг. Выделение множества M^k (множество вершин, все последователи которых принадлежат множеству $\bigcup_{j=0}^{k-1} M^j$). Для каждой вершины $x \in M^k$ сравнение $\omega(x)$ и $\omega(i_0)$, где i_0 такова, что $\omega(i_0) \leq \omega(i)$ для всех $i \in x$. Если $\omega(x) \geq \omega(i_0)$, то "склеивание" работ x и i_0 и переход к эквивалентной задаче. (Для эквивалентной задачи проводится аналогичное сравнение эквивалентной работы J с ее последователями.)

II. Решение эквивалентной нормальной задачи (упорядочение работ по неубыванию приоритетов $\omega(i)$).

III. Замена эквивалентных работ в решении эквивалентной нормальной задачи на работы исходного множества R .

Описанный алгоритм аналогичен алгоритмам Хорна [60] и Гордона-Танаева [13]. Для алгоритма Хорна известна оценка трудоемкости $O(n \log n)$, полученная в работе [56]. Эта же оценка справедлива и для алгоритма 2.1.

В случае, когда сеть предшествования сводится к параллельно-последовательной сети, алгоритм имеет вид.

Алгоритм 2.2

I шаг. Приведение к нормальному виду всех последовательных мо-

дугей (за счет "склеивания" работ и перехода к эквивалентным задачам).

2 шаг. Если вся сеть приведена к последовательному модулю нормального вида, то переход к шагу 4.

3 шаг. Проведение всех возможных слияний последовательных модулей нормального вида в параллельных модулях. Переход к шагу 1.

4 шаг. Замена эквивалентных работ на работы исходного множества.

Трудоемкость процесса нормализации последовательных модулей будет $O(n)$, т.к. если имеются последовательные модули с числом вершин $k_1, k_2, \dots, k_e$, то $O(k_1) + O(k_2) + \dots + O(k_e) \leq O(n)$ в силу того, что $\sum_{i=1}^e k_i \leq n$. Аналогично трудоемкость процесса слияний на шаге 3 будет $O(n)$. Каждое слияние последовательных модулей нормального вида в параллельных модулях уменьшает число последовательных модулей на единицу. Так как число последовательных модулей ограничено числом вершин n , то возможно не более n слияний, т.е. число циклов в алгоритме не превышает n и трудоемкость каждого из них $O(n)$.

Таким образом, трудоемкость алгоритма будет не более $O(n^2)$, где n — число работ исходного множества R . Следует отметить, что трудоемкость алгоритма для П-сети может быть понижена до $O(n \log n)$ [55, 67, 70].

Предложенный алгоритм позволяет эффективно решать задачи в некоторых случаях, когда ограничение задано более сложным образом, чем сеть, сводящаяся к П-сети, например на некоторых сетях, содержащих Н-сети^{1/}.

Пусть задано множество работ $R = \{r_1, r_2, \dots, r_n\}$, где $\{r_i\} =$ 1/Сеть, которую нельзя представить в виде последовательного или параллельного соединения сетей, называется Н-сетью [17, 54].

$= \{c_1, \dots, c_k, c_{k+1}, \dots, c_n\} = \{c_1, \dots, c_k\}, \{p_m\} = \{p_1, \dots, p_m\}$ и имеет место ограничение предшествования в виде, представленном на рис.3. Введением двух фиктивных работ сведем исходную сеть к двухполюсной сети, представляющей собой H-сеть. Очевидно, что в оптимальном допустимом расписании работа a_j предшествует некоторой работе $a_{j'}$ ($1 \leq j' \leq n$). Решив задачу для всех возможных значений j' (на n различных H-сетях) и выбрав решения с минимальным значением критерия, получим оптимальное допустимое расписание исходной задачи с трудоемкостью $O((k + l + m)^2) \leq O(n^3)$.

2.5 Примеры составления расписаний

Пример 1.

Рассмотрим задачу 3. Пусть имеется множество работ $R = \{1, 2, \dots, 10\}$, параметры и приоритеты $\omega(i)$ которых представлены в таблице 1. Пусть ограничения предшествования заданы в виде сети Γ , представленной на рис.4. Применим алгоритм 2.1.

1. Приведение исходной задачи к эквивалентной задаче нормативного вида.

0 шаг. Выделяем множество висячих вершин:

$$M^0 = \{1, 3, 6, 8, 9, 10\}.$$

1 шаг. Выделяем множество вершин M^1 , все последователи которых принадлежат множеству M^0 : $M^1 = \{5, 7\}$.

Сравниваем работу 5 с ее последователями: $M_5 = \{8, 9\}$, $\omega(9) < \omega(8)$, $\omega(5) > \omega(9)$. Требуется рассчитать параметры эквивалентной работы $J = J(5, 9)$ и перейти к эквивалентной задаче: $\alpha_{J(5, 9)} = 7 + 2 - 2 = 7$, $\beta_{J(5, 9)} = 18 + 20 - 2 = 36$, $\omega(J(5, 9)) = 207$, $R = \{1, 2, 3, 4, J(5, 9), 6, 7, 8, 10\}$.

Сравниваем работу $J(5, 9)$ с ее последователями

$$M_{J(5, 9)} = \{8\}, \omega(J(5, 9)) < \omega(8).$$

Сравниваем работу 7 с ее последователями: $M_7 = \{10\}$,

$\omega(7) > \omega(10)$. Требуется рассчитать параметры эквивалентной работы $J = J(7, 10)$ и перейти к эквивалентной задаче:

$$\alpha_{J(7,10)} = 19+9-1 = 27, \beta_{J(7,10)} = 1+15-1 = 15,$$

$$\omega(J(7,10)) = 199, R = \{1, 2, 3, 4, J(5,9), 6, J(7,10), 8\}.$$

Новая сеть предшествований представлена на рис.5.

2 шаг. Выделим множество вершин M^2 , все последователи которых принадлежат множеству $(M^1 \cup M^1)$; $M^2 = \{4\}$. Сравним работу 4 с ее последователями: $M_4 = \{J(5,9), 6, J(7,10)\}$, $\omega(J(5,9)) < \omega(J(7,10))$, $\omega(J(5,9)) < \omega(6)$, $\omega(4) > \omega(J(5,9))$. Рассчитываем параметры эквивалентной работы $J = J(4, J(5,9))$ и переходим к эквивалентной задаче:

$$\alpha_{J(4, J(5,9))} = 20+7-7 = 20, \beta_{J(4, J(5,9))} = 8+36-7 = 37,$$

$$\omega(J(4, J(5,9))) = 194.$$

Сравниваем работу $J(4, J(5,9))$ с ее последователями:

$$M_{J(4, J(5,9))} = \{8, 6, J(7,10)\}, \omega(J(7,10)) < \omega(8),$$

$$\omega(J(7,10)) < \omega(6), \omega(J(4, J(5,9))) < \omega(J(7,10)).$$

$$\text{Новое множество } R = \{1, 2, 3, J(4, J(5,9)), 6, 8, J(7,10)\}.$$

Сеть предшествований представлена на рис.6.

3 шаг. Выделим множество вершин M^3 , все последователи которых принадлежат $\bigcup_{j=0}^2 M^j$; $M^3 = \{2\}$. Сравниваем работу 2 с ее последователями: $M_2 = \{3, J(4, J(5,9))\}$, $\omega(3) < \omega(J(4, J(5,9)))$,

$$\omega(2) < \omega(3).$$

Таким образом получена эквивалентная нормальная задача на множество работ $R = \{1, 2, 3, J(4, J(5,9)), 6, J(7,10), 8\}$

с ограничением предшествования, заданным сетью, представленной на рис.6

II. Построение оптимального расписания S_0 для эквивалентной нор-

Таблица I.

Номер работы i	Время выполнения работы		Приоритет $\omega(i)$
	α_i	β_i	
1	6	13	- 208
2	3	13	- 211
3	4	16	- 210
4	20	8	206
5	7	18	- 207
6	14	5	209
7	19	1	213
8	12	9	205
9	2	20	- 212
10	9	15	- 215

I.



Рис. 4

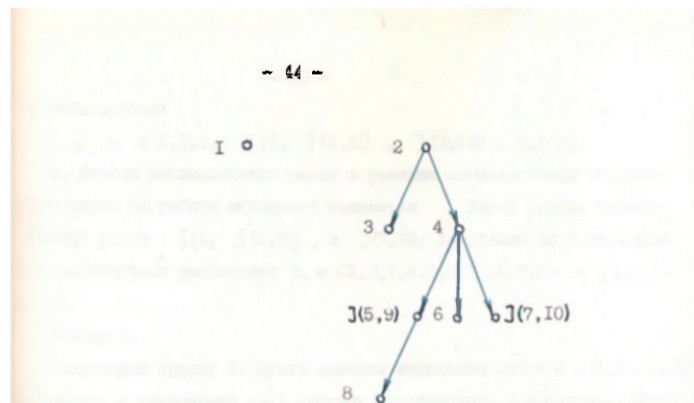


Рис. 5

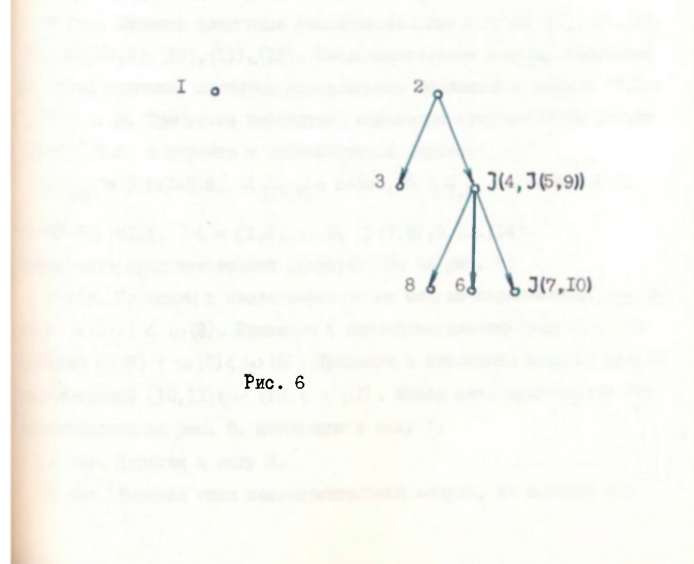


Рис. 6

мальной задачи:

$$\bar{s}_0 = \langle 2, 3, 1, J(4, J(5, 9)), J(7, 10), 8, 6 \rangle.$$

III. Замена эквивалентных работ в решении эквивалентной нормальной задачи на работы исходного множества. После замены эквивалентных работ $J(4, J(5, 9))$ и $J(7, 10)$ на работы исходного множества получаем расписание $s_0 = \langle 2, 3, 1, 4, 5, 9, 7, 10, 8, 6 \rangle$ с $f(s_0) = 121$.

Пример 2.

Рассмотрим задачу I. Пусть имеется множество работ $R = \{1, 2, \dots, 14\}$ параметры и приоритеты $\omega(i)$ которых представлены в таблице 2. Пусть ограничения предшествования заданы в виде сети, представленной на рис. 7. Введением фиктивной работы сведем исходную сеть к II-сети. Применим алгоритм 2.2.

I шаг. Переход к шагу 2.

2 шаг. Имеются следующие последовательные модули: $\{1\}, \{2\}, \{3\}, \{5\}, \{6\}, \{7, 8\}, \{10\}, \{11\}, \{12\}$. Последовательные модули, состоящие из одной вершины, являются нормальными. Нормализуем модуль $\{7, 8\}$: $\omega(7) > \omega(8)$. Требуется рассчитать параметры эквивалентной работы $J = J(7, 8)$ и перейти к эквивалентной задаче:

$$J(7, 8) = 7.5 + 1 = 8.5, \quad \omega_{J(7, 8)} = 5 + 1 = 6, \quad b_{J(7, 8)} = 2 + 4 \cdot 5 \cdot 1 = 1.$$

$$\omega(J(7, 8)) = 1.4, \quad R = \{1, 2, \dots, 6, J(7, 8), 9, \dots, 14\}.$$

Исходная сеть предшествования представлена на рис. 8.

3 шаг. Приводим к последовательному модулю параллельный модуль $\{1, 2\}$: $\omega(1) < \omega(2)$. Приводим к последовательному модулю параллельный $\{5, 6\}$: $\omega(5) < \omega(6)$. Приводим к последовательному модулю параллельный $\{10, 11\}$: $\omega(10) < \omega(11)$. Новая сеть предшествования представлена на рис. 9. Переходим к шагу I.

I шаг. Переход к шагу 2.

2 шаг. Имеется один последовательный модуль, не имеющий нор-

Таблица 2.

Номер работы i	Время выполнения работы τ_i	Параметры		Приоритет $\omega(i)$
		a_i	b_i	
1	1	1	15	1
2	6	2	7	3
3	2	1	8	2
4	4	4	12	1
5	4.5	3	10	1.5
6	5	2	30	2.5
7	7.5	5	2	1.5
8	1	1	4	1
9	20	4	1	5
10	3	3	0	1
11	5.2	2	7	2.6
12	1.5	0.5	8	3
13	7.5	3	5	2.5
14	2	1	4	2

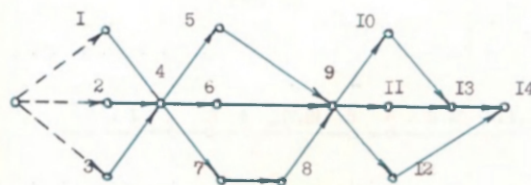


Рис. 7

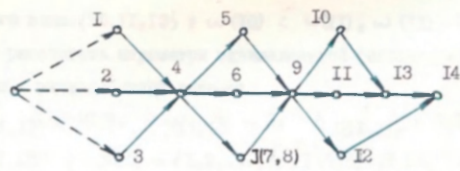


Рис. 8

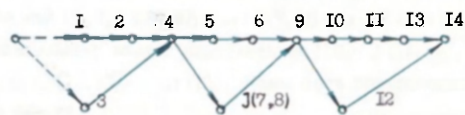


Рис. 9

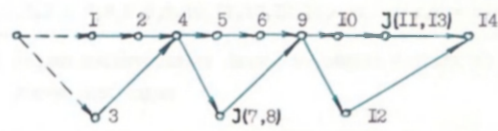


Рис. 10

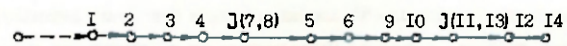


Рис. 11

мального вида: $\{I0, II, I3\} : \omega(I0) < \omega(II), \omega(II) > \omega(I3)$. Требуется рассчитать параметры эквивалентной работы $J = J(II, I3)$ и перейти к эквивалентной задаче:

$\hat{J}(II, I3) = 12.7$, $\alpha_{J(II, I3)} = 5$, $\beta_{J(II, I3)} = 7.5 - 2 \times 7.5 = -3$,
 $\omega(J(II, I3)) = 2.5\%$, $R = \{1, 2, \dots, 6, J(7, 8), 9, I0, J(II, I3), I2, I4\}$,
 $\omega(I0) < \omega(J(II, I3))$. Новая сеть представлена на рис. I0.

3 шаг. Приводим к последовательному модулю параллельный $\{1, 2, 3\} : \omega(1) < \omega(3) < \omega(2)$. Приводим к последовательному модулю параллельный $\{5, 6, J(7, 8)\} : \omega(J(7, 8)) < \omega(5) < \omega(6)$. Приводим к последовательному модулю параллельный $\{I0, J(II, I3), I2\} : \omega(I0) < \omega(J(II, I3)) < \omega(I2)$. Новая сеть предшествования представлена на рис. II.

Переход к шагу I.

I шаг. Переход к шагу 4.

4 шаг. Получение оптимального расписания:

$S_0 = \langle 1, 3, 2, 4, 7, 8, 5, 6, 9, I0, II, I3, I2, I4 \rangle$ с $f(s_0) = 1239.8$

2.6 Анализ характеристик одного семейства близких по постановке задач

Большое значение имеет сравнительный анализ сложности близких по постановке задач, который может позволить выявить на интуитивном уровне дальнейшие направления исследований:

- построение точного эффективного точного метода решения;
- доказательство того, что задача является NP -полной и построение приближенных алгоритмов (с гарантированной точностью, статистически эффективных, эвристических).

В таблице 3 приведены характеристики сложности одного семейства близких по постановке задач теории расписаний в случае идентичных процессоров. В каждой клетке таблицы для соответствующей задачи

использованы следующие обозначения: "ж" - существует эффективный точный алгоритм, "!" - задача является универсальной переборной, "7" - вопрос о сложности задачи в настоящее время является открытым. Очевидно, что сложность задач не уменьшается при следующих изменениях в постановке:

- переход от рассмотрения одного процессора к двум и более;
- переход от равных времен выполнения работ к неравным;
- учет ограничения предшествования более сложного вида (от отсутствия условия предшествования к дереву, П-сети, или ациклическому направленному графу).

Введем следующие позиционные обозначения постановок задач рассматриваемого семейства: $A | B | \Gamma$, где в позиции А проставляется число процессоров; в позиции В - тип ограничения предшествования (если в этой позиции - пропуск, то ограничение предшествования отсутствует); в позиции Γ - возможные значения времен выполнения работ; в позиции Γ - вид критерия из числа приведенных в таблице 3. Число процессоров может быть равно 1, 2 или m (более 2-х). Возможны следующие обозначения типов ограничений предшествования: дерево (основность деревьев), П-сеть (параллельно-последовательная сеть), G (ациклический направленный граф). Значения времен выполнения работ могут характеризоваться тремя вариантами обозначений: I (времена выполнения равны некоторой константе); I, 2 (времена выполнения равны некоторой константе или удвоенному ее значению); I' (времена выполнения произвольны из интервала $(0, \infty)$).

Для некоторых задач в таблице 3 указаны ссылки на литературный источник, в котором показывается принадлежность задачи к классу универсальных переборных проблем или приводится полиномиальный точный алгоритм решения. В последнем случае указана также минимальная оценка трудоемкости алгоритма. На основе этих задач, как исходных,

Таблица 3

Число выпол- няемых работ	Время выпол- нения	Без ограничений производительности				Вид				оценки				пришествования			
		Вид целевой функции				Детермо				П-сеть				Алгоритмы на языке мат. программирования			
		ΣC_i	$\Sigma a_i x_i$	$\Sigma q_i C_i$	ΣC_i	$\Sigma a_i x_i$	$\Sigma q_i C_i$	ΣC_i	$\Sigma a_i x_i$	$\Sigma q_i C_i$	ΣC_i	$\Sigma a_i x_i$	$\Sigma q_i C_i$	ΣC_i	$\Sigma a_i x_i$	$\Sigma q_i C_i$	ΣC_i
1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	1,2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0∞	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
2	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	1,2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0∞	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
$m \geq 3$	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	1,2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	0∞	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* - существует эффективный точный алгоритм, ! - задача является минимальной переборной, ? - вопрос о сложности задачи остается открытым.

