В.В. СЕМЕНЕЦ, И.В. ГРЕБЕННИК,
С.В. ЛИСТРОВОЙ, С.В. МИНУХИН,
А.О. ОВЕЗГЕЛЬДЫЕВ

МОДЕЛИ И МЕТОДЫ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ В ПРОЕКТИРОВАНИИ И УПРАВЛЕНИИ

КИЕВ НАУКОВА ДУМКА 2019

УДК 519.85;004.75

В.В. Семенец, И.В. Гребенник, С.В. Листровой, С.В. Минухин, А.О. Овезгельдыев. Модели и методы комбинаторной оптимизации в проектировании и управлении. — Киев: Наукова думка, 2019. — 176 с.
ISBN 978-966-00-1684-2

Монография посвящена разработке и исследованию математических моделей и методов комбинаторной оптимизации, ориентированных на решение задач проектирования и управления, среди которых конструкторское проектирование микроэлектронных устройств, геометрическое проектирование и управление распределенными вычислениями. Приведены общая формальная постановка и постановки основных классов задач конструкторского проектирования микроэлектронных устройств, а также методы и алгоритмы их решения. Изложен единый подход к формализации описаний, генерации и перечислению классов комбинаторных конфигураций, обладающих заданными свойствами, для математического моделирования и решения задач, имеющих сложную комбинаторную структуру. В рамках единого подхода построена и проанализирована математическая модель многокритериальной задачи размещения параллелепипедов. Рассмотрены математические постановки и методы решения задач планирования в распределенных вычислительных системах (РВС), сводящиеся к задачам дискретной оптимизации с линейными и нелинейными функциями цели, для решения которых предложен общий подход. В рамках решения задач управления распределенными вычислениями предложен метод мониторинга состояния компонент РВС, использующий представление РВС в виде неполносвязного графа, на основе решения задач нахождения кратчайшего пути и кратчайшего гамильтонова цикла.

Для научных работников и специалистов, занимающихся оптимизацией широкого круга задач проектирования и управления системами, а также аспирантов соответствующих специальностей.

Научно-издательский отдел физико-математической
и технической литературы

Редактор *В.В. Вероцкая*

ISBN 978-966-00-1684-2

ВВЕДЕНИЕ

Развитие человечества на современном этапе все более определяется уровнем развития науки и технологии, возможностями эффективно решать новые сложные задачи. Современные научные подходы к решению проблем при решении задач проектирования и управления системами практически во всех областях базируются на достижениях в области математического моделирования и компьютерных технологий.

Важным направлением в проектировании систем является разработка и исследование оптимизационных алгоритмов решения задач синтеза конструкции электронных схем большой размерности с учетом дополнительных функциональных ограничений. В основном это относится к микроэлектронным устройствам (МЭУ) повышенной степени интеграции (многокристальные микросборки, гибридные интегральные схемы) — классу конструкций, обладающих рядом особенностей, существенно усложняющих процесс проектирования. Кроме повышенной интеграции к ним относятся разногабаритность модулей, высокая плотность печатного монтажа и, как следствие, необходимость учета паразитных связей в конструкции. Алгоритмы, предназначенные для решения указанных задач, могут быть эффективно применены и для решения задач синтеза систем иной природы, встречающихся в промышленности, строительстве и т. п. и допускающих соответствующую постановку. Созданию методов автоматизированного синтеза конструкций МЭУ посвящены работы Л.Б. Абрайтиса, Р.П. Базилевича, А.М. Бершадского, Б.Н. Деньдобренко, В.М. Курейчика, К.К. Морозова, А.И. Петренко, Г.Г. Рябова, В.А. Селютина, Ю.Г. Стояна, А.Я. Тетельбаума и др.