в таблице 3 указана сложность других, близких по постановке (для универсальных переборных в направлениях вниз и вправо, для решаемых эффективно - вверх и влево). Дополнительно, в качестве исходной использованы задачи $m || I, 2 | C_{\max}$, $m || 1 | \Sigma a_i e^{C_i}$, $m || 1 | \Sigma a_i C_i$, т.к. для них существуют очевидные точные алгоритмы решения.

Представляет интерес исследование задач, помеченных в таблице 3 знаком "?". Следует отметить, задачи с критериями $\Sigma a_i C_i$ и $\Sigma a_i e^{C_i}$ очевидно близки по сложности, поэтому целесообразно распространение результатов, полученных для задач при первом критерии, на задачи со вторым критерием, например: доказательство принадлежности к классу универсальных переборных проблем задач $2 || C | \Sigma a_i e^{C_i}$; $1 | G | 1 | \Sigma a_i e^{C_i}$. Кроме того, исследование задач $1 || 1 | \Sigma a_i C_i$ и $1 || 1 | \Sigma a_i e^{C_i}$ на принадлежность к классу NP - полных проблем может позволить в значительной степени заполнить таблицу 3.

Одним из наиболее красивых результатов в теории расписаний является алгоритм Т.С.Ху [50] с трудоемкостью $O(n)$ для задачи $m | \text{дерево} | C_{\max}$. Этот алгоритм основан на составлении расписания с конца, начиная с наиболее удаленных от корня дерева висячих вершин. В работе [7] предложен аналогичный метод решения этой задачи, позволяющий получать некоторый класс оптимальных расписаний. При этом показано, что оптимальное решение можно получать и при упорядочивании работ, начинающихся с корня дерева. По мнению автора целесообразна попытка распространения этих результатов на задачу $m | \text{ш-сеть} | C_{\max}$.

Другим важным направлением исследования представленных в таблице 3 задач является построение приближенных методов решения для задач, являющихся универсальными переборными. Например, для задачи $m || C | C_{\max}$ построен приближенный алгоритм в работе [71]. Следует иметь в виду, что некоторые из указанных задач могут "не допускать"

эффективных приближенных алгоритмов [30]. Для таких задач, а они как правило, представляют наибольший интерес с точки зрения организации реальных вычислительных процессов, целесообразен анализ вершинических алгоритмов.

Выводы к главе 2

1. Создание экономных методов упорядочивания программ в процессе составления оптимального расписания и точных решающих правил оптимизации является одной из основных задач, возникающих при анализе и синтезе вычислительного процесса методами детерминированного планирования.

2. В связи с тем, что во многих практических ситуациях программы информационно связаны, т.е. выходные результаты одних программ являются входными данными для других, актуальна постановка и решение задач детерминированного планирования с учетом ограничений предшествования.

3. Сложность задач теории расписаний, возникающих при исследовании систем организации вычислительного процесса методами детерминированного планирования, обуславливает важность выделения классов задач, для которых возможно построение эффективных точных методов решения. Большое значение имеет также анализ характеристик задач, который может позволить выявить наиболее целесообразные подходы к построению приемлимых алгоритмов.

4. Выделен класс задач теории расписаний, для которых возможно построение эффективных точных методов решения при ограничении предшествования в виде совокупности деревьев или параллельно-последовательной сети:

- при отсутствии ограничений предшествования задачи этого класса могут быть решены упорядочением работ по неубыванию одноместных функций-приоритетов, вычисляемых для работ упорядочиваемого мно-

чества (на основе перестановочного приема [42]) ;

- в этих задачах должно выполняться условие "склеивания работ" [24, 27].

Сформулирован и доказан ряд лемм, на основе которых построены эффективные точные алгоритмы (за счет эквивалентного преобразования задач). Показано, что к введенному классу относятся шесть известных задач теории расписаний, в том числе задача минимизации суммы штрафа в случае одного процессора с линейными или экспоненциальными функциями штрафа и задача Беллмана-Джонсона в случае двух процессоров. Работа алгоритмов продемонстрирована на примерах решения различных задач. Показано, что предложенный подход позволяет эффективно решать задачи рассматриваемого класса и на некоторых сетях, более сложных, чем параллельно-последовательные.

5. Проведен анализ характеристик семейства близких по постановке задач теории расписаний. Он позволяет выявить наиболее перспективные направления возможностей решения задач.

Глава 3. Экстремальная модель организации данных при ограничении на общий объем используемой опе- ративной памяти

В современных вычислительных системах широко используется концепция иерархической многоуровневой памяти. При этом на высшем уровне памяти, в ОП, находится ограниченная часть данных, а данные, размещенные в ВП должны перед использованием переписываться в ОП. Таким образом, возникает задача определения дисциплины размещения данных в ОП и обмена с ВП [31]. Для организации обмена информация делится на блоки: переменной длины - сегменты, фиксированной длины - страницы. Определение состава сегментов или страниц является одним из основных задач общей задачи, указанной выше [2, 31].

Одним из основных критериев является число обменов между ОП и ВП [2, 6, 7, 31]. В данной главе рассматривается один из вариантов задачи организации данных, когда за счет объединения модулей в сегменты возможно сокращение среднего числа обменов между ОП и ВП при ограничении на объем выделяемой ОП. Эта задача рассматривается на примере компоновки модулей некоторого программного комплекса. Даже простейшие частные случаи задачи являются универсальными пореборными, поэтому анализ задачи и построение алгоритмов ее решения основано на переходе от простейших постановок ко все более сложным. Рассматривается ситуация, когда связь между модулями имеет вид дерева. Построены эффективные приближенные алгоритмы с гарантированным относительным погрешностью.

Некоторые простейшие варианты задачи близки по постановке к задачам, рассмотренным в [6, 7]. Отличие заключается в критериях и в том, что автором, для случая неравных объемов модулей, предложены

эффективные приближенные методы решения.

При дальнейшем обобщении задачи, когда связи между модулями имеют вид ациклического графа, ее сложность значительно возрастает. Причем это сказывается уже на этапе формальной постановки. В работе показано, что в этом случае задача может быть при помощи эвристических методов сведена к рассмотренным древовидным постановкам. Следует отметить, что и на этапе постановки, и на этапе решения задачи важное значение имеет участие разработчика системы. Это может позволить учесть некоторые неформализованные факторы.

Число модулей в программных комплексах обычно имеет порядок $10^3 - 10^5$. Поэтому использование методов полного перебора может привести к затратам, соизмеримым с выигрышем за счет рациональной компоновки. Это обуславливает важность построения эффективных алгоритмов. Автор отдает себе отчет в том, что предложенные модели являются в определенной степени упрощенными по отношению ко многим реальным программным комплексам. Но анализ этих моделей поможет специалистам в значительной степени формализовать свой опыт и интуицию и возможно послужит основой для дальнейших исследований.

Пример использования предложенного в данной главе подхода к организации данных применительно к реальной подсистеме АСУ приводится в следующей главе.

3.1 Постановка задачи

В этой главе предлагаются некоторые задачи организации данных и алгоритмы их решения. Задачи заключаются в такой компоновке данных, которая обеспечивает максимально возможное уменьшение среднего числа обменов между ВП и ОП (по сравнению со случаем, когда компоновка не проводилась) при ограничении на используемый объем ОП. Предлагаемые задачи рассматриваются на примере компоновки про-

граммных модулей, входящих в состав некоторого комплекса.

При функционировании программных комплексов (операционных систем, пакетов прикладных программ) имеют место ситуации, когда одни программные модули, находящиеся в ОП вызывают для выполнения из ОП другие. За счет объединения программных модулей возможно снижение среднего числа вызовов из ОП. В качестве моделей программных комплексов обычно рассматриваются ациклические направленные графы, вершины которых взаимно-однозначно соответствуют программным модулям, а дуги указывают связи между вызывающими и вызываемыми модулями [20, 32, 36]. При этом часто предполагается, и это соответствует многим реальным системам, что передача по управлению осуществляется с возвратом в вызывающий модуль [20, 32].

Задача компоновки сводится к выбору стягиваемых дуг во взвешенном ориентированном графе. Предполагается, если это не оговорено особо, что граф представляет собой дерево. Даже в простейшем случае, когда система состоит из основного сегмента и вызываемых подпрограмм, задача компоновки не проще задачи о рюкзаке, которая принадлежит к классу универсальных переборных проблем [3].

Пусть задано направленное дерево $G = (\hat{R}, \Gamma)$ с ориентацией дуг от корня к висшим вершинам, где $\hat{R}$ - множество вершин, взаимно-однозначно соответствующее множеству программных модулей, Γ - многозначное отображение $\hat{R}$ в $\hat{R}$, характеризующее отношение вызова между программными модулями. Каждая вершина $a \in \hat{R}$ имеет вес $v(a) \geq 0$ (требуемый объем ОП). Дугу из a в b ($a, b \in \hat{R}$) будем обозначать (a, b) , направление дуги указывает вершину, соответствующую вызываемому модулю. Обозначим корень дерева через a_0 .

Определение 3.1 Назовем упорядоченное множество вершин $L(a_1, a_2) = \langle a_1, \dots, a_i, \dots, a_\ell \rangle$ путем от a_1 к a_ℓ , $a_i \in \hat{R}$ ($i = 1, \ell$), если $a_{i+1} \in \Gamma a_i$ ($i = 1, \ell - 1$). Величину $v(L(a_1, a_2)) = \sum_{i=1}^{\ell} v(a_i)$ назовем весом

пути $L(a, a_e)$. Будем говорить, что вершина b достижима из a или a предшествует b , если существует путь $L(a, b)$.

Определение 3.2 Назовем критическим путем дерева G путь $L^*(G) = L(a_0, a)$, где $a \in \{b \in K \mid \Gamma b = \emptyset\}$, если $v(L(a_0, a)) = \max_{b \in K} v(L(a_0, b))$. Величину $v(G) = v(L^*(G))$ назовем весом дерева G , $v(G) = 0$ при $K = \emptyset$.

Пусть $G_a = (R_a, \Gamma_a)$ - поддереву с корнем $a \in K$ (в R_a кроме a входят все вершины из R , достижимые из a).

Определение 3.3 Назовем "хвостом" вершины $a \in K$ графа $(\{R_a \setminus a\}, \Gamma_a)$, а весом "хвоста" вершины a величину $\hat{v}(a) = v(G_a) - v(a)$. Очевидно, что $\hat{v}(a) = \max_{b \in \Gamma_a} v(G_b)$.

В зависимости от исходных данных при работе модуля $a \in K$ возможны n_a вариантов, каждый из которых характеризуется набором значений средних чисел вызовов модулей из множества Γ_a . Обозначим через d_a^j среднее число вызовов модуля $b \in \Gamma_a$ модулем a при варианте j ($j = \overline{1, n_a}$). Пусть каждый вариант характеризуется вероятностью p_j ($\sum_{j=1}^{n_a} p_j = 1$). Среднее число вызовов модуля b модулем a равно $d_a(a) = \sum_{j=1}^{n_a} p_j d_a^j$. Определим среднее число вызовов программного модуля a при функционировании системы следующим рекуррентным соотношением: $d(a) = \sum_{c \in \Gamma_a} d_a(c) d(c)$, $d(a_0) = 1$. Тогда среднее число вызовов модуля b модулем a при функционировании системы (вес дуги от a к b) равно $w(a, b) = d(a) d_b(a)$. Предполагается, что при вызове модуля b модулем a последний находится в Π , а модуль b загружается в Π из Π (адрес загрузки $\forall b \in K$: $\gamma(b) = \max_{a \in \Gamma^{-1}(b)} (\gamma(a) + v(a))$, $\gamma(a_0) = 1$). Вес дерева G определяет требуемый для функционирования программного комплекса объем Π , а среднее число вызовов программных модулей из Π равно:

$$W(G) = \sum_{\substack{a, b \in K \\ b \in \Gamma_a}} w(a, b). \quad (3.1)$$

Заметим, что вершина $a \in K$ в свою очередь может представлять собой совокупность взаимосвязанных модулей, т.е. взвешенный граф и вес a равен весу этого графа.

Определение 3.4 Назовем дерево G допустимым, если $v(G) \leq V$, где V - ограничение на объем ОП, предоставляемой системе.

Объединение (склеивание) программных модулей a и b ($b \in \Gamma_a$), позволяет заменить вызов модуля b модулем a из W на переход к модулю b внутри ОП. Это снижает значение (3.1) на $w(a, b)$, но увеличивает вес графа на величину $\Delta(a, b) \in [0, v(b)]$.

Пусть $\{P\} = \Pi$. Введем множество дуг $U = \{u_1, \dots, u_i, \dots, u_{n-1}\}$. Обозначим через $G = (K, \Gamma)$ граф, полученный из G склеиванием вершин $a, b \in K, b \in \Gamma_a$, т.е. в G вместо вершин a и b входит вершина $J = J(a, b)$ с весом $v(J(a, b)) = v(a) + v(b), \Gamma J = \{\Gamma_a \cup \Gamma_b\} \setminus b$. При этом множество дуг в G : $\bar{U} = U \setminus (a, b)$. Очевидно, что операция преобразования графа при склеивании вершин ассоциативна, т.е. порядок склеивания вершин (стягивания дуг) не существен. Введем переменную x_i ($i = 1, n-1$), которая равна 1, если дуга u_i стягивается и 0 в противном случае. Тогда множество пар склеиваемых вершин можно задать булевым вектором $x = (x_1, \dots, x_i, \dots, x_{n-1})$. Перейдем от (3.1) к критерию

$$W(x) = \sum_{i=1}^{n-1} x_i w(u_i).$$

С учетом стягивания дуг для любого $a \in K$ вес вершины и вес "хвоста" можно рассматривать как функцию вектора x : $v(a, x), v(a, x)$. Таким образом, задача состоит в определении вектора x , который максимизирует (3.2) при выполнении условия допустимости графа.

$$v(a_0, x) + v(a_0, x) \leq V. \quad (3.3)$$

Назовем эту задачу постановкой I. Далее потребуются вспомогатель-

ная задача. Пусть V^- - ограничение на вес вершины a_0 , V^+ - ограничение на вес хвоста a_0 , $V^- + V^+ = V$.

Постановка 2. Максимизировать $W(x)$ при условиях

$$v(a_0, x) \leq V^-, \quad \bar{v}(a_0, x) \leq V^+, \quad (3.4)$$

Конкретизируем понятие приближенного решения задачи, предложенное в главе I (определение I.5) применительно к некоторым вариантам задачи компоновки. Пусть $\varepsilon, \delta \in [0, 1]$, x^* - точное решение задачи (3.2), (3.3) или (3.2), (3.4). Введем для некоторых x функции относительной ошибки целевой функции $d(x)$ и ресурса $\beta(x)$:

$$d(x) = \max \left[0, \frac{W(x^*) - W(x)}{W(x^*)} \right], \quad (3.5)$$

$$\beta(x) = \max \left[0, \frac{v(x) - V}{V} \right], \quad (3.6)$$

где $v(x) = v(a_0, x) + \bar{v}(a_0, x)$, $W(x^*) \neq 0$.

Задача состоит в определении при фиксированных ε и δ такого x^0 , что

$$d(x^0) \leq \varepsilon, \quad (3.7)$$

$$\beta(x^0) \leq \delta. \quad (3.8)$$

Если задача (3.2), (3.3) не имеет решения или $W(x^*) = 0$, то это можно определить при помощи полиномиального алгоритма. Назовем алгоритмом 3.1 метод, позволяющий определить существование решения задачи (3.2), (3.3). Пусть $M_0 = \{a_0\}$ - множество висячих вершин. Метод основан на определении веса пути из корня a_0 в каждую вершину $b \in M_0$, т.е. $v(L(a_0, b))$. Если $\max_{b \in M_0} v(L(a_0, b)) > V$, то задача (3.2), (3.3) не имеет решения. Алгоритм 3.1 является очевидной модификацией метода динамического программирования для задачи о

кратчайшем пути [3]. Оценка трудоемкости и памяти алгоритма 3.1 - $O(n)$. В случае, когда G - ациклический направленный граф, сложность задачи увеличивается и оценка алгоритма равна $O(n^2)$.

На основе алгоритма 3.1 возможна также проверка выполнения равенства $W(x^*) = 0$.

Утверждение 3.1. Если для любой вершины $a \in K \setminus a_0$ склеивание ее с вершиной $b \in P^+ a$ приводит к задаче (3.1), (3.2), для которой не существует решения, то в исходной задаче (3.1), (3.2) $W(x^*) = 0$. Если хотя бы для одной вершины $a \in K \setminus a_0$ склеивание ее с некоторой предшественницей приводит к задаче, для которой существует решение, то в исходной задаче $W(x^*) > 0$.

Доказательство утверждения 3.1 очевидно. Таким образом на основе утверждения 3.1 и алгоритма 3.1 просто строится полиномиальный алгоритм 3.2 для проверки выполнения равенства $W(x^*) = 0$ (например, за счет применения алгоритма 3.1 после проведения каждого склеивания вершин). Это верно и в случае, когда G - ациклический направленный граф.

Далее, если это не оговаривается особо, предполагается, что к задаче компоновки предварительно применяются алгоритмы 3.1 и 3.2 для проверки исходных условий.

Следует отметить, что при равенстве весов дуг, поставленная задача является часто рассматриваемой задачей минимизации числа модулей оверлейной структуры.

3.2 Простейшие частные случаи

Рассмотрим приближенные алгоритмы решения простейших частных случаев поставленной задачи, гарантирующие заданную относительную ошибку целевой функции в соответствии с (3.7).

В одном из простейших случаев, когда система состоит из основ-

ного сегмента и входящих подпрограмм, G представляет собой двух-
уровневое дерево. Пусть $\Gamma a_0 = \{a_1, \dots, a_i, \dots, a_m\}$, a_i - дуга
(a_0, a_i), $W(a_i) = w_i$. В этом случае $v(a_0, x) = v(a_0) + \sum_{i=1}^m x_i v(a_i)$,
 $v(a_0, x) = \max_{0 \leq x_i \leq 1} [(1-x_i)v(a_i)]$ и задача коммюны будет иметь сле-
дующий вид (постановка I): максимизировать $W(x) = \sum_{i=1}^m x_i w_i$ при ус-
ловиях

$$v(a_0) + \sum_{i=1}^m x_i v(a_i) + \max_{0 \leq x_i \leq 1} [(1-x_i)v(a_i)] \leq \bar{V}, x_i \in [0, 1]. \quad (3.9)$$

Рассмотрим для решения задачи (3.9) приближенный эффективный алго-
ритм (алгоритм 3.3), являющийся модификацией "алгоритма разбиения
на интервалы" для решения задачи о рюкзаке [7]. Схема изложения
этого и других приближенных алгоритмов аналогична использованной
в работе [9]. Будем искать допустимое решение $x^0 = (x_1^0, \dots, x_m^0)$
задачи (3.9) такое, чтобы выполнялось (3.7).

Пусть вершины множества Γa_0 переименованы так, что $v(a_1) \geq v(a_2) \geq \dots \geq v(a_m)$
при $1 \leq i < j \leq m$. Тогда очевидно следующее:

Утверждение 3.2 Для любого x

$$v(a_0, x) = v(a_{j_0}), \quad (3.10)$$

где j_0 - наименьший индекс среди индексов нулевых компонент век-
тора x .

Алгоритм поиска x^0 состоит из m шагов, на каждом из которых
формируется некоторое множество допустимых решений S^k ($k = 1, m$).
На 1-м шаге $S^1 = \{(0, 0, \dots, 0), (1, 0, \dots, 0)\}$. Пусть на $(k-1)$ -м
шаге получено множество S^{k-1} . Решения, входящие в S^{k-1} имеют вид
 $x^{k-1} = (x_1, \dots, x_{k-1}, 0, \dots, 0)$. Следующий k -й шаг состоит из шести
операций с множеством S^{k-1} : 1) "к-сдвиг": из каждого вектора
 $x^{k-1} \in S^{k-1}$ образуется вектор $\bar{x}^k = (x_1, \dots, x_{k-1}, 1, 0, \dots, 0)$; 2) "упо-
рядочение": все векторы $\bar{x}^k$ вместе с векторами x^{k-1} располагаются
в порядке убывания величин $W(x)$; 3) "вычисление ресурса": для
каждого вектора полученного множества вычисляется значение остав-

искомый ресурс

$$\rho(x) = V - v(a_0) - \sum_{i=1}^k x_i v(a_i) - \hat{v}(a_0, x), \quad 3.11$$

где $\hat{v}(a_0, x)$ вычисляется по формуле (3.10); 4) "первое выбрасывание": выбрасываются векторы x , для которых $\rho(x) < 0$; 5) "второе выбрасывание": из полученного упорядоченного множества выбрасываются те векторы, для которых имеются доминирующие (вектор x называется доминирующим для вектора y , если $W(x) \geq W(y)$ и $\rho(x) > \rho(y)$); 6) "разбиение на группы": пусть P^k - полученное упорядоченное множество векторов, $\rho = \max_{x \in P^k} W(x)$, разобьем интервал $[0, \rho]$ на $[m/\varepsilon]$ подинтервалов так, чтобы длина каждого из них не превосходила $[\varepsilon \rho / m]$, если в P^k имеется несколько векторов x , у которых $W(x)$ попадает в один и тот же подинтервал, то выбрасываем из P^k часть этих векторов с тем, чтобы в каждом подинтервале остался только один вектор x с наибольшим (в этом подинтервале) значением $\rho(x)$.

В результате остается искомое множество S^m . Выбрасывание векторов может привести к абсолютной погрешности, не превышающей $\varepsilon W(x^*)/m$. За m шагов алгоритма может наводиться относительная погрешность, не превышающая ε . Следовательно, искомым вектором является вектор из множества S^m с наибольшим значением величины $W(x)$. Заметим, что если $V < V(a_0) + \max_{i=1, \dots, k} v(a_i)$, то задача (3.9) не имеет решения, в противном случае $S^m \neq \emptyset$.

Приведенный алгоритм отличается от "алгоритма разбиения на интервалы" только операцией "вычисление ресурса", а точнее, определением и хранением индексов j_i . Вследствие утверждения 1 пересчет индекса j_0 необходимо производить только для вектора $\bar{x}$ при условии, что $x_i = 1 \quad \forall i = 1, k-1$, а это требует при работе алгоритма не более $O(m)$ действий. Таким образом, характеристики алгоритма 3.3

соответствует характеристикам "алгоритма разбиения на интервалы", т.е. оценка трудоемкости и памяти имеет вид $O(m^2/\epsilon)$.

Рассмотрим некоторые усложненные варианты задачи (3.9), решение которых потребуются далее. Первый вариант представляет собой задачу в постановке 2: максимизировать $W(x) = \sum_{i=1}^m x_i w_i$ при условиях

$$V(a_0) + \sum_{i=1}^m x_i v(a_i) \leq \bar{V}, \quad \max_{1 \leq i \leq m} [(1-x_i)v(a_i)] \leq \bar{V}^+, \quad (3.12)$$

$$x_i = 0 \vee 1.$$

Модифицируем алгоритм 3.3 для решения задачи (3.12): 1) заменим формулу вычисления оставшегося ресурса (3.11) на $\mu(x) = \bar{V} - V(a_0) - \sum_{i=1}^m x_i v(a_i)$; 2) введем перед третьей операцией в алгоритме 3.3 дополнительное выбрасывание векторов x , для которых $\hat{V}(a_0, x) > \bar{V}^+$, где $\hat{V}(a_0, x)$ вычисляется в соответствии с (3.10); 3) введем после шестой операции включение в множество $\bar{S}^k$ при $k = \bar{I}, m - \bar{I}$ вектора x такого, что $x_i = 1$ при $i = 1, k$ и $x_i = 0$ при $i > k$, если такой вектор не содержался в $\bar{S}^k$ и $\rho(x) \geq 0$. Таким образом, получим алгоритм 3.4 для решения задачи (3.12). Несложно показать, что оценки алгоритма 3.4 аналогичны оценкам алгоритма 3.3.

Рассмотрим второй вариант усложнения задачи (3.9) в виде постановки I: максимизировать $W(x) = \sum_{i=1}^m x_i w_i$ при условиях

$$V(a_0) + \sum_{i=1}^m x_i v^-(a_i) + \max_{1 \leq i \leq m} [(1-x_i)v^+(a_i) + v^+(a_i)] \leq \bar{V}, \quad (3.13)$$

$x_i = 0$ или 1 .

Эта задача соответствует случаю, когда $\forall a_i \in \Gamma a_0$ вес равен $v^-(a_i)$, каждая вершина a_i имеет одного последователя с весом $v^+(a_i)$ (это вес "хвоста" вершины $a_i - v(a_i)$) и возможно склеивание дуг вида $[a_i, a_i]$. При этом $V(a_0, x) = V(a_0) + \sum_{i=1}^m x_i v(a_i)$ и $\hat{V}(a_0, x) = \max_{1 \leq i \leq m} [(1-x_i)v^-(a_i) + v^+(a_i)]$. Пусть вершины множества Γa_0

перенумерованы так, что $v^-(a_i) + v^+(a_j) \geq v^-(a_j) + v^+(a_i)$ при $i < j \leq m$. Тогда очевидно следующее:

Утверждение 3.3 Для любого x

$$\hat{v}(a_0, x) = \max [v^-(a_{j_0}) + v^+(a_{i_0}), \max_{\substack{1 \leq i_0 \leq j_0 \\ i_0 \in \{i | x_i = 1\}}} v^+(a_{i_0})], \quad (3.14)$$

где j_0 определяется как в утверждении 1.

Модифицируем алгоритм 3.3 для решения задачи (3.13): 1) заменим формулу вычисления оставшегося ресурса (3.11) на

$$\rho(x) = V - v(a_0) - \sum_{i=1}^k x_i v^-(a_i) - \hat{v}(a_0, x), \quad (3.15)$$

где $\hat{v}(a_0, x)$ вычисляется по формуле (3.14); 2) введем после шестой операции включение в множество S^k при $k = 1, m-1$ векторы x такого, что $x_i = 1$ при $i = 1, k$ и $x_i = 0$ при $i > k$, если такой вектор не содержится в S^k и $\rho(x) > 0$. В результате имеем алгоритм 3.5 для решения задачи (3.13). Несложно показать, что его оценки аналогичны оценкам алгоритма 3.3.

Рассмотрим третий вариант - усложнение задачи (3.13) в виде постановки 2: максимизировать $W(x) = \sum_{i=1}^m x_i w_i$ при условиях

$$v(a_0) + \sum_{i=1}^m x_i v^-(a_i) \leq V^-, \quad \max_{1 \leq i \leq m} [(1-x_i)v^-(a_i) + v^+(a_i)] \leq V^+, \quad (3.16)$$

$x_i = 0$ или 1.

Модифицируем алгоритм 3.5 для решения задачи (3.16): 1) заменим формулу вычисления оставшегося ресурса (3.15) на $\rho(x) = V^- - v(a_0) - \sum_{i=1}^m x_i v^-(a_i)$; 2) введем перед третьей операцией дополнительное выбрасывание векторов x , для которых $\hat{v}(a_0, x) > V^+$, где $\hat{v}(a_0, x)$ вычисляется по формуле (3.14). Таким образом, получим алгоритм 3.6 для решения задачи (3.16). Несложно показать, что оценки алгоритма 3.6 аналогичны оценкам алгоритма 3.5 (соответственно и алгоритма 3.3).

На рис. 12 а - 12 д (слева) представлена графическая интерпре-

табря рассмотренных задач (включая задачу о рюкзаке).

3.3 Вспомогательные задачи

Рассмотрим некоторые вспомогательные задачи, которые потребуются при решении задачи компоновки в случае древовидной связи между модулями. Эти задачи могут иметь и самостоятельные приложения. Например, в главе 4 используется одна из приведенных в данном разделе моделей при определении содержания номенклатурного массива. Вспомогательные задачи являются усложнением лестничной задачи о рюкзаке; они заключаются не только в определении объединяемых модулей, но и в выборе варианта реализации модуля из набора вариантов. Введем для задач, рассматриваемых в данном разделе, набор булевых векторов $X = \{x_{ij} = (x_{ij}^1, x_{ij}^2, \dots, x_{ij}^q) \mid x_{ij}^j = 0 \text{ или } 1, j = \overline{1, q}, i = \overline{1, m}\}$. При этом в постановке задач необходимо учитывать следующие общие ограничения: $\sum_{j=1}^q x_{ij}^j = 1 \forall i; x_{ij}^j \leq x_{ij}^i \forall i, j$. Приведем "лестничную" задачу о рюкзаке: максимизировать $W(X) = \sum_{i=1}^m \sum_{j=1}^q x_{ij}^j w(a_{ij})$ при условиях $v(a_0) + \sum_{i=1}^m \sum_{j=1}^q x_{ij}^j v(a_{ij}) \leq V$. (3.17)

Для этой задачи разработан приближенный эффективный алгоритм с трудоемкостью $O(\frac{1}{\epsilon} \sum_{i=1}^m a_i)$ [29].

Приведем первую вспомогательную задачу:

максимизировать $W(X) = \sum_{i=1}^m \sum_{j=1}^q (x_{ij}^1 w(a_{ij}^1) + x_{ij}^2 w(a_{ij}^2))$ при условиях:

$$v(a_0) + \sum_{i=1}^m \sum_{j=1}^q x_{ij}^j v(a_{ij}^j) \leq V. \quad (3.18)$$

Задача (3.18) может иметь следующую интерпретацию. Пусть a_0 - головной сегмент с весом $v(a_0)$ и имеются вызываемые модули с номерами $1, \dots, i, \dots, m$. Для каждого модуля с номером i имеется множество вариантов реализации $\{a_{i1}, \dots, a_{iq}, \dots, a_{i1}, \dots, a_{iq}\}$. Каждый вариант a_{ij} может представлять собой взвешенный граф, в котором

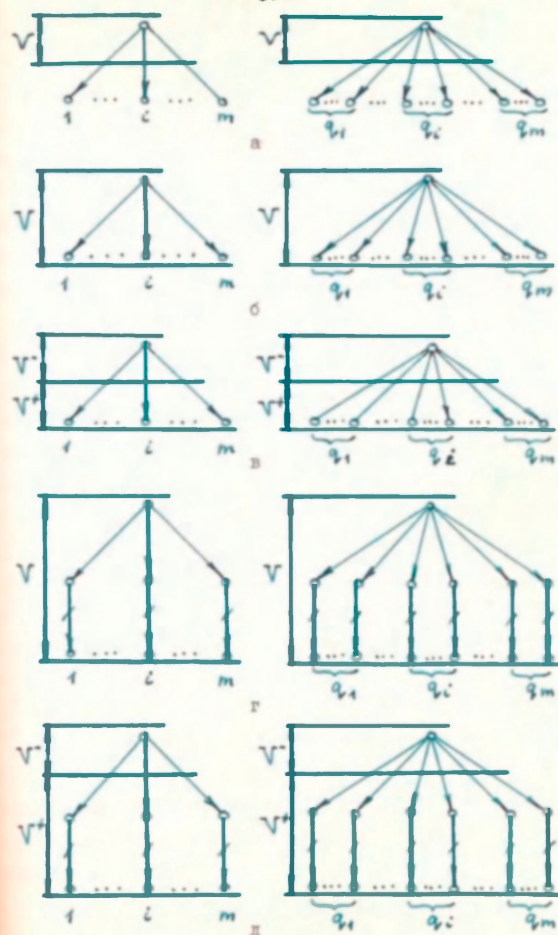


Рис. 12

осуществлено склеивание некоторых ветвей. Варианту a_{ij} соответствует вес $v(a_{ij})$, среднее число вызовов модулем $a_0 = w(a_{ij})$ и число $w(a_{ij})$, равное уменьшению среднего числа обращений к ИД при его работе. Требуется осуществить для каждого i выбор одного варианта реализации модуля ($x_{ij} = 1$, если выбирается вариант a_{ij} и 0 в противном случае) и присоединить его к a_0 ($x_{ij} = 1$) или не присоединить ($x_{ij} = 0$). Очевидно, что множество X определяет набор стягиваемых дуг $X = (x_1, \dots, x_n)$. Первое условие в (3.18) ограничивает объем головного сегмента и объединяемых с ним модулей.

Рассмотрим алгоритм решения задачи (3.18) - алгоритм 3.7, основанный на методе решения задачи (3.17). Будем искать допустимое решение X^0 такое, чтобы выполнялось соотношение типа (3.7), где вместо $x = X$, X^* - оптимальное решение задачи (3.18). Алгоритм поиска X^0 состоит из m шагов, на каждом из которых формируется множество допустимых решений S^k ($k=1, m$). Первым шагом $S^1 = \{X^1 | x_{ij} = (0,0) \forall i,j\}$. Пусть на $(k-1)$ -м шаге получено множество S^{k-1} . Решения, входящие в S^{k-1} имеют вид $X^{k-1} = (x_1, \dots, x_n)$ при $i > k$. Следующий k -й шаг состоит из семи операций с множеством S^{k-1} : 1) "1-й сдвиг": из каждого $X^{k-1} \in S^{k-1}$ формируется q_k множества Y_e^k ($e = 1, q_k$) таких, что в Y_e^k $x_{ke} = (1,0)$, а остальные векторы совпадают с векторами из X^{k-1} ; 2) "2-й сдвиг": из каждого $X^{k-1} \in S^{k-1}$ формируется q_k множества Z_e^k ($e = 1, q_k$) таких, что в Z_e^k $x_{ke} = (1,1)$, а остальные векторы совпадают с векторами из X^{k-1} ; 3) "упорядочение": все решения типа X^{k-1} , Y_e^k, Z_e^k располагаются в порядке возрастания величин $W(X)$; 4) "вычисление ресурса": для каждого элемента полученного упорядочения вычисляется значение оставшегося ресурса

$$\rho(X) = V - v(a_0) - \sum_{i=1}^k \sum_{j=1}^{q_i} x_{ij}^2 \cdot v(a_{ij}); \quad (3.19)$$

5) "первое выбрасывание": из полученного упорядочения выбрасываются те X , для которых $\rho(X) < 0$; 6) "второе выбрасывание": из полученного упорядочения выбрасываются те решения, для которых есть доминирующие (решение X называется доминирующим для решения Y , если $W(X) \geq W(Y)$ и $\rho(X) > \rho(Y)$); 7) "разбиение на группы": пусть P^* — полученное упорядочение решений, $p = \max_{X \in P^*} W(X)$, разобьем интервал $[0, p]$ на $\lceil m/\epsilon \rceil$ подинтервалов так, чтобы длина каждого из них не превышала $\lceil \epsilon p/m \rceil$; если в P^* имеется несколько решений X , у которых $W(X)$ попадает в один и тот же подинтервал, то из P^* выбрасывается часть этих множеств с тем, чтобы в каждом подинтервале осталось только одно решение с наибольшим (в этом подинтервале) значением $\rho(X)$. В результате остается искомое множество S^* . Выбрасывание решений может привести к абсолютной погрешности, не превышающей $\epsilon W(X^*)/m$. За m шагов алгоритма может накопиться относительная погрешность, не превышающая ϵ . Искомым является решение из множества S^m с наибольшим значением величины $W(X)$. Трудоемкость k -го ($k = 1, m$) шага алгоритма составляет $O(\frac{m}{\epsilon} q_k)$. Следовательно трудоемкость алгоритма $O(\frac{m}{\epsilon} \sum_{k=1}^m q_k)$, оценка памяти — $O(\frac{m}{\epsilon} \max_{k=1, \dots, m} q_k)$.

Рассмотрим усложненные варианты задачи (5.16). Первым вариантом представляется собой задачу в постановке типа 2: максимизировать

$$W(X) = \sum_{i=1}^m \sum_{j=1}^{q_i} (x_{ij}^+ w(a_{ij}) + x_{ij}^- w(a_{ij}^-)) \quad \text{при условиях}$$

$$v(a_0) + \sum_{i=1}^m \sum_{j=1}^{q_i} x_{ij}^+ v(a_{ij}) \leq \bar{V}^-; \quad \max_{i,j} [(x_{ij}^+ - x_{ij}^-) v(a_{ij})] \leq \bar{V}^+ \quad (3.20)$$

Модифицируем алгоритм 3.7 для решения задачи (3.20): 1) заменим формулу вычисления оставшегося ресурса (3.19) на

$$\rho(X) = \bar{V}^- - v(a_0) - \sum_{i=1}^k \sum_{j=1}^{q_i} x_{ij}^+ v(a_{ij}); \quad (3.21)$$

2) введем перед шестой операцией алгоритма 3.7 дополнительное

выбрасывание решений X , для которых

$$(x_{ij}^1 - x_{ij}^2) v(a_{ij}) > V^+ \quad (3.22)$$

Таким образом получим алгоритм 3.8 для задачи (3.20). Несложно показать, что оценки алгоритма 3.8 аналогичны оценкам алгоритма 3.7.