Дискретный характер основных задач конструкторского проектирования предполагает использование моделей и алгоритмов комбинаторной оптимизации. Комбинаторные задачи подобного рода отличаются большой размерностью, сложной геометрией

допустимой области, многоэкстремальностью, которая характеризуется, как правило, наличием большого числа ограничений. В этих условиях особое значение приобретают вопросы повышения глобальных свойств алгоритмов их решения, быстродействия, способности к адаптации. Наметившаяся тенденция к использованию методов искусственного интеллекта, элементов экспертных систем в системах автоматизации проектирования требует унификации проектных операций, использования единой схемы процесса проектирования, общей структуры моделей, алгоритмов и критериев. Это позволяет создавать абстрактные модели процесса проектирования, лучше согласовывать, а в некоторых случаях и объединять различные его этапы. Для схем большой размерности актуальна разработка быстродействующих приближенных методов расчета, обеспечивающих компромисс между точностью и временными затратами.

Методы математического моделирования реальных объектов и процессов играют важную роль при разработке систем, ориентированных на обработку геометрической информации. Задачи этого класса возникают в различных сферах человеческой деятельности, областях науки и техники. Системы базируются на математических моделях объектов реального мира, методах оптимизации и теории построения интеллектуальных систем.

Теория геометрического проектирования охватывает широкий круг фундаментальных и прикладных проблем, связанных с математическим моделированием процесса размещения реальных объектов и созданием эффективных методов оптимизации этого процесса в соответствии с выбранными критериями оптимальности.

Задачи геометрического проектирования состоят в определении оптимального размещения множества геометрических объектов относительно заданной области размещения с учетом технологических ограничений в соответствии с критериями качества размещения. К этому классу задач относятся задачи построения генеральных планов промышленных предприятий, компоновки оборудования, управления сложными техническими системами, задачи конструирования электронных устройств, задачи оптимального раскроя промышленных материалов, задачи теории расписаний и объемно-календарного планирования, задачи покрытия. Результаты исследований в этом направлении приведены в работах Л.В. Канторовича, В.Л. Рвачева, Ю.Г. Стояна, В.А. Залгаллера, С.В. Яковлева, Э.А. Мухачевой, Н.И. Гиля, Т.Е. Романовой, В.М. Комяк, J. Terno, G. Scheithauer, V. Milenkovich, J. Ferreira, J. Oliverra и других.

Многие задачи геометрического проектирования по своей структуре являются комбинаторными, а их математические модели сводятся к соответствующим классам задач комбинаторной оптимизации. Это объясняется дискретно-непрерывной природой самих задач и жесткой системой ограничений, накладываемых в ряде случаев на параметры размещения геометрических объектов. Исследованиям задач комбинаторной оптимизации и методов их решения посвящены работы отечественных и зарубежных авторов, среди них И.В. Сергиенко, Ю.Г. Стоян, С.В. Яковлев, C. Papadimitriou, P. Pardalos, В.П. Шило, М.З. Згуровский, Л.Ф. Гуляницкий, Н.В. Семенова.

Разработаны подходы к решению комбинаторных оптимизационных задач геометрического проектирования, основанные на методах ветвей и границ, отсечений, локальной оптимизации, эвристических методах, целочисленном, линейном и нелинейном программировании, полиэдральной комбинаторике, модификациях метода оптимизации по группам переменных. Этой тематике посвящены исследования Ю.Г. Стояна, С.В. Яковлева, О.А. Емца и их учеников. В результате их многолетней работы создана теория евклидовой комбинаторной оптимизации.

Для разработки эффективных методов решения комбинаторных оптимизационных задач геометрического проектирования необходим общий подход к математическому моделированию, построению и анализу адекватных математических моделей конкретных классов задач. Следовательно, актуальной является проблема создания общего подхода к математическому моделированию и решения комбинаторных оптимизационных задач геометрического проектирования.

Разработка математических моделей задач геометрического проектирования основана на аналитическом описании теоретико-множественных отношений включения (принадлежности объекта области размещения, в том числе расположение объекта на допустимом расстоянии до границы области) и непересечения (неналожение объектов, в том числе расположение на заданном допустимом расстоянии каждой пары размещаемых объектов). Общая задача построения адекватных математических моделей отношений геометрических объектов, обладающих произвольными пространственными формами в евклидовых двух- и трехмерных пространствах, является очень сложной. Она требует подробного и полного исследования отношений разнообразных пар геометрических объектов. При этом одной из наиболее акту-

альных задач является конструктивное представление геометрической информации, необходимой для адекватного моделирования реального процесса размещения. Исследования в этом направлении проводились в научной школе Ю.Г. Стояна.