Рассмотрим второй вариант усложненной задачи 3.18 в виде постановки 2: максимизировать $W(X) = \sum_{i=1}^m \sum_{j=1}^n (x_{ij}^1 \bar{w}(a_{ij}) + x_{ij}^2 \bar{w}(a_{ij}))$ при условиях

$$v(a_0) + \sum_{i=1}^m \sum_{j=1}^n x_{ij}^2 v(a_{ij}) \leq V^-, \max\{x_{ij}^1 [(1-x_{ij}^2) v(a_{ij}) + v(a_{ij})], x_{ij}^2 v(a_{ij})\} \leq V^+ \quad (3.23)$$

Эта задача соответствует случаю, когда a_0 - головной сегмент, каждый вызываемый модуль имеет одно о последователя и возможно скливание a_0 с вызываемыми модулями.

Модифицируем алгоритм 3.7 для решения задачи (3.23): 1) заменим (3.21) на $\rho(X) = V^- - v(a_0) - \sum_{i=1}^m \sum_{j=1}^n x_{ij}^2 v(a_{ij})$; 2) заменим (3.22) на $x_{ij}^1 [(1-x_{ij}^2) v(a_{ij}) + v(a_{ij})] > V^+$.

В результате получим алгоритм 3.9 для решения задачи (3.23).

Очевидно, что оценки алгоритма 3.9 аналогичны оценкам алгоритма 3.7. Рассмотрим третью вспомогательную задачу (постановка типа I):

$$\text{максимизировать } W(X) = \sum_{i=1}^m \sum_{j=1}^n (x_{ij}^1 \bar{w}(a_{ij}) + x_{ij}^2 \bar{w}(a_{ij})) \text{ при условиях} \\ v(a_0) + \sum_{i=1}^m \sum_{j=1}^n x_{ij}^2 v(a_{ij}) + \max\{x_{ij}^1 [(1-x_{ij}^2) v(a_{ij}) + v(a_{ij})], x_{ij}^2 v(a_{ij})\} \leq V. \quad (3.24)$$

В качестве решения (3.24) будем искать такое X^0 , чтобы выполнялось соотношение типа (3.7) и (3.8). Введем $\tau = \frac{V^+}{V^- + V^+} + 1$, $\delta_0 = \frac{V^+}{V^- + V^+}$. Для решения задачи (3.24) рассмотрим аналогичную (3.23) при следующих правых частях ограничений на ресурс: вместо V^- поставим $V \delta_0 \ell$, вместо $V^+ = V \delta_0 (\tau - \ell)$, где $\ell = 0, \tau$.

Лемма 3.I. Если $\tau + 1$ раз решить задачу (3.23) при помощи алгоритма 3.9 при $\epsilon = 0, \tau$ с точностью ϵ и выбрать из полученного набора решений $\{X_\ell\}$ решение $\bar{X}$ с наибольшим значением $W(X)$, то это решение будет искомым $\bar{X}^*$ для задачи (3.24).

Доказательство леммы 3.I приводится в приложении I.

Назовем подход к решению задачи (3.24), содержащийся в лемме 3.I, алгоритмом 3.IO. Оценки его трудоемкости и памяти равны соответственно $O(\frac{m}{\epsilon} \sum_{i=1}^m a_i)$ и $O(\frac{m}{\epsilon} \max_{1 \leq i \leq m} a_i)$.

Вспомогательные задачи являются обобщением частных случаев, рассмотренных в предыдущем разделе. На рис. 12 а- 12 д (справа) представлена графическая интерпретация вспомогательных задач.

3.4 Древовидная связь между данными

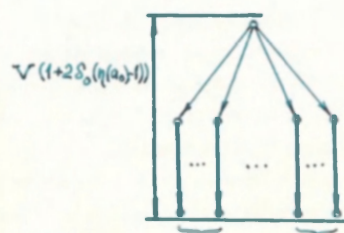
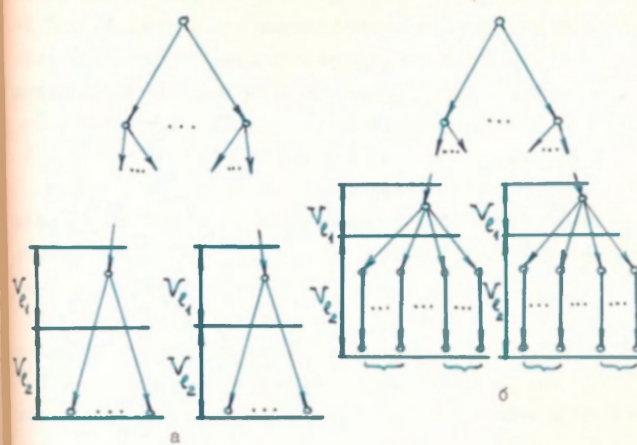
Пусть G является $(k+1)$ -уровневым деревом (a_0 -0-й уровень). Рассмотрим алгоритм 3.II для решения задачи компоновки программных модулей на дереве G . Введем для каждой вершины $a \in R$ следующие характеристики: $m(a) = |Ga|$, $\eta(a) = \{k_1, \dots, k_{m(a)} \mid Ga = \{a_{k_i}\}\}$. Выделим из R множества $M_0 = \{a \in R \mid Ga = \emptyset\}$, $M_1 = \{a \in R \mid Ga = \{a_1\}\}$, $M_2 = \{a \in R \mid Ga \subset (M_0 \cup M_1)\}$ и т.д.

Введем $\tau = \lfloor \frac{1}{\delta_1} \rfloor$, $\delta_1 = \frac{\epsilon}{2\eta(a_0)}$.
 $\epsilon = \frac{\epsilon}{\eta(a_0)} \cdot \tau \cdot \delta_1 = \tau \cdot \delta_1 \cdot \epsilon$.

Алгоритм 3.II.

1-й шаг. Сформулируем для каждой вершины $a_i \in M_1$, как для корневой, задачи (3.12) при $V^- = V_1, V^+ = V_2$ ($0 \leq \ell_1 \leq \tau+2, 0 \leq \ell_2 \leq \tau+2-\ell_1, \forall \ell_1$) и решим их при помощи алгоритма 3.4 с точностью ϵ . Перенумеруем полученные для каждой вершины a_i решения с помощью индекса j и введем следующие обозначения характеристик: $v^-(a_{ij}) = v(a_i, x)$, $v^+(a_{ij}) = v^+(a_i, x)$, $w(a_{ij}) = w(x)$, $w(a_{ij}) = w(\bar{v}, a_i)$, где $\bar{v} = \bar{v}^-(a_i), a_i \in M_1$ для всех i .

μ -й шаг ($\mu = 2, k-1$). Воспользовавшись результатами предыдущего



B

FIG. 13

шага, сформулируем для каждой вершины $a_i \in M_m$, как для корневой, задачу (3.23) при $V^+ = \bar{V}_a^+, V^- = \bar{V}_a^-$ ($0 \leq \ell \leq \tau + 2\eta(a_i)$, $0 \leq \ell \leq \tau + 2\eta(a_i) - 2$) и решим ее при помощи алгоритма 3.9 с точностью ε_0 . Перенумеруем полученные для каждой вершины решения индексом j и введем следующие обозначения характеристик $v^+(a_{ij}) = v(a_i; x)$, $v^-(a_{ij}) = \bar{v}(a_i; x)$, $w(a_{ij}) = W(X)$, $w(a_{ij}) = w(\delta, a_i)$, где $\delta = \tau + 2\eta(a_i)$ для всех i .

k -й шаг. Воспользовавшись результатами предыдущего шага, сформулируем для вершины a_0 задачу (3.24) при ограничении в первом условии, равном $V(1 + 2\delta_0(\eta(a_0) - 1))$ и решим ее при помощи алгоритма 3.10 с точностью ε_0 и ε_0^2 .

Работа алгоритма 3.11 иллюстрируется на рис. 13, где 1-му шагу соответствует рис. 13а, m -му шагу - 13б, k -му шагу - рис. 13в.

В результате получим решение $x^0 = (x_1^0, \dots, x_{n-1}^0)$. Обозначим компоненты вектора $x = (x_1, \dots, x_{n-1})$, относящиеся к дугам в поддереве G_n ($a \in R$) через $x(a)$.

Лемма 3.2. $v(x^0) \leq V + \delta^2 V$.

Доказательство леммы 3.2 приводится в приложении I.

Лемма 3.3. При решении задачи (3.12) для вершины $a \in M_1$ на 1-м шаге алгоритма 3.11 среди полученных вариантов имеется решение X такое, что для соответствующего ему $\tilde{x}(a)$ выполняется: $L(\tilde{x}(a)) \leq \varepsilon_0$.

Доказательство леммы 3.3 приводится в приложении I.

Лемма 3.4. При решении задачи (3.23) для вершины $a \in M_m$ на m -м ($m = 2, k-1$) шаге алгоритма 3.11 среди полученных вариантов имеется такое решение X , что для соответствующего ему $\tilde{x}(a)$ $L(\tilde{x}(a)) \leq \varepsilon_0 \eta(a)$.

Доказательство леммы 3.4 приводится в приложении I.

Лемма 3.5. При решении задачи (3.24) на k -м шаге алгоритма 3.11 получается решение X такое, что для соответствующего ему x^0

$$L(x^0) \leq \varepsilon_0 \eta(a_0).$$

Лемма 3.5 доказывается аналогично лемме 3.4.

Рассмотрим характеристики алгоритма 3.II. На I-м шаге к каждой вершине $a \in M$, порядка $\frac{1}{\delta}$ раз применяется алгоритм 3.4 с трудоемкостью $O(\frac{1}{\delta^2})$. Общая трудоемкость шага $O(\frac{1}{\delta^2} \sum_{a \in M} m^2(a))$. На μ -м шаге ($\mu = 2, k-1$) к каждой вершине $a \in M_\mu$ применяется алгоритм 3.9 с трудоемкостью $O(\frac{m(a)}{\delta^\mu} (\frac{1}{\delta} + 2\eta(a))^\mu)$. Общая трудоемкость шага μ равна $O(\frac{1}{\delta^\mu} \sum_{a \in M_\mu} m(a)(1+2\delta\eta(a))^\mu)$. На k -м шаге к a_0 применяется алгоритм 3.I0 с трудоемкостью $O(\frac{m(a_0)}{\delta^k} (1+2\delta\eta(a_0))^k)$. Таким образом, оценка трудоемкости алгоритма 3.II составляет $O(\frac{1}{\delta^k} \sum_{a \in M} m(a)(1+2\delta\eta(a))^k)$. Оценка памяти алгоритма 3.II - $O(\frac{1}{\delta^k})$, где $m = \max_{a \in M} m(a)$.

Следует заметить, что широкое применение имеет 3-х уровневые системы (монитор, функциональные программы, подпрограммы). В этом случае алгоритм 3.II имеет оценку трудоемкости $O(\frac{1}{\delta^3})$.

3.5. Некоторые частные случаи компоновки при одинаковых характеристиках программных модулей

При разработке реальных систем часто имеют место случаи, когда некоторые характеристики программных модулей равны. Это может быть связано с технологией разработки программного обеспечения. Например, в работе [32] рекомендуется построение сложных программных комплексов для АСУ на базе программных блоков, каждый из которых выполняет законченную функцию и имеет объем 500-1000 команд. Таким образом, на основе предположения равенства характеристик модулей, можно приближенно определять варианты компоновки на ранних этапах проектирования систем, например, на этапе составления технических заданий на программирование.

Рассмотрим задачу (3.8) в случае, когда веса вершин множества

Γa_0 равны v . Задача будет иметь вид: максимизировать $w(x) = \sum_{i=1}^m x_i w_i$ при условиях

$$v(a_0) + v \left[\sum_{i=1}^m x_i + \max_{1 \leq i \leq m} (1 - x_i) \right] \leq \bar{V}, \quad x_i = 0 \vee 1. \quad (3.25)$$

Для задачи (3.25) существует точный алгоритм (алгоритм 3.12) с трудоемкостью $O(m \log m)$. Необходимо упорядочить элементы Γa_0 в порядке невозрастания весов соответствующих дуг и последовательно присоединять их к корневой вершине, проверяя допустимость получаемого графа. Заметим, что данный алгоритм совпадает с "greedy" алгоритмом. В случае, когда в задаче (3.8) равны веса дуг, направленных в вершины Γa_0 , эффективным точным алгоритм аналогичен.

Рассмотрим постановку задачи компоновки в случае трехуровневого дерева. Обозначим $a \in K \setminus a_0$, w_a - вес дуги направленной в a ; x_a - булева переменная, равная 1, если вершина a объединяется с предшественницей, и 0 в противном случае. Тогда задача имеет вид:

максимизировать $\sum_{a \in K \setminus a_0} x_a w_a$ при условиях

$$v(a_0) + \sum_{a \in \Gamma a_0} x_a [v(a) + \sum_{b \in \Gamma a} x_b v(b) + \max_{a \in \Gamma a_0} \{ (1 - x_a) [v(a) + \sum_{b \in \Gamma a} x_b v(b) + \max_{b \in \Gamma a} \{ (1 - x_b) v(b) \} \}] \leq \bar{V};$$

$$x_a x_b = 0 \text{ или } 1.$$

(3.26)

Очевидным образом доказывается

Утверждение 3.4. Пусть в задаче (3.26) веса вершин множества $K \setminus a_0$ равны v . Если некоторая вершина $a \in \Gamma a_0$ объединена с u_a своими последователями, то не изменяя объема памяти и не уменьшая значения критерия можно перейти к аналогичному варианту компоновки, в котором в состав Γa_0 объединяемых с a вершин входят те последователи, у которых веса соответствующих дуг из a наибольшие.

Таким образом, с учетом равенства весов вершин множества $K \setminus a_0$ можно перейти к задаче: максимизировать $\sum_{a \in K \setminus a_0} x_a w_a$ при условиях

$$\sum_{a \in \Gamma a_0} x_a (y_a + 1) + \max_{a \in \Gamma a_0} \{ (1 - x_a) (y_a + 1) \operatorname{sign}[m(a) - y_a] \} \leq \frac{V - v(a_0)}{v} \quad (3.27)$$

$x_a = 0$ или 1 ;

где $\forall a \in \Gamma a_0, y_a = \sum_{b \in \Gamma a} x_b, m(a) = |\Gamma a|, 0 \leq y_a \leq m(a)$.

От задачи (3.27) можно перейти к эквивалентным. Введем $\forall a \in \Gamma a_0$ булеву переменную $z(y_a)$, которая равна 1 , если к a присоединяется y_a последователей (в порядке невозрастания весов направленных в них дуг) и 0 - в противном случае. Таким образом, $\forall a \in \Gamma a_0$ имеется $m(a) + 1$ вариантов, каждый из которых характеризуется числом присоединяемых последователей y_a и величиной $w(y_a)$, равной сумме весов стянутых при этом дуг. Тогда эквивалентная задача имеет вид:

максимизировать $\sum_{a \in \Gamma a_0} [x_a w_a + \sum_{y_a=0}^{m(a)} z(y_a) w(y_a)]$ при условиях

$$\sum_{a \in \Gamma a_0} \sum_{y_a=0}^{m(a)} x_a z(y_a) (y_a + 1) \leq \ell; \max_{a \in \Gamma a_0} \{ (1 - x_a) (y_a + 1) z(y_a) + \operatorname{sign}[m(a) - y_a] \} \leq \frac{V - v(a_0)}{v}; 0 \leq y_a \leq m(a); \sum_{y_a=0}^{m(a)} z(y_a) = 1 \quad \forall a \in \Gamma a_0; \quad (3.28)$$

$x_a, z(y_a) = 0$ или 1 ; где $\ell = 0, \lceil (V - v(a_0)) / v \rceil$.

Задача (3.28) при фиксированном ℓ соответствует (с точностью до слагаемого $\operatorname{sign}[m(a) - y_a]$ в ограничении, равного 1 или 0) задаче (3.20), представленной графически на рис. 12в (справа), для которой существует эффективный приближенный алгоритм, гарантирующий заданную относительную погрешность целевой функции в соответствии с (3.7) - алгоритм 3.8. Несложная модификация этого алгоритма (алгоритм 3.13), учитывающая отличия в одном ограничении задачи (3.28), позволяет аналогично решать последнюю задачу. Оценки трудоемкости и памяти этого алгоритма $- O(\frac{|\Gamma a_0|}{\varepsilon} \sum_{a \in \Gamma a_0} m(a)) \leq O(\frac{n^2}{\varepsilon})$. Для решения задачи (3.27) необходимо решить задачу (3.28) при всех значениях ℓ и выбрать решение с наибольшим значением целевой функции. При этом оценка трудоемкости и памяти метода $O(\frac{n^2}{\varepsilon})$.

3.6 Задача на ациклическом направленном графе

В предыдущих разделах настоящей главы была сформулирована задача компоновки данных и предложены эффективные приближенные алгоритмы для различных случаев древовидной связи между данными. В практике связи между данными часто имеют более сложный вид. Решение задачи в этих ситуациях должно основываться на учете особенностей исходных параметров модели и связано с использованием эвристических подходов: декомпозицией, выявлением "узких мест" и т.п. Применение таких подходов целесообразно как на этапе постановки задачи, так и в процессе решения. При этом к получающимся моделям могут быть применены алгоритмы из числа предложенных выше. Примером декомпозиционного подхода является работа [37], в которой среди других рассматривается задача оптимизации структуры данных за счет объединения элементов данных в группы с целью уменьшения времени на чтение-запись информации.

Модель, когда $G = (R, \Gamma)$ - корневой ациклический направленный граф, в большей степени, чем древовидная соответствует реальным сложным программным комплексам [20, 33]. Пусть $G_a = (R_a, \Gamma_a)$ - корневой ациклический направленный подграф с корнем в $a \in R$ (в R_a кроме a входят все вершины из R , достижимые из a). Естественным образом на случай ациклического направленного графа распространяются определения 3.2, 3.3, 3.4.

Определение 3.5 [3]. Назовем вершину a доминатором вершины b ($a, b \in R$), если любой путь из корня в b проходит через a . Доминатор вершины b , ближайший к ней и отличный от нее, назовем непосредственным доминатором и обозначим $I(b)$.

Дополним граф G дугами следующим образом. Направим дугу от a к b ($a, b \in R$), если существует G_c ($c \in R$) такой, что $a, b \in R_c$ и $a = I(b)$ в G_c . Назовем дугу (a, b) доминаторной и обозначим мно-

кество доминаторных дуг через D . Припишем дуге (a, b) вес

$$w(a, b) = \sum_{\substack{a_i \in R_c \\ a_i \in \Gamma^{-1}b \\ (a_i, b) \in D}} w(a_i, b). \quad (3.29)$$

Если в G имела дуга от a к b , то исключим ее. В результате введения в граф G всех возможных доминаторных дуг, получим новый граф (будем обозначать его, как и прежде, $G = (R, \Gamma)$), причем Γ расширилось, т.к. $\Gamma := \Gamma + \tilde{D}$.

Задача компоновки в случае, когда G - ациклический направленный граф, представляет собой сложную комбинаторную проблему. Рассмотрим эвристический подход к ее решению, основанный на учете особенностей исходных параметров модели.

Связи между программными модулями часто не однородны. Например, реорганизация базы данных в информационных системах производится значительно реже, чем ввод, корректировка, поиск. Таким образом, веса дуг, направленных из вершины, соответствующей монитору, в модули, реализующие функции реорганизации, крайне малы. Такие ситуации могут иметь место и в других случаях (преимущественное использование модулей обмена с МД по сравнению с модулями обмена с МЛ). Сравнительный анализ исходных параметров модели может показать, что пренебрежение некоторыми связями практически не повлияет на значение критерия. С другой стороны, могут иметь место чрезвычайно сильные связи например, частота использования модулей поиска практически во всех информационных системах чрезвычайно высока. В этом случае целесообразно исключить сильные связи из дальнейшего рассмотрения за счет присоединения часто используемых модулей к соответствующим вызывающим. В результате последовательного осуществления указанных упрощений исходной задачи можно получить достаточно простую модель в которой величины, характеризующие связь между модулями имеют один порядок. Последнее обстоятельство не позволит

дальнейшее упрощение модели указанными двумя методами, но за счет простоты получившейся модели возможно применение разработанных алгоритмов.

Приведем методику решения задачи компоновки, основанную на рассмотренном подходе.

I. Введение в модель множества доминаторных дуг (с расчетом их весов).

II. Анализ корректности задачи. На этом этапе осуществляется проверка существования решения (алгоритм 3.1), проверка выполнения равенства $W(x^*) = 0$ (алгоритм 3.2).

III. Выявление слабых связей между модулями. Этот этап осуществляется следующим образом. Можно выделить множество дуг $U_1 = \{u_i \in U \mid w(u_i) \leq \xi_1, \sum_{u_i \in U} w(u_i) = 1\}$, где величина коэффициента ξ_1 , например, имеет порядок $10^{-4} - 10^{-2}$. Дуги, принадлежащие множеству U_1 , не рассматриваются далее при выборе стягиваемых дуг.

IV. Выявление сильных связей между модулями. По аналогии с предыдущим этапом можно выделить множество дуг $U_2 = \{u_i \in U \mid w(u_i) \geq \xi_2, \sum_{u_i \in U} w(u_i) = 1\}$, где величина коэффициента ξ_2 имеет порядок $10^{-2} - 10^{-1}$. Далее осуществляется последовательное стягивание дуг из множества U_2 .

При проведении преобразований исходной задачи на III-IV этапах можно получить модель, для решения которой пригодны алгоритмы 3.3-3.13. В этом будет заключаться этап V. Возможен случай, в котором получится модель с $W(x^*) = 0$, что свидетельствует о получении некоторого решения исходной задачи.

Следует подчеркнуть, что на каждом этапе рассмотренной методики важно участие разработчика системы (или специалиста по организации эксплуатации), т.к. он может учесть неформализованные факторы и вмешиваться в процесс решения.

Выводы к главе 3

1. Предложена формальная постановка экстремальной задачи организации данных в случае древовидной связи между группами данных (модулями). Задача рассмотрена на примере выбора структуры системы программных модулей, входящих в некоторый комплекс. В качестве целевой функции рассматривается максимально возможное снижение среднего числа обменов между ОП и ВП, а в качестве ограничения - объем ОП, предоставляемый системе.
2. Анализ задачи и построение эффективных алгоритмов ее решения основывается на переходе от простейших постановок ко все более сложным. При этом для некоторых вариантов предложены точные эффективные алгоритмы, для других - эффективные приближенные с гарантированной относительной погрешностью целевой функции, для более сложных - эффективные приближенные алгоритмы с гарантированной относительной погрешностью целевой функции и невязки ограничения на объем ОП.
3. Предложен эвристический подход, позволяющий получать решение задачи компоновки в случае, когда связь между модулями имеет вид ациклического графа, или сводить исходную задачу к тем вариантам постановки, для которых предложены эффективные методы решения (точные или приближенные).

Глава 4. Распределение вычислительных ресурсов на примере подсистемы АСУ-Росгеология

В данной главе приводятся примеры применения результатов проведенного исследования (моделей и алгоритмов) для рационального использования ресурсов ЭВМ при организации функционирования подсистемы АСУ-Росгеология. Приведены описания системы и решение задачи компоновки программных модулей.

4.1 Назначение подсистемы

Рассматриваемая подсистема АСУ-Росгеология предназначена для выполнения двух функций: регистрации и выработки плановых решений. Схема преобразования информации в подсистеме представлена на рис.14.

В создании подсистемы в первую очередь были разработаны структура информационной базы и программное обеспечение для реализации первой функции [25]. Информационные массивы соответствуют наборам данных. С целью повышения оперативности работы системы массивы размещены на МД и использован прямой метод доступа. Хранение информации на МД позволяет в случае необходимости осуществлять реорганизацию информационной базы. Подсистема ориентирована на использование моделей ЕС ЭВМ, начиная с ЕС 1040. Программное обеспечение работает под управлением ДОС ЕС. Ядро программного комплекса представляет собой в определенной степени универсальное средство для управления базой данных. На основе подключения к ядру дополнительных программных модулей (принцип вертикальных слоев [49]) осуществлялось расширение функций подсистемы, включая такие функции управления, как многовариантные оптимизационные плановые расчеты [40]. Программный комплекс имеет модульную структуру. При разработке программных модулей, реализующих обмен с внешними устройствами,

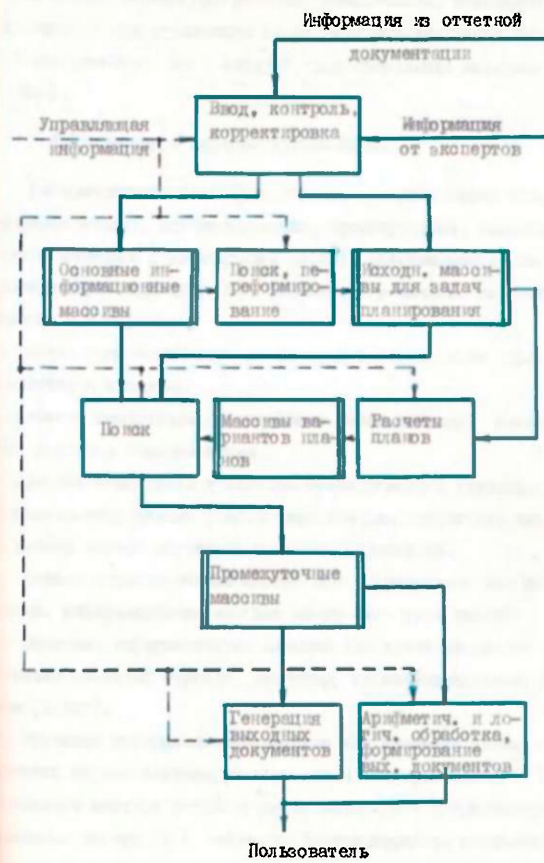


Рис. 14

организацию работы программного обеспечения, используя язык Ассемблер, при реализации алгоритмов решения задач математического программирования - Фортран, для оформления выходных документов - ПЛ/Г.

4.2 Структура информационной базы.

Информационная база (рис.15) представляет собой совокупность информационных, вспомогательных, промежуточных, служебных массивов и массивов пользователей [2,25]. Информационные массивы имеют позиционную структуру [5], основные ограничения на которую заключаются в следующем:

- запись (документ) состоит из последовательности показателей (простых и сложных),
- сложный показатель представляет собой матрицу, элементами которой являются подпоказатели,
- простой показатель и подпоказатель являются наименьшей единицей именованных данных (набор символов или десятичное число),
- размер записи ограничен размером дорожки МД.

Каждый информационный массив имеет уникальное имя из пяти символов. Информационные массивы могут быть трех типов:

1. Основные информационные массивы для хранения данных о функционировании объектов отрасли, например, по месторождениям, по структурам (XXXX).

2. Исходные информационные массивы для хранения данных, на основе которых осуществляется решение задач планирования (YYYYY). Записи исходного массива разбиваются на варианты - последовательно расположенные наборы из k записей. Записи варианта соответствуют объектам отрасли, при этом одна запись соответствует министерству, а ($k - 1$) - территориально-геологическим организациям (ТГО).

3. Результирующие информационные массивы для хранения вариантов оптимальных планов (ZZZZZ).

Для хранения вспомогательной информации по каждому информационному массиву служат вспомогательные массивы:

- массив шаблона, содержащий структуру записи информационного массива и характеристики показателей;
- массив наименований, содержащий названия простых и сложных показателей, строк и столбцов сложных показателей;
- поисковый массив для организации двухэтапного поиска (комбинированная схема организации поиска [41]), содержащий выбранные из каждой записи информационного массива элементы значений задаваемого перечня показателей;
- массив вариантов, содержащий данные о характере разбиения информационных массивов на варианты.

Первые символы имен вспомогательных массивов совпадают с именем соответствующего информационного массива, а шестой имеет фиксированное значение: S - для массива шаблона, N - для массива наименований, L - для поискового массива, W - для массива вариантов. Массивы наименований и шаблона далее будут называться массивами описания данных. Использование поисковых массивов и массивов вариантов не обязательно. На рис. 15 показано, что при работе с основными информационными массивами применяются поисковые массивы, а с исходными - массивы вариантов. Это связано с тем, что поиск данных в основных информационных массивах большого объема (порядка 1000 до-рожек) необходимо осуществлять при помощи двухэтапной процедуры с использованием поисковых массивов. Разбиение исходных массивов (YYYY) на варианты обусловлено технологией проведения оптимизационных плановых расчетов, в процессе которых пользователям необходимо создавать конкурирующие варианты исходных данных.

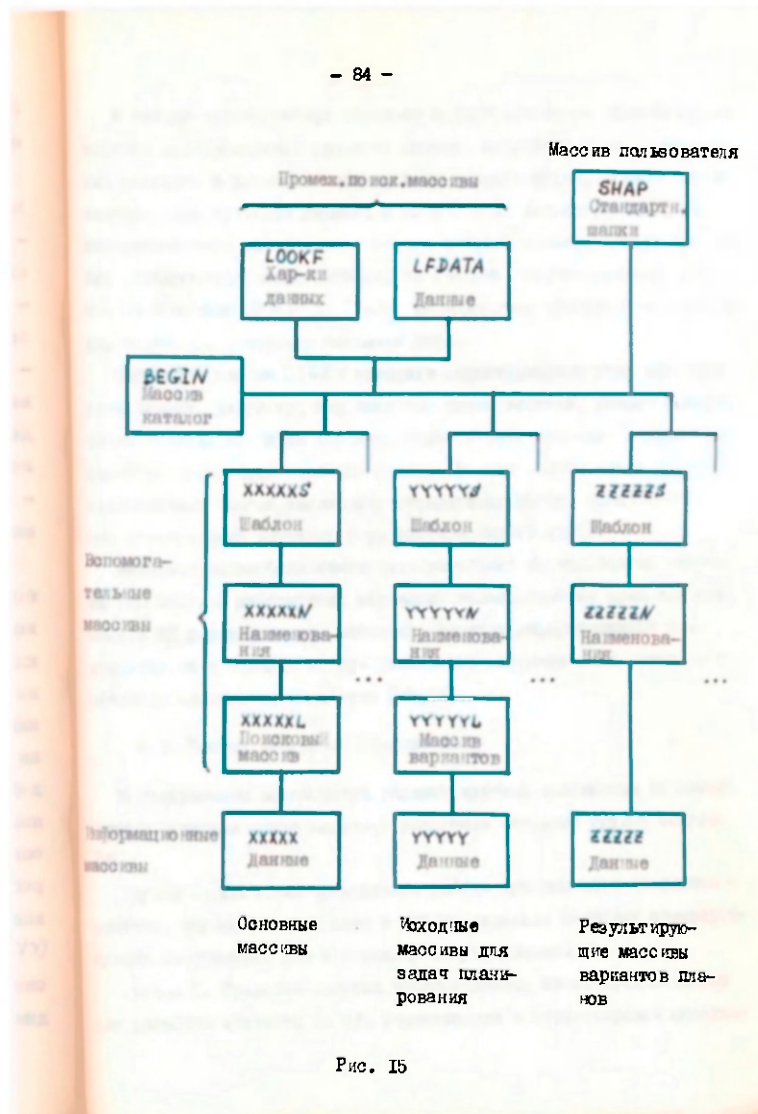


Рис. 15

В состав промежуточных массивов входят поисковые промежуточные массивы для временного хранения данных, выбранных из информационных массивов в процессе поиска, и их характеристик. *LFDATA* предназначен для хранения данных, а *LOOKF* - их характеристик (имя информационного массива из которого выбраны данные: номера выбранных показателей; число записей, из которых выбраны данные). Данные из массивов *LOOKF* и *LFDATA* используются программами обработки, например, генерации выходных форм.

Служебный массив *BEGIN* содержит характеристики всех массивов базы данных, например, имя массива, число записей, размер записи, число записей на одной дорожке, адрес первой дорожки, записи массивов располагаются последовательно. Массив пользователя *SNAP* представляет собой библиотеку стандартных шапок, используемых при формировании выходных форм фиксированного вида.

Информационная база может быть размещена на нескольких пакетах МД (до 255). В действующих вариантах использовались один или два пакета МД для размещения массивов. Вариант информационной базы представлен в таблице 4. При размещении массивов была учтена возможность совместной обработки [25,26].

4.3. Состав программного комплекса

В программном обеспечении рассматриваемой подсистемы по выполняемым функциям можно выделить следующие основные группы модулей [25]:

Группа 0. Средства организации работы программного комплекса - монитор, осуществляющий ввод и анализ задания, загрузку соответствующих программных фаз и передачу им управления.

Группа I. Средства запуска информационной базы. Предназначены для разметки экстенда на МД, формирования и корректировки массива-

Таблица 4.

№	Имя массива	Размер записи	Число записей на дорожке	Номер тома	Общее число записей
0	BEGIN	3200	I	0	I
I	MESTOS	3200	I	0	I
2	MESTOM	3200	I	0	3
3	MESTOL	3200	I	0	I2
4	MESTO	3200	I	0	I500
5	LOOKF	3200	I	0	50
6	LFDATA	3200	I	0	50
7	SHAP	3200	I	0	50
8	STRUCS	3200	I	I	I
9	STRUCN	3200	I	I	3
10	STRUCL	3200	I	I	I2
11	STRUC	3200	I	I	I800
12	EFFECTS	3200	I	I	I
13	EFFECTN	3200	I	I	3
14	EFFECTL	3200	I	I	I2
15	EFFECT	300	10	I	200

каталога, массивов описания данных и вариантов.

Группа 2. Средства ведения информационной базы. Включают в свой состав модули ввода, поиска, корректировки, контроля, копирования, получения справок о состоянии информационной базы.

Группа 3. Средства арифметической и логической обработки информации и формирования выходных документов. Предназначены для обработки данных по типовым алгоритмам и генерации выходных документов.

Группа 4. Средства математической обработки. представляют собой программные модули, реализующие алгоритмы решения задач математического программирования.

Группа 5. Средства обмена с внешними устройствами. Включают в свой состав модули обмена между ОП и внешними устройствами.

При проектировании программного комплекса была реализована возможность поиска в каждом из информационных массивов документа по поисковому предписанию, представляющему собой перечень условий, объединяемых логическими функциями дизъюнкции и конъюнкции. Условия имеют вид:

$$N \Theta Z$$

где N - номер показателя (простого или подпоказателя) в документе массива, Θ - знак ($=, \neq, >, \geq, <, \leq$), Z - значение.

Из документа, соответствующего поисковому предписанию, в процессе поиска выбираются значения заданного набора показателей. Таким образом, поиск документа может осуществляться по значению любого элементарного данного из документа. Такое требование к системе связано с возможной адаптацией к различным задачам планирования, информационного обеспечения руководства.

При некоторых вариантах использования программного комплекса объемы информационных массивов велики, поэтому была реализована

следующая двухэтапная процедура поиска данных. Для каждого информационного массива XXXX может быть организован вспомогательный поисковый массив XXXXL, содержащий элементы значений тех данных, по которым может производиться поиск документов, т.е. которые входят в состав условия поисковых предписаний. При предварительном просмотре массива XXXX в процессе поиска производится, как правило, отсев значительного числа документов информационного массива (порядка 90%). Применение двухэтапной процедуры поиска на основе предварительного просмотра вспомогательного поискового массива позволило уменьшить время одной операции поиска с нескольких минут до нескольких секунд.

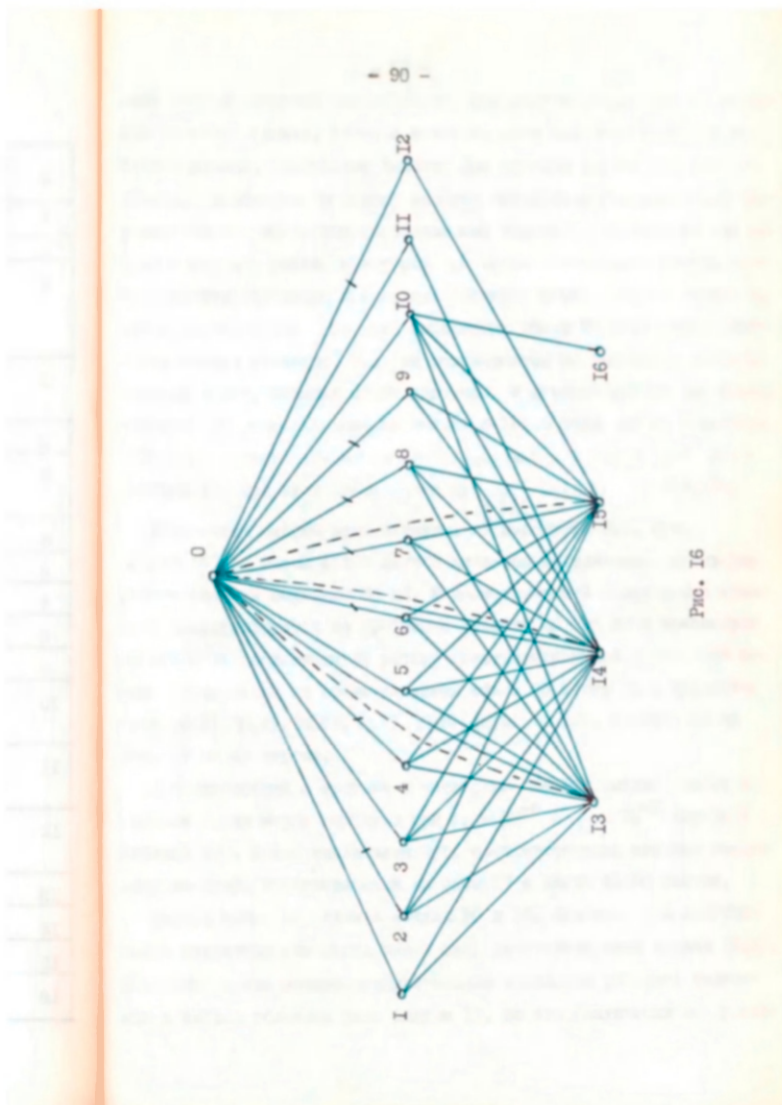
В процессе проектирования и эксплуатации подсистемы происходит совершенствование и дополнение программного обеспечения, например, число модулей доведено до 40. Возможны использования усеченных вариантов программного обеспечения для реализации конкретных функций АСУ: проведение оптимизационных расчетов по фиксированной задаче планирования, информационное обеспечение руководства. Постановка и решение задачи выбора структуры комплекса при функционировании подсистемы далее будут рассмотрены на примере усеченного варианта программного обеспечения. Состав используемого при этом программного комплекса приведен в таблице 5. Графовая модель комплекса представлена на рис. 16 (пунктирные линии соответствуют доминаторным дугам).