При решении широкого круга задач, возникающих в системах распределенных вычислений и управлении такими системами, используются математические модели, отражающие формализацию построения структуры распределенной системы, формализацию процессов распределения потоков заданий, формализацию выбора подходящих ресурсов, оптимизацию расписания выполнения заданий на ресурсах и т. д. К наиболее распространенным относятся задачи об оптимальном разбиении, распределении (назначении) заданий между вычислительными ресурсами распределенной среды, решаемые на основе задачи о назначении, задачи управления ресурсами для минимизации времени выполнения, выбора эффективной в смысле некоторого критерия дисциплины обслуживания очередей заданий и др. Как правило, это класс *NP*-трудных задач, для решения которых требуется разработка полиномиально разрешимых алгоритмов, позволяющих осуществлять обработку заданий за время, приемлемое для эффективного управления вычислительными процессами.

Особенностью этих методов управления является то, что они относятся к задачам линейного и нелинейного булевого программирования, а это в общем случае не позволяет обеспечивать корректную сопоставимость вследствие достаточно большого количества различий при построении тех эвристик, которые при этом используются. Это связано с тем, что полиномиально разрешимые методы решения задач при практическом использовании трудно сопоставимы при больших размерностях решаемых задач и степени полиномиальности алгоритма, превышающей 4 и более. В связи с этим для данной проблематики перспективным является разработка общего подхода — рангового, предложенного и исследованного в работах С.В. Листрового. Он позволяет разрабатывать методы и алгоритмы, имеющие малую вычислительную сложность и низкую погрешность получаемых решений, для указанных задач в рамках единых принципов. Следовательно, разработка методов управления распределенными вычислениями в рамках единого подхода — актуальная задача, требующая рассмотрения и развития.

Данная монография посвящена разработке и исследованию подходов и методов решения указанных проблем.

МОДЕЛИ ОПТИМИЗАЦИИ В ЗАДАЧАХ КОНСТРУКТОРСКОГО ПРОЕКТИРОВАНИЯ

Проблемы конструкторского проектирования, к которым относятся задачи размещения, компоновки и трассировки при проектировании цепей во встроенных системах (СБИС) являются взаимосвязанными, их решению посвящено значительное количество публикаций [1—6].

Задача размещения заключается в таком расположении всех компонентов на плате, которое бы удовлетворяло ограничениям технологического процесса. После этого все соединения между выводами (контактами) компонентов должны быть маршрутизированы на плате (задача трассировки) в соответствии с логическим дизайном. Понятно, что решение задачи размещения оказывает существенное влияние на решение задачи трассировки. Поскольку условия на самом месте размещения обычно относительно просты (например, компонентам нельзя перекрываться), что приводит к необходимости определения минимальных расстояний, основная цель размещения заключается в обеспечении хорошей и выполнимой трассировки. В принципе, было бы лучше всего выполнить размещение и маршрутизацию одновременно. Однако на практике эти задачи обычно выполняются последовательно [7].

Следствием разделения этапа конструкторского проектирования на отдельные задачи явилось различие в формальной постановке этих задач. Это различие состоит, прежде всего, в выборе моделей. Например, сведение задачи размещения к задаче оптимизации на перестановках существенно ограничивает в выборе алгоритмов оптимизации, затрудняет формальное описание процесса получения начального решения. В задачах размещения смешанного набора модулей перестановочные модели становятся неадекватными. По этой же причине неэффективны попытки использования идей оптимизационных перестановочных алго-

ритмов размещения в задачах разбиения, расслоения, трассировки и т. д.

Вместе с тем, несмотря на содержательное отличие, задачи конструкторского проектирования допускают формулировку общей постановки.

Введение общей формальной постановки имеет существенные преимущества: возможность унификации проектных процедур, разработка общей формальной схемы алгоритма оптимизации с последующей конкретизацией ее для каждой из задач. Такая унификация особенно необходима при создании экспертных систем автоматизированного проектирования (САПР), внесении в САПР элементов искусственного интеллекта. Наличие общей постановки облегчает также использование алгоритмов или отдельных приемов, применяющихся при решении одних задач, для решения других задач. Так, идеи снятия и перетрассировки могут быть эффективно использованы в задачах разбиения и размещения.