4.4 Выбор варианта структуры программного комплекса

Для определения варианта компоновки программных модулей системы, состав которой приведен в таблице 5, применяется методика из параграфа 3.6. Исходная информация для решения задачи компоновки представлена в таблицах 6-18. Данные соответствуют примерно гсдс-

Таблица 5.

№	Имя модуля	Назначение	Объем в байтах
1	2	3	4
0	EOJ	Организация работы комплекса	300
1	BEGIN	Разметка экстенда на МД, формирование и корректировка массива-каталога	4000
2	SHABLOW	Формирование массивов описания данных	12000
3	PROB	Ввод данных	7200
4	FIND	Формирование поисковых массивов	8000
5	CORR	Корректировка данных	7500
6	COR1	Ускоренная корректировка данных	8000
7	LOOK	Поиск данных	12000
8	LOK1	Ускоренный поиск данных	12500
9	FORMA	Генерация выходных форм	14000
10	COPY	Копирование данных с МД на МЛ, с МЛ на МД	7500
11	BEG	Формирование справок о состоянии базы данных	3500
12	SPRAV	Формирование фиксированных выходных документов	40000
13	DATA	Ввод данных с перфокарт	500
14	PCN	Вывод данных из ОП на АПУ	1000
15	DISK	Обмен между ОП и МЛ	1800
16	TAPE	Обмен между ОП и МЛ	2000



вому периоду опытной эксплуатации. Для модулей формирования массивов описания данных, ввода и корректировки выделены варианты исходных данных, содержащие ошибки. Для модулей поиска выделен вариант, при котором по одному запросу осуществляется несколько операций поиска информации (многошаговый запрос). Аналогично для модуля 9 выделен режим, в котором при одном обращении к модулю осуществляется генерация нескольких выходных форм. Следует отметить, что этап подготовки исходной информации имеет большое (даже самостоятельное) значение, т.к. он основывается на выявлении основных классов задач, стоящих перед системой, и функциональном построении модулей. На основе данных из таблиц 6-18 по формулам из параграфа 3.1 несложно рассчитываются характеристики дуг графа (рис. 16) - таблица 19. При этом сумма весов дуг $\sum_{a \in G} w(a, b) = 4353,335$.

Рассмотрим второй этап методики из параграфа 3.6. Путь $\langle 0, 12, 15 \rangle$ с весом 42100 байтов является критическим. Пусть для работы системы выделено 50 кб. Рассматриваемый программный комплекс эксплуатируется на ЕС-1033 с объемом ОП 256 кб и теснования на объем ОП соответствуют режиму мультипрограммной работы. На основе ограничения на представляемый объем ОП могут быть исключены дуги $(0, 2), (0, 4), (0, 6), (0, 7), (0, 8), (0, 9), (0, 12)$, помеченные на рис. 16 одной чертой.

В соответствии с третьим и четвертым этапами выявим слабые и сильные связи между модулями при $\xi_1 = 10^{-2}$ и $\xi_2 = 10^{-2}$ (графа 5 таблицы 19). После стягивания дуг, соответствующих сильным связям получим граф, представленный на рис. 17 с весом 45400 байтов.

Теперь можно объединить модули 10 и 16. Заметим, что получившаяся структура комплекса имеет вид, соответствующий задаче (3.9). В данном случае возможно единственное очевидное решение: включение в состав головной фазы модуля 11, но это фактически не умень-

Таблица 6

Номер варианта j	Назначение варианта	Частота варианта P_j	Среднее число вызовов модуля i модулем 0 α'_{ji}													
			$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$	$i=9$	$i=10$	$i=11$	$i=12$	$i=13$	$i=14$
1	Разметка МД	0,005	I										I		2	II
2	Корректировка разметки МД	0,005	I										2		3	II
3	Формирование на МД массивов описания данных	0,0075		I											I	IO
4	Ввод данных	0,5			I										I	IO
5	Получение контрольных справок	0,125							I		I				2	II
6	Корректировка данных	0,1					I	I	I	I	2				6	IS
7	Создание поисковых массивов	0,005				I									I	IO
8	Копирование данных	0,0025										I			I	IO
9	Получение справок	0,25							8	8	IO			5	3I	40

Таблица 7

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 1	
			$i = I3$	$i = I4$
1	Разметка МД	0,5	5	5
2	Корректировка размет- ки МД	0,5	4	4

Таблица 8

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 2		
			$i = I3$	$i = I4$	$i = I5$
1	Исходные данные без ошибок	0,5	100	100	4
2	Исходные данные с ошибками	0,5	100	100	0

Таблица 9

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 3		
			$i = I3$	$i = I4$	$i = I5$
1	Исходные данные без ошибок	0,75	20	20	6
2	Исходные данные с ошибками	0,25	20	20	0

Таблица I0

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 4 d_j^4		
			$i=13$	$i=14$	$i=15$
I	Формирование поискового массива	I	5	6	1000

Таблица II

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 5 d_j^5		
			$i=13$	$i=14$	$i=15$
I	Запрос без ошибок	0,8	20	22	1000
2	Запрос с ошибками	0,2	20	22	0

Таблица I2

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 6 d_j^6		
			$i=13$	$i=14$	$i=15$
I	Запрос без ошибок	0,8	20	22	7
2	Запрос с ошибками	0,2	20	22	0

Таблица I3

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 7 d^j_i		
			$i = I3$	$i = I4$	$i = I5$
1	Одношаговый запрос	0,5	10	10	1000
2	Многошаговый запрос	0,5	20	20	2000

Таблица I4

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 8 d^j_i		
			$i = I3$	$i = I4$	$i = I5$
1	Одношаговый запрос	0,5	10	10	7
2	Многошаговый запрос	0,5	20	20	14

Таблица I5

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 9 d^j_i	
			$i = I4$	$i = I5$
1	Генерация одной таблицы	0,5	50	6
2	Генерация несколь- таблиц	0,5	100	10

Таблица I6

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 10 d^j_{10}			
			$i=12$	$i=14$	$i=15$	$i=16$
1	Перезапись ин- формационной базы	0,5	2	2	1000	1000
2	Перезапись ин- формационных массивов	0,5	2	2	700	700

Таблица I7

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 11 d^j_{11}	
			$i=14$	
1	Получение спра- вки об информа- ционной базе	1	20	

Таблица I8

Номер варианта j	Назначение варианта	Частота варианта P_j	Вызовы модулем 12 d^j_{12}	
			$i=13$	$i=15$
1	Генерация фик- сированных справок	1	3	10

Таблица 19.

№	Дуга	Тип	Вес	Тип связи
I	2	3	4	5
I	0,1	-	0,01	слабый
2	0,2	-	0,0075	слабый
3	0,3	-	0,5	слабый
4	0,4	-	0,005	слабый
5	0,5	-	0,1	слабый
6	0,6	-	0,1	слабый
7	0,7	-	2,225	слабый
8	0,8	-	2,1	слабый
9	0,9	-	2,825	слабый
10	0,10	-	0,0025	слабый
11	0,11	-	0,015	слабый
12	0,12	-	1,25	слабый
13	0,13	-	8,89	
14	0,14	-	18,14	
15	1,13	-	0,045	слабый
16	1,14	-	0,045	слабый
17	2,13	-	0,75	слабый
18	2,14	-	0,75	слабый
19	2,15	-	0,015	слабый
20	3,13	-	10	
21	3,14	-	10	
22	3,15	-	2,25	слабый
23	4,13	-	0,025	слабый
24	4,14	-	0,03	слабый

1	2	3	4	5
25	4, I5	-	50	СИЛЬНЫЙ
26	5, I3	-	2	СЛАБЫЙ
27	5, I4	-	2,2	СЛАБЫЙ
28	5, I5	-	30	СИЛЬНЫЙ
29	6, I3	-	2	СЛАБЫЙ
30	6, I4	-	2,2	СЛАБЫЙ
31	6, I5	-	0,56	СЛАБЫЙ
32	7, I3	-	33,375	
33	7, I4	-	33,375	
34	7, I5	-	3337,5	СИЛЬНЫЙ
35	8, I3	-	31,5	
36	8, I4	-	31,5	
37	8, I5	-	22,5	
38	9, I4	-	201,875	СИЛЬНЫЙ
39	9, I5	-	22,6	
40	10, I3	-	0,005	СЛАБЫЙ
41	10, I4	-	0,005	СЛАБЫЙ
42	10, I5	-	2,125	СЛАБЫЙ
43	10, I6	-	2,125	СЛАБЫЙ
44	11, I4	-	0,3	СЛАБЫЙ
45	12, I3	-	3,75	СЛАБЫЙ
46	12, I5	-	12,5	
47	0, I3	И	92,34	СИЛЬНЫЙ
48	0, I4	Л	300,41	СИЛЬНЫЙ
49	0, I5	И	3530,05	СИЛЬНЫЙ

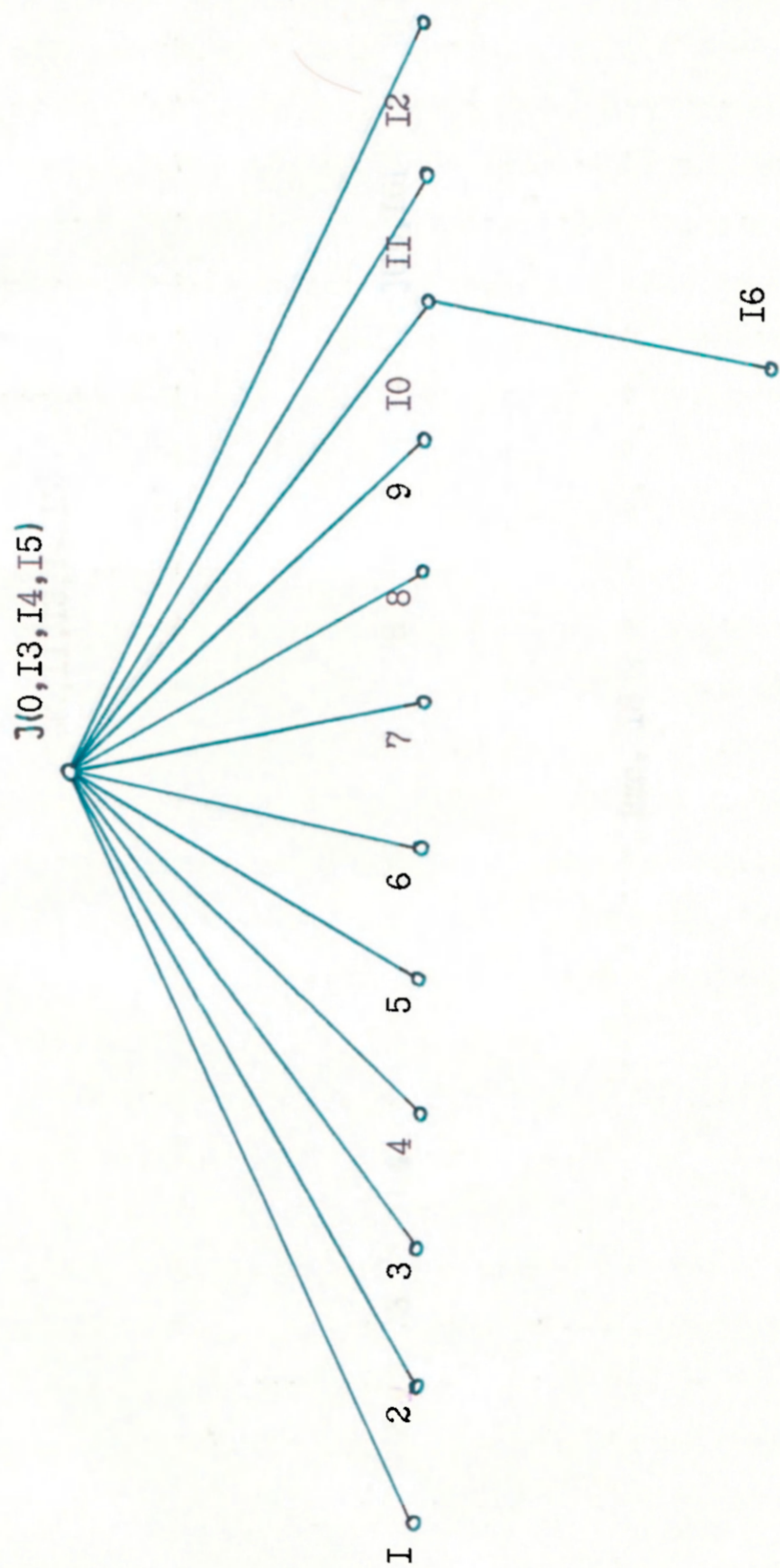


Рис. 17



Рис. 18

лит среднее число обменов. Таким образом, получим граф, представленный на рис.18, со средним общим числом вызовов модулей из ВП равным 9,125. Значение целевой функции на полученном решении равно 4344,71. Приведенные две величины соответствуют одному обращению к рассматриваемому программному комплексу. При времени обращения к устройствам на МД порядка 2×10^{-3} сек выигрыш составляет порядка 8 секунд. При числе обращения в рассматриваемом примере равном 5000 выигрыш составляет порядка 11 часов. При действующей в настоящее время цене на 1 час ЭВМ ЕС-1033 это соответствует 1210 рублям.

Следует отметить, что возможно использование и других решений задачи, при которых значение целевой функции незначительно отличается от приведенного. Основанием для выбора решения могут являться такие факторы, как возможное направление расширения программного обеспечения. Например, если рассматриваемый вариант программного комплекса предполагается дополнить модулями, использующими операцию обмена между ОП и МД, то вместо модуля II в состав 10-й фазы следует включить модуль I6. Значение целевой функции изменится при этом по сравнению с решением на рис.18 меньше, чем на 10-3%.

приведенный в настоящем параграфе пример решения задачи выбора варианта структуры (компоновки модулей) и связанных с этим анализ функционирования программного комплекса не требуют значительных трудозатрат. В то же время, кроме указанного эффекта, это позволяет разработчикам и специалистам по эксплуатации систем в значительной степени структуризовать стоящие перед ними проблемы. Например, в результате можно прийти к такому выводу: некоторые программные модуль (или группа модулей) редко используется, поэтому целесообразно заменить его более простым (требующим меньшего объе-

ма ОП или исключить. Особенно важное значение анализ структуры программного обеспечения имеет на этапе проектирования. Величина выигрыша от рациональной компоновки модулей линейно зависит от частоты использования программного комплекса.

Следует отметить, что при мультипрограммном режиме работы ЭВМ соответствующая указанному выше выигрышу производительность центрального процессора частично используется для выполнения других программ.

Выводы к главе 4

1. Приведено построение подсистемы "АСУ-Ростгеология", назначение, структура информационной базы и состав программного комплекса.
2. Приведен пример решения задачи выбора структуры программного комплекса компоновки модулей для одного из вариантов состава программного обеспечения.
3. Выбранный вариант структуры программного комплекса позволяет значительно уменьшить число вызовов модулей из ВП при эксплуатации системы.
4. Предложенная методика выбора структуры позволяет проводить анализ функционирования программного обеспечения. Такой подход применяется на этапах проектирования и эксплуатации ряда систем, в том числе рассмотренной подсистемы "АСУ-Ростгеология".

Приложение I

Доказательство леммы 2.1.

Пусть s_0 оптимально, а $\bar{s}_0 = \Phi(s_0)$ - не оптимально. Тогда существует $\bar{s}_1 \in N(i, j)$, такое, что

$$f(\bar{s}_1) \leq f(\bar{s}_0). \quad (\text{П.1})$$

В этом случае по (2.3) имеем

$$f(s_0) = f(\bar{s}_0) + c, \quad f(s_1) = f(\bar{s}_1) + c, \quad (\text{П.2})$$

где $s_1 = \Phi^{-1}(\bar{s}_1)$.

Очевидно, что такой обратный переход возможен. Из (П.1) и (П.2) получаем $f(s_1) \leq f(s_0)$, что противоречит предположению об оптимальности s_0 . Доказательство в обратную сторону аналогично.

Доказательство леммы 2.2.

Пусть имеется произвольное допустимое решение s элементов множества R , в котором между работами x и y_0 находятся элементы некоторого множества $A \subset R$. Если $A = \emptyset$, то это расписание является искомым. Пусть $A \neq \emptyset$. Очевидно, что любой элемент множества A образует с элементом y_0 независимую пару элементов, т.к. рассматривается допустимое расписание и сеть Γ , задающая ограничение предшествования, является совокупностью деревьев, причем по условию леммы $y \in \Gamma x$. Следовательно, возможны два случая:

1) когда любой элемент множества A образует с элементом независимую пару элементов;

2) среди элементов множества A есть элементы, которые не могут стоять в допустимом расписании перед элементом x .

Рассмотрим первый случай. Найдем параметры работы J , эквивалентной работам множества A и перейдем к эквивалентному расписа-

нию $\bar{s}$, причем

$$f(s) - f(\bar{s}) = c. \quad (\text{П.3})$$

Подсчитаем приоритет $\omega(J)$ эквивалентной работы J .

Возможны два варианта:

а) Если $\omega(J) > \omega(\gamma_0)$, то поменяем эти работы местами, причем значение критерия не ухудшится.

б) Если $\omega(J) < \omega(\gamma_0)$, то по условию леммы $\omega(J) < \omega(x)$.

Тогда поменяем работы J и x местами, причем значение критерия не ухудшится.

В обоих вариантах допустимость не нарушалась. В результате имеем допустимое эквивалентное расписание $\bar{s}_1$, причем

$$f(s_1) \leq f(s). \quad (\text{П.4})$$

Заменяя эквивалентную работу J в расписании $\bar{s}_1$ на исходные работы множества A получим допустимое расписание s_1 работ множества R , в котором работы x и γ_0 находятся рядом. Причем т.к.

$$f(s) - f(\bar{s}_1) = c, \text{ с учетом (П.2) и (П.4) имеем: } f(s_1) \leq f(s).$$

Рассмотрим второй случай. Если какой-то элемент $z \in A$ не может стоять в допустимом расписании перед элементом x , то элемент z входит в поддерево с корнем x . Тогда, учитывая условие леммы и то, что все поддеревья поддерева с корнем x являются нормальными модулями имеем:

$$\omega(z) \geq \omega(\gamma_0). \quad (\text{П.5})$$

Обозначим множество всех таких элементов через B , $B \subseteq A$.

Рассмотрим элемент $z_0 \in B$ такой, что в s z_0 правее всех $z \in B \setminus z_0$.

В расписании s между элементами z_0 и γ_0 могут находиться работы некоторого множества $C \subseteq A$. Если $C = \emptyset$, то поменяем элементы z_0 и γ_0 местами и получим допустимое расписание без ухудшения значения критерия. Пусть $C \neq \emptyset$. Очевидно, что по отношению к элементам z_0 , γ_0 и множеству A имеет место ситуация, аналогичная только

что разобранному первому случаю, так как любой элемент множества S образует с z_0 независимую пару, т.е. можно без ухудшения критерия перейти к допустимому расписанию, в котором элементы z_0 и y_0 находятся рядом. Затем можно поменять эти элементы местами без ухудшения значения критерия.

Т.к. мощность множества B уменьшена на 1, а множество конечно, то продолжая процесс, описанный выше, за конечное число шагов приходим к допустимому расписанию, в котором работы x и y_0 находятся рядом и значение критерия не больше, чем $f(z)$.

Доказательство леммы 2.3.

Рассмотрим последовательный модуль (рис.1). Если $\omega(\tau_i) < \omega(\tau_{i+1})$, $1 \leq i < k$, то исходный модуль имеет нормальный вид. Пусть это не так, т.е. существует τ_i такой, что $\omega(\tau_i) \geq \omega(\tau_{i+1})$. Согласно определению 2.7 все элементы множества работ R , не входящие в рассматриваемый модуль, или предшествуют всем элементам модуля, или следуют за ними, или независимы с ними. Пусть имеется некоторое допустимое оптимальное расписание s работ множества R . В нем между элементами τ_i и τ_{i+1} находятся элементы некоторого множества $A \subseteq R \setminus \{\tau_i\}$. Очевидно, что элементы множества A образуют независимые пары с элементами τ_i и τ_{i+1} , т.к. расписание является допустимым.

Т.к. выполняется условие "склеивания", можно "склеить" работы множества A и перейти к эквивалентному расписанию $\bar{s}$, в котором вместо работ множества A рассматривается эквивалентная работа $J = J(A)$, причем $f(\bar{s}) = f(s) + c$. Возможны два случая:

1) $\omega(\tau_i) > \omega(J(A))$; 2) $\omega(\tau_i) \leq \omega(J(A))$.

В первом случае можно поменять местами работы τ_i и $J(A)$ и перейти к расписанию $\bar{s}_1$, причем $f(\bar{s}_1) \leq f(\bar{s})$. Если затем заменить в расписании $\bar{s}_1$ $J(A)$ на работы множества A , получим расписание

s_1 , причем $f(s_1) = f(s_1) + c$ и $f(s_1) \leq f(s)$.

Во втором случае по предположению имеем: $\omega(r_{i+1}) \leq \omega(J(A))$, т.е. можно аналогично, как и в первом случае, поменять местами работы $J(A)$ и r_{i+1} , заменить работу $J(A)$ на работу множества A и получить расписание s_2 , причем $f(s_2) \leq f(s)$.

В обоих случаях при переходе от расписания s к s_1 или s_2 допустимость не нарушалась.

Очевидно, что если $A = \emptyset$, то сразу имеем расписание, в котором работы r_i и r_{i+1} находятся рядом.

Таким образом, от исходного допустимого расписания, не нарушая допустимости и не ухудшая значения критерия, всегда можно перейти к некоторому расписанию s_0 , в котором работы r_i и r_{i+1} находятся рядом. Т.е. оптимальное допустимое расписание работ множества R можно искать на множестве $N(r_i, r_{i+1})$. За счет "склеивания" работ r_i, r_{i+1} можно перейти к эквивалентному множеству расписаний $\bar{N}(r_i, r_{i+1})$, в котором вместо работ r_i, r_{i+1} рассматривается работа $J(r_i, r_{i+1})$, и, по лемме 2.1, искать оптимальное допустимое расписание на множестве расписаний $\bar{N}(r_i, r_{i+1})$. Т.к. число элементов множества $\{r_i\} - k$, конечно, то за конечное число проверок ($\omega(r_i) \geq \omega(r_{i+1})$), и если требуется "склеивание" можно перейти к эквивалентной задаче, в которой цепочка $\{r_i\}$ имеет нормальный вид. Если проводить процесс нормализации цепочки $\{r_i\}$, начиная с $i = 1$ (или с $i = k - 1$), то требуется не более $k - 1$ "склеивание" и $2(k - 1)$ сравнение приоритетов, т.е. $O(k)$ операций.

Доказательство леммы 2.4.

Рассмотрим параллельный модуль (рис.2). По лемме 2.3 за $O(k) + O(\ell) = O(m)$ действий последовательные модули, составляющие параллельный модуль, можно привести к нормальному виду. Таким

образом, можно считать

$$\omega(\tau_i) < \omega(\tau_{i+1}), \quad 1 \leq i < k; \quad (\text{II.6})$$

$$\omega(q_i) < \omega(q_{i+1}), \quad 1 \leq i < l. \quad (\text{II.7})$$

Пусть имеется произвольное допустимое расписание s работ множества R . Если для любых $x, y \in \{\tau_i\} \cup \{q_j\}$ таких, что работа x будет в s левее y , выполняется $\omega(x) < \omega(y)$, то модуль имеет нормальный вид. Пусть это не так, т.е. существуют $\tau_\lambda \in \{\tau_i\}, q_\gamma \in \{q_j\}$ такие, что τ_λ будет в s правее q_γ и

$$\omega(\tau_\lambda) < \omega(q_\gamma). \quad (\text{II.8})$$

Покажем, что в этом случае, не ухудшая значения критерия, можно перейти к допустимому расписанию, в котором работа τ_λ предшествует работе q_γ . Пусть в расписании s между работами τ_λ и q_γ находятся работы множества A . Выделим пять возможных случаев:

- 1) $A = \emptyset$;
- 2) $A \neq \emptyset, A \cap \{\tau_i\} = \emptyset, A \cap \{q_j\} = \emptyset$;
- 3) $A \cap \{\tau_i\} = \emptyset, A \cap \{q_j\} \neq \emptyset$;
- 4) $A \cap \{\tau_i\} \neq \emptyset, A \cap \{q_j\} = \emptyset$;
- 5) $A \cap \{\tau_i\} \neq \emptyset, A \cap \{q_j\} \neq \emptyset$.

Случай 1 и 2 очевидны (см. доказательство леммы 2.3). В случае 3 множество A включает в себя работы множества $Q = \{q_{\gamma+1}, q_{\gamma+2}, \dots, q_{\gamma+m}\}$. В силу (II.7) и (II.8) имеем $\omega(\tau_\lambda) < \omega(z)$ для любого $z \in Q$.

Рассмотрим последовательно пары элементов τ_λ и $q_{\gamma+i}$, где i изменяется от m до 0. Для каждой такой пары элементов имеет место один из уже рассмотренных случаев. Таким образом, учитывая, что множество A конечно, за конечное число шагов, не ухудшая значения критерия, можно перейти к допустимому расписанию, где работа τ_λ предшествует работе q_γ . Случай 4 симметричен случаю 3.

Случай 5 сводится к одному из рассмотренных. Пусть $A \cap \{r_i\} = \{x_1, x_2, \dots, x_m\}$, при этом x_n в s будет левее x_{n+1} , $1 \leq n \leq m$. Согласно (П.7), $\omega(x_n) < \omega(x_{n+1})$, а согласно (П.8), $\omega(x_n) < \omega(a_n)$, $1 \leq n \leq m$. Обозначим элемент r_n через x_{m+n} . Рассмотрим последовательно элементы x_n , где n изменяется от 0 до $m + 1$. Для работ a_n и x_n имеет место один из первых трех случаев. Т.к. множество A — конечно, то за конечное число шагов, не ухудшая значения критерия, можно перейти к допустимому расписанию, в котором работа r_n предшествует работе a_n .

Таким образом, оптимальное расписание работ множества R можно искать на множестве расписаний, где элементы множеств $\{r_i\}$ и $\{a_j\}$ упорядочены по неубыванию приоритетов, т.е. параллельный модуль можно заменить последовательным, в котором работы упорядочены по неубыванию приоритетов. Слияние двух упорядоченных последовательностей чисел с количеством k и l требует $O(k+l)$ действий [19]. Поскольку могут существовать x, y такие что $x \in \{r_i\}$, $y \in \{a_j\}$ и $\omega(x) = \omega(y)$, получившийся модуль следует привести к нормальному виду (по лемме 2.3 это требует $O(m)$ действий).

Доказательство леммы 3.1.

Пусть X^* — оптимальное решение задачи (3.24). Существует такое ℓ_0 , что для X^* выполняется соотношение $V \delta_0 (\ell_0 - 1) \leq V(a_0) + \sum_{i=1}^m \sum_{j=1}^{q_i} x_{ij}^2 v^-(a_{ij}) \leq V \delta_0 \ell_0$. Среди наборов $\{X_\ell\}$ имеется решение X_{ℓ_0} , для которого $V^- = V \delta_0 \ell_0$, $V^+ = V \delta_0 (\ell_0 - 1)$. Для решения X^* можно рассмотреть задачу (3.23) с $V^- = V(a_0) + \sum_{i=1}^m \sum_{j=1}^{q_i} x_{ij}^2 v^-(a_{ij})$, $x_{ij} = (x_{ij}^1, x_{ij}^2) \in X^*$, $V^+ = V^-$ — задачу I.

В качестве задачи 2 рассмотрим задачу (3.23), соответствующую X_{ℓ_0} . Очевидно, что X^* — является оптимальным решением задачи I. Пусть Y^* — оптимальное решение задачи 2. Т.к. в задаче 2 значение

ограничении на ресурс не меньше, чем в задаче I, то

$$W(Y^*) \geq W(X^*). \quad (\Pi.9)$$

Поскольку X_{e_0} получено при решении задачи 2 алгоритмом 3.9, то $W(Y^*) \geq W(X_{e_0})$ и $W(X_{e_0}) \geq W(Y^*)(1-\varepsilon)$, т.е. с учетом (II.9) $W(X_{e_0}) \geq W(X^*)(1-\varepsilon)$. Т.к. для решения $\bar{X}$ верно по условию леммы 3.1 соотношение $W(\bar{X}) \geq W(X_{e_0}) \forall \varepsilon$, то $\alpha(\bar{X}) \leq \varepsilon$. Очевидно, что $\beta(\bar{X}) \leq \delta$.

Доказательство леммы 3.2.

Во всех постановках задач в алгоритме 3.II правая часть условий, связанных с ресурсом, не больше величины $V(1+2\delta_0\eta(a_0))$, т.е. $\eta(a_0) \geq \eta(a) \quad \forall a \in A$.

Доказательство леммы 3.3.

Рассмотрим набор компонент оптимального решения $x^*(a)$. Этому набору соответствуют величины $W(x^*(a))$ и $\varepsilon_1^*, \varepsilon_2^*$, такие, что $V_{\varepsilon_1^*-1} \leq v(a, x^*(a)) \leq V_{\varepsilon_1^*}$, $V_{\varepsilon_2^*-1} \leq \hat{v}(a, x^*(a)) \leq V_{\varepsilon_2^*}$. Среди полученных вариантов имеется некоторый $\tilde{x}(a)$, являющийся решением задачи (3.I2) при $V^- = V_{\varepsilon_1^*}$, $V^+ = V_{\varepsilon_2^*}$. Пусть $\bar{x}(a)$ - оптимальное решение задачи (3.I2) при аналогичных ограничениях. Тогда $W(\bar{x}(a)) - W(\tilde{x}(a)) \leq \varepsilon_0 W(\bar{x}(a))$, $W(x^*(a)) \leq W(\bar{x}(a))$. Следовательно, или $W(\tilde{x}(a)) \geq W(x^*(a))$, или $W(x^*(a)) - W(\tilde{x}(a)) \leq \varepsilon_0 W(\tilde{x}(a))$, т.к. $W(\tilde{x}(a)) \geq (1-\varepsilon_0)W(\bar{x}(a)) \geq (1-\varepsilon_0)W(x^*)$.

Доказательство леммы 3.4.

Пусть $\mu = 2$, $\Gamma a = \{\theta_1, \dots, \theta_m\}$. Рассмотрим набор компонент оптимального решения $x^*(a) = \{x_1^*, \dots, x_m^*, x^*(\theta_1), \dots, x^*(\theta_m)\}$. Ему соответствуют величины $W(x^*(a))$ и $\varepsilon_1^*, \varepsilon_2^*$, такие, что $V_{\varepsilon_1^*-1} \leq v(a, x^*(a)) \leq V_{\varepsilon_1^*}$, $V_{\varepsilon_2^*-1} \leq \hat{v}(a, x^*(a)) \leq V_{\varepsilon_2^*}$. Рассмотрим набор $x(a) = \{x_1^*, \dots, x_m^*, \tilde{x}(\theta_1), \dots, \tilde{x}(\theta_m)\}$, где наборы компонент $\tilde{x}(\theta_i)$ при $i = \overline{1, m}$ соответствуют наборам из леммы 3.3.

Набору $x(a)$ соответствуют такие e_1, e_2 , что $v(a, x(a)) \leq v_{e_1}$, $\hat{v}(a, x(a)) \leq \bar{v}_{e_2}$. Индексам e_1, e_2 соответствуют некоторые варианты решения задачи (3.23) — $\tilde{x}(a)$, т.е. $v(a, \tilde{x}(a)) \leq v_{e_1}$, $\hat{v}(a, \tilde{x}(a)) \leq \bar{v}_{e_2}$. Пусть $\tilde{x}(a)$ — оптимальное решение задачи (3.23) при e_1, e_2 . Тогда $W(\tilde{x}(a)) \geq (1 - \varepsilon_0) W(x^*(a))$. Пусть $W(\tilde{x}(a)) > W(x^*(a))$. Тогда $W(\tilde{x}(a)) \geq (1 - \varepsilon_0) W(x^*(a))$ и $d(\tilde{x}(a)) \leq \varepsilon_0$. Рассмотрим случаи когда $W(\tilde{x}(a)) < W(x^*(a))$. Очевидно, что $W(x(a)) \leq W(\tilde{x}(a))$. Вследствие леммы 3.3 $|W(x^*(a)) - W(x(a))| \leq \varepsilon_0 m(a) W(x^*(a))$. Если $W(\tilde{x}(a)) > W(x(a))$, то $d(\tilde{x}(a)) \leq \varepsilon_0 m(a)$. Если $W(\tilde{x}(a)) < W(x(a))$ то т.к. $W(x(a)) - W(\tilde{x}(a)) \leq \varepsilon_0 W(\tilde{x}(a))$, имеем $W(x(a)) - W(\tilde{x}(a)) \leq \varepsilon_0 W(\tilde{x}(a))$ и $W(x^*(a)) - W(\tilde{x}(a)) \leq \varepsilon_0 m(a) W(x^*(a)) + \varepsilon_0 W(\tilde{x}(a)) \leq \varepsilon_0 (m(a) + 1) W(x^*(a))$, т.е. $d(\tilde{x}(a)) \leq \varepsilon_0 \eta(a)$. Для $M = 3, k = 1$ доказательство проводится аналогично.

Приложение 2

В статье [25] М.Ш.Левин непосредственно участвовал в разработке структуры информационной базы и программного обеспечения, метода реализации двухэтапного поиска. Ему принадлежат основные идеи организации информационной базы и программного комплекса, оптимизационные модели размещения данных и программ системы в памяти ЭВМ. В статье [26] М.Ш.Левин принадлежат основные идеи постановки задачи размещения данных и подхода к решению.

В статье [27] М.Ш.Левин непосредственно участвовал в формулировке условия склеивания работ, леммы о свойствах эквивалентных расписании, теоремы о приведении задачи к нормальному виду, алгоритма решения задачи. Ему принадлежат доказательства лемм, основные идеи доказательства теоремы и пример.

Министерство геологии РСФСР
ЦЕНТРАЛЬНЫЙ
ГЕОФИЗИЧЕСКИЙ ТРЕСТ

Справка о внедрении результатов диссертационной работы
Левина Марка Ивановича

от 11.03.80 10.67
на № _____ от _____

В Опытнометодической экспедиции АСУ Центрального геофизического треста Министерства геологии РСФСР проводятся работы по созданию ряда подсистем отраслевой автоматизированной системы управления АСУ-Росгеология.

Сотрудник ОМЭ АСУ М.И. Левин возглавлял и непосредственно участвовал в работах по проектированию программного комплекса для ЕС ЭВМ в рамках тем: "Разработка и внедрение I-ой очереди АСУ "АСУ-Росгеология" по подсистеме "Геология" на уровне Центрального аппарата Министерства геологии РСФСР "ИПС-Геология", № гос. регистрации I-75-81/4; "создание комплекса подсистем "Управление и экономика" "АСУ-Росгеология" и "АСУ-ЦТ", № гос. регистрации I-79-145/2; "Разработка и внедрение экономико-математических моделей оптимального планирования геологоразведочных работ на нефть и газ для уровня центрального аппарата ИТ РСФСР", № гос. регистрации I-78-42/1.

В процессе создания программного обеспечения тов. Левиным М.И. были предложены оригинальные оптимизационные модели организации данных, отраженные в его публикациях, вошедших в диссертационную работу. Модели были применены при разработке структуры информационной базы и программного комплекса, что обеспечило экономное использование памяти и производительности ЭВМ. В настоящее время модели, предложенные тов. Левиным М.И., используются при проектировании программного обеспечения по теме "Разработка и внедрение автоматизированной системы многовариантных оптимизационных плановых расчетов для задач нефтегазовой геологии на уровне Министерства геологии РСФСР", № гос. регистрации I-80-62/4.

Управляющий треста
Гербовая печать

К.В. Кавалеров

КОМИТЕТ ПО ФИЗИЧЕСКОЙ КУЛЬТУРЕ И СПОРТУ
ПРИ СОВЕТЕ МИНИСТРОВ СССР

ГЛАВСПОРТПРО

ВСЕСОЮЗНЫЙ ПРОЕКТНО-ТЕХНОЛОГИЧЕСКИЙ
И ЭКСПЕРИМЕНТАЛЬНО-КОНСТРУКТОРСКИЙ ИНСТИТУТ
ПО СПОРТИВНЫМ И ТУРИСТСКИМ ИЗДЕЛИЯМ
ВИСТИ

гор. Москва 127474, Дзигитровское шоссе 86. Тел. 487-32-65 секретарь
Расч. счет 608330 в Тиняряевском отд. Госбанка 487-32-54 бухгалтерия

№ _____ " " 19 г

Председатель Специализированного
Совета ВНИИСи

Справка о внедрении

тов. Емельянову С.В.

В ВИСТИ проводятся работы по созданию целевых комплексных программ совершенствования спортивных изделий. При расчете экономических показателей программ используются методы математического программирования.

При реализации оптимизационных расчетов на ЭВМ возникли вопросы эффективного использования ограниченного объема оперативной памяти. В процессе организации оверлейных структур программных комплексов используются результаты исследований аспиранта Московского государственного университета им. М.В. Ломоносова Левина Г.И., отраженные в публикациях, вошедших в его диссертационную работу, в том числе: модели программного комплекса в рамках оверлейной древовидной структуры и методы решения соответствующих дискретных оптимизационных задач компоновки модулей, направленных на уменьшение среднего числа обращений к внешним устройствам при ограничении на объем оперативной памяти, предоставляемой для работы комплекса. Применение указанных результатов позволяет осуществлять обоснованный выбор вариантов оверлейных структур программных комплексов.

В настоящее время на основе использования указанного программного обеспечения проводятся расчеты, являющиеся методической основой разработки комплексных тем создания изделий для различных видов спорта.

Главный инженер ВИСТИ
канд. техн. наук

В.Г. Цветков

Гербовая печать

- 114 -

С С С Р
МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ
Московский ордена Ленина, ордена Октябрьской революции
и ордена Трудового Красного Знамени Государственный
Университет имени М.В.Ломоносова

ЭКОНОМИЧЕСКИЙ ФАКУЛЬТЕТ

оси. 117234. Ленинские гор. П/учебн. корпус. Тел. 139-31-76
д. 30.9.23 /507 " 5 " июля 1981г.

Справка

о внедрении результатов диссертационной
работы Левина Марка Ивановича "Некото-
рые дискретные оптимизационные задачи
распределения ресурсов в вычислительных
системах"

Изложенный в диссертационной работе тов. М.И.Левина подход к
использованию вычислительной техники, а также оригинальные эффектив-
ные точные методы решения одно- и двухпроцессорных задач теории рас-
писания при ограничениях предшествования в виде совокупности деревь-
ев или параллельно-последовательных сетей и эффективные приближенные
алгоритмы решения экстремальной задачи на взвешенном дереве обоб-
щение задачи о рюкзаке и задачи о лестничном рюкзаке включены в
учебный курс "Исследование операций" и используются на семинарских
занятиях со студентами и аспирантами.

Зам. Декана Экономического факультета

ИТУ им. М.В.Ломоносова

доктор экономических наук,

профессор

В.Н.Черковец

Гербовая печать.

Заключение

Диссертация посвящена исследованию проблем построения дискретных моделей распределения ресурсов в вычислительных системах и разработке эффективных методов решения возникающих оптимизационных задач. Рассматриваются задачи организации вычислительного процесса и организации данных.

Вопросы, решенные в работе, вызваны потребностями практики и окажутся полезными разработчикам реальных вычислительных систем. Некоторые результаты могут иметь другие приложения, например, при календарном планировании производства.

Основные научные результаты работы состоят в следующем:

1. Исследован класс детерминированных задач составления расписания при организации вычислительного процесса в одно- или двухпроцессорном вычислительном комплексе в случае одновременного поступления заявок и ограничения предшествования в виде дерева или параллельно-последовательной сети. В качестве целевых функционалов в рассмотренных задачах выбраны: общее время выполнения заданий; сумма (или среднее значение) моментов окончания выполнения заданий; взвешенная сумма (или средневзвешенное значение) моментов окончания выполнения заданий, взвешенная сумма (или средневзвешенное значение) величины, экспоненциально зависящих от моментов окончания выполнения работ.

На основе свойств задач этого класса разработан общий эффективный метод, позволяющий получать оптимальное расписание.

2. Проведено сравнение характеристик класса детерминированных задач составления расписания при организации вычислительного процесса в одно- и многопроцессорных вычислительных комплексах.

3. Предложена формализация экстремальной задачи организации данных на примере компоновки программных модулей, входящих в некоторую систему. Дискретная оптимизационная постановка заключается в мини-

мизации среднего числа обменов между оперативной и внешней памятью при ограничении на общий объем оперативной памяти, предоставляемой для работы комплекса, при предположении, что передача по управлению осуществляется с возвратом в вызывающий модуль и связи между модулями имеют вид дерева.

4. Задача компоновки сведена к задаче выбора стягиваемых дуг во взвешенном ориентированном дереве.

5. Выделены различные частные случаи поставленной задачи (в зависимости от структуры связей между модулями и значений параметров модулей) и предложены эффективные методы решения: точные алгоритмы, приближенные алгоритмы с гарантированной относительной погрешностью целевой функции; приближенные алгоритмы с гарантированной относительной погрешностью целевой функции и гарантированной относительной погрешностью выполнения ограничения на память.

6. Для случая, когда связи между модулями имеют вид ациклического направленного графа, предложен эвристический подход, который позволяет на основе пренебрежения слабыми связями между модулями и первоочередного учета сильных связей или получать некоторое решение задачи, или свести ее к одному из рассмотренных древовидных случаев.

7. Рассмотрена реализация предложенного подхода при выборе варианта структуры программного комплекса АСУ-Ростгеология (компоновка модулей).

8. Предложенная методика решения задачи компоновки программных модулей позволяет проводить анализ функционирования программного обеспечения АСУ. Такой подход применялся на этапах проектирования и эксплуатации ряда систем, в том числе подсистемы "АСУ-Ростгеология".

Литература

1. Материалы XXVI съезда КПСС.-М.: Политиздат, 1981.-223 с.
2. Авен О.И., Коган Я.А. Управление вычислительным процессом в ЭВМ.-М.: Энергия, 1978.-240 с.
3. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. с англ. /Под ред. Ю.В. Матиясевича.-М.: Мир, 1979.-535 с.
4. Основы создания больших АСУ /В.А. Баранюк, Е.С. Бичуров, А.И. Черкыщенко, Ш.У. Уразельдиев; Под ред. В.А. Баранюка.-М.: Сов. радио, 1979.- 360 с.
5. Белоногов Г.Г., Новоселов А.П. Автоматизация процессов накопления, поиска, и обобщения информации. - М.: Наука, 1979,-256 с.
6. Бронштейн И.И. Выбор стратегии обмена информацией между двумя уровнями памяти. - Методы реализации алгоритмов контроля и управления АСУ.-М.: Изд. ИГУ АН СССР, 1974, вып.3, 34 с.
7. Бронштейн И.И. Об оптимальном разбиении оперативной памяти по группам программ. - Автоматика и телемеханика, 1971, № 10, с.133-141.
8. Бурков В.Н., Рубинштейн М.И., Соколов В.Б. Некоторые задачи оптимального размещения информации в памяти большого объема.- Автоматика и телемеханика, 1969, № 9, с.83-91.
9. Пенс Г.В., Левнер Е.В. Об эффективных ϵ -алгоритмах для некоторых универсальных задач теории расписаний.-Изв. АН СССР, Техническая кибернетика, 1978, № 6, с.38-43.
10. Имади Э.Х., Глебов Н.И., Перемелица В.А. Алгоритмы с оценкой для задач дискретной оптимизации.-Проблемы кибернетики.-М.: Наука, 1976, вып.31, с.36-42.
11. Глушков В.М. и др. Обработка информационных массивов в автоматизированных системах управления.-Киев: Наука, 1970.- 181 с.

12. Гольдтабер Е.М. Задача минимизации времени исполнения проекта работ, заданного деревом.- Кибернетика, 1977, № 2, с. 102-107.
13. Гордон В.С., Танаев В.С. Детерминированные системы обслуживания с одним прибором, древовидным упорядочением требований и экспоненциальными функциями штрафа.- В кн. Вычислительная техника в машиностроении.- Минск: изд-во ин-та технического кибернетики АН БССР, 1973, с. 3-10.
14. Турин Н.Н. Оптимальное размещение блоков информации на магнитном диске с подвижными головами.- Автоматика и телемеханика, 1978, № 12, с. 143-151.
15. Турин Н.Н., Дашко В.Е., Коган Я.А., Леман А.А. Исследования взаимодействия операционной системы с запоминающими устройствами на магнитных дисках.- Автоматика и телемеханика, 1974, № 2, с. 152-162.
16. Джонсон С. Оптимальное расписание для двух- и трехступенчатых процессов. Кибернетический сборник, № 1, Новая серия.- М.: Мир, 1965, с. 78-86.
17. Дискретная математика и математические вопросы кибернетики. /Под ред. С.В. Яблонского и О.Б. Луанова.- М.: Наука, 1974, - 312 с.
18. Кнут Д. Искусство программирования для ЭВМ, т.3, сортировка и поиск: Пер. с англ. /Под ред. Ю.М. Баяковского, В.С. Штаркмана.- М., 1978, - 844 с.
19. Коган Я.А., Файнштейн И.А., Шейнман М.В. Исследование и оптимизация систем программного обеспечения.- Автоматика и вычислительная техника, 1976, № 2, с. 35-63.
20. Колпанова Т.В., Орлов С.Г., Филиппович В.В. Исследование характеристик иерархической структуры программ АСУ. Управляющие системы и машины, 1976, № 5, с. 42-52.

21. Конвей Р.В., Максвелл В.Л., Миллер Л.В. Теория расписаний: Пер. с англ. /Под ред. Г.П.Башарина.-М., Наука, 1975.-360 с.
22. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование.- М.: Наука, 1969.-368 с.
23. Лапшин Ю., Коровкин В. Экономическая эффективность использования вычислительной техники.-Плановое хозяйство, 1976, №4, с.101-107.
24. Левин М.Ш. Об эффективном решении некоторых задач теории расписаний на сетях.-Кибернетика, 1980, № 1, с.123-127.
25. Левин М.Ш., Гильберштейн А.П. Информационно-поисковая система "ИИС-Геология". - В кн.: Разработка и создание АСУ-Геология,-М.: ВИАМС, 1979, вып.4 18, с.41-54.
26. Левин М.Ш., Молочков Л.И. Задача размещения данных на магнитных дисках. - В кн.: Сборник трудов Науч.-исслед. и проектного ин-та систем управления Госнаб СССР "Проблемы типизации проектных решений АСУ МТС".-Тула: 1979, вып.7, с.60-65.
27. Левин М.Ш., Левнер Е.В. Об эффективном решении задачи Беллмана-Джонсона на сети, имеющей вид дерева.- Автоматика и телемеханика, 1978, № 10, с.110-118.
28. Левнер Е.В. Теория расписаний в экономических системах, некоторые математические вопросы. Препринт.-М.: ЦЭМИ АН СССР, 1977;53 с.
29. Левнер Е.В., Генс Г.В. Дискретные оптимизационные задачи и эффективные приближенные алгоритмы. Препринт.-М.: ЦЭМИ АН СССР, 1978.-55 с.
30. Лившиц Э.М. Последовательность операции при изготовлении сложной детали.-Автоматика и телемеханика, 1968, № 11, с.94-95.
31. Липаев В.В. Распределение ресурсов в вычислительных системах.- М.: Статистика, 1979. - 247 с.
32. Липаев В.В., Филиппович В.В. Принципы и правила модульного построения сложных комплексов АСУ.- Управляющие системы и машины, 1975, № 1, с.15-22.

33. Липаев В.В., Яшков С.Ф. Эффективность методов организации вычислительного процесса в АСУ.-М.: Статистика, 1975.-255 с.
34. Литвинов В.А. Некоторые методы обработки файлов в памяти прямого доступа и оценка их эффективности.- Управляющие системы и машины, 1978, № 5, с. 12-19.
35. Мамиконов А.Г., Пискунов А.Н., Цвиркун А.Д. Модели и методы проектирования информационного обеспечения АСУ.-М.: Статистика, 1978.-221 с.
36. Моисеенко Г.Е. Оптимальное разбиение системы на подсистемы.- Автоматика и телемеханика, 1979, № 7, с.103-112.
37. Немировский А.С., Юдин Б.Д. Сложность задач и эффективность методов оптимизации.-М.: Наука, 1979.-384 с.
38. Погребной Е.П. Эффективность АСУ.-М.: Наука, 1979.-103 с.
39. Сигалов Г.Г., Курьянова Н.И. Минимизация среднего времени доступа к накопителю информации на магнитном диске.-Управляющие системы и машины, 1976, № 6, с.116-122.
40. Современное состояние теории исследования операций.-М.: Наука 1979.-464 с.
41. Сэлтон Г. Автоматизированная обработка, хранение и поиск информации: Пер. с англ. /Под ред. А.И.Китова.- Советское радио, 1973.-560 с.
42. Танаев В.С., Шкурба В.В. Введение в теорию расписаний.- М.: Наука, 1975.-256 с.
43. Тарьян Р.Э. Сложность комбинаторных алгоритмов. Кибернетический сборник № 17, Новая серия.-М.:Мир, 1980,-с.61-113.
44. Трифонов Ю.В. Размещение файлов в памяти прямого доступа. Автоматика и вычислительная техника, 1976, № 3, с.63-67.
45. Управление исследованиями, разработками и внедрением новой техники.-М.: Экономика, 1978.-287 с.

46. Управление социалистическим производством. Вопросы теории и практики.-М.: Экономика, 1978.-765 с.
47. Финкельштейн Ю.Ю. Приближенные методы и прикладные задачи дискретного программирования.-М.: Наука, 1976. - 264 с.
48. Фридман А.А. О некоторых современных направлениях в дискретной оптимизации.-Экономика и математические методы, 1977, т. XIII, вып. 5, с. 1115-1131.
49. Фуксман А.Л. Технологические аспекты создания программных систем.-М.: Статистика, 1979.-184 с.
50. Ху Т.С. Параллельное упорядочивание и проблемы линии сборки. Кибернетический сборник № 4, новая серия.-М.: Мир, 1967. с. 43-56.
51. Швецов В.И. Оптимизация многоступенчатой организации данных.- Изв. АН СССР, Техническая кибернетика, 1978, № 4, с. 24-28.
52. Швецов В.И. Экстремальные задачи обработки данных. - Изв. АН СССР. Техническая кибернетика, 1978, № 1, стр. 111-120.
53. Юдин Д.Б., Немировский А.С. Оценки информационной сложности задач математического программирования. - Экономика и математические методы, 1976, т. XII, вып. I, с. 123-142.
54. Яблонский С.В. Введение в дискретную математику.-М.: Наука, 1979.- 272 с.
55. Adolphson D.L. Single Machine Sequencing with Precedence Constraints.- SIAM J. Comput., 1977, 6, p. 40-54.
56. Adolphson D., Hu T.C. Optimal Linear Ordering. - SIAM J. Appl. Math., 1973, v 25, No 3, p. 403-423.
57. Brucker P., Lenstra J.K., Rinnooy Kan A.H.G. Complexity of Machine Scheduling Problems. - Mathematisch Centrum Afdeling Mathematische Besliskunde B.V., 1975, No 43.

58. Coffman E.G. Jr. (ed.) Computer and Job/Shop Scheduling Theory. - New York: J. Wiley and sons, 1971. - 299 p.
59. Coffman E.G., Graham R.L. Optimal Scheduling for Two-Processor System. - Acta Informatica, 1972, 1, p. 200-213.
60. Horn W.A. Single-Machine Job Sequencing with Tree-like Precedence Ordering and Linear Delay Penalties. - SIAM J. Appl. Math., 1972, v. 25, no. 2, p. 189-202.
61. Ibarra O.H., Kim C.E. Fast Approximation Algorithms for the Knapsack and Sum of Subsets Problems. - J. ACM, 1975, 22, p. 463-468.
62. Kondo H., Yoshida I. Effective File Allocation Methods onto Disc Devices. - Proceedings of the 8th IFIP Conference on Optimization Techniques, part 2, Berlin, Heidelberg, New York: Springer-Verlag, 1978, p. 473-484.
63. Kurisu T. Two-machine scheduling under required precedence among jobs. - J. of the Oper. Res. Society of Japan, 1976, vol. 17, p. 1-13.
64. Kurisu T. Two-machine scheduling under arbitrary precedence constraints. - J. of the Oper. Res. Society of Japan, 1977, vol. 20, No 2, p. 113-131.
65. Kurisu T. Three-machine scheduling problems with precedence constraints. - J. of the Oper. Res. Society of Japan, 1977, vol. 20, No 3, p. 231-242.
66. Lawler E.L. Optimal Sequencing of a Single Machine Subject to Precedence Constraints. - Management Sci., 1973, 19, p. 544-546.
67. Lawler E.L., Sivasubramanian B.D. Minimization of

Time-Varying Costs in Single-Machine Scheduling. -
Oper. Res., 1978, vol. 26, No 4, p. 563-569

68. Lenstra J.K., Rinnooy Kan A.H.G. Complexity of Scheduling under Precedence Constraints. - Oper. Res., 1978, v. 1, No 1, p. 22-35

69. Lenstra J.K. Sequencing By Enumerative Methods
Mathematisch Centrum. - Amsterdam: 1977. - 198p.

70. Monma C.L., Sidney J.B. A General Algorithms
for Optimal Job Sequencing with Series-Parallel
Precedence Constraints. - Tech. Report.
No 347, School of Operations Research and
Industrial Engineering, Cornell University,
July 1977.

71. Sahni S. Algorithms for Scheduling Independent
Tasks. - J. ACM, 1976, vol. 23, No 1, p. 114-127.

72. Smith W. E. Various Optimizers for Single
Stage Production. - Naval Res. Logist. Quart.,
1956, vol. 3, No 1, p. 59-66.

73. Symposium on the Theory of Scheduling and
its Applications / Edited by S. E. Elmaghraby, VIII,
Berlin, Heidelberg, New York: Springer-Verlag, 1973, vol. 86-473p.

74. Ullman J. D. NP-Complete Scheduling Problems. -
J. Comp. and Syst. Sci., 1975, vol. 10, p. 384-393.