В основу оптимизационных задач конструкторского проектирования могут быть положены классические модели комбинаторной оптимизации. Однако применение классических результатов в некоторых случаях не позволяет учесть специфику задач конструкторского проектирования.

В параграфе 1.1 изложены подходы к моделированию задач конструкторского проектирования на основе классических задач комбинаторной оптимизации. В последующих параграфах приведена общая постановка и базирующиеся на ней оптимизационные модели задач конструкторского проектирования, учитывающие специфику предметной области. Основные результаты данных исследований изложены в работах [8—19].

1.1. БАЗОВЫЕ ЗАДАЧИ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ В КОНСТРУКТОРСКОМ ПРОЕКТИРОВАНИИ

При решении задач конструкторского проектирования поиск физически допустимого места размещения, как правило, достаточно просто реализуем. Поэтому целью размещения является обеспечение качественной трассировки — минимизация общей длины проводки всех соединений [7].

Точное решение задачи определения длины каждой цепи чаще всего неизвестно, а эвристики работают в условиях перенаправления отдельных сегментов компонентов. Так как даже вы-

числение минимальной длины одной цепи (сети) в общем случае является *NP*-трудной задачей, сводимой к определению длины минимального дерева Штейнера, то необходимо оценить общую длину цепи. Обычно это определяется суммированием длины каждой отдельной цепи. Методы оценки длины конкретной цепи включают в себя: эвристическое вычисление дерева Штейнера между вводами, вычисление минимального остовного дерева между вводами, построение цепи через вводы или получение половины длины периметра ограничительной рамки, включающей все вводы цепи. При этом для цепей с двумя вводами и с использованием манхэттенской метрики — типичной метрики в печатной плате и конструкции СБИС — все эти задачи дают одинаковый результат. Другой целью задачи размещения является минимизация количества переходов. Согласно [7, 20] ее можно свести к задаче пересечения, которая формулируется следующим образом: пусть есть $G = (V, E)$ — двудольный граф с вершинами V_1 и V_2 и G вложено в плоскость, так что узлы в V_i занимают различные позиции на линии $y = i$, а ребра — прямые. Для вложения $f(G)$ в плоскости G пересечение $C_f(G)$ — это число пересечений прямых, определяемое f. Это число зависит только от перестановки V_i вдоль $y = i$, а не на конкретных x-координатах. Число пересечений двудольного графа $C(G)$ определяется как $C(G) = \min_f C_f(G)$. Вычисление $C(G)$ — *NP*-трудная задача [21]. Некоторые эвристики для минимизации пересечений приведены в [20], где также выполнено их сравнение.

Квадратичное назначение. Задача квадратичного назначения в конструкторском проектировании состоит в следующем: учитывая некоторые компоненты и количество позиций на поверхности платы, требуется назначить каждому компоненту такую позицию на плате, чтобы выполнялись определенные ограничения, и при этом длина общего соединения была минимальной.

Такая постановка позволяет построить следующую математическую модель [7]:

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} c_{ij} d(i,k,j,l) x_{ik} x_{jl} + \tag{1.1}$$

$$+ \lambda_0 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} o(i,k,j,l) x_{ik} x_{jl} \tag{1.2}$$

при условии $\sum_{k \in Z(i)} x_{ik} = 1$ для всех $i = 1,...n$, (1.3)

$$x_{ik} \in \{0,1\} \text{ для всех } i = 1,...n, k \in Z(i), \quad (1.4)$$

где $c_{ij} \geq 0$ — коэффициенты сходства между компонентами i и j, $c_{ij} = \sum_{t \in T} \frac{1}{\alpha_t - 1}$, $d(i,k,j,l)$ определяет кратчайшее манхэттенское расстояние между компонентами i и j. Второе слагаемое (1.2) целевой функции оценивает количество перекрытий. Коэффициенты $o(i,k,j,l)$, определенные для $i \in \{1,...,n\}$, $j \in \{1,...,n\}$, $k \in Z(i)$ и $l \in Z(j)$, подсчитывают количество ячеек сетки, которые разделяются компонентами i и j, когда они назначаются ячейкам сетки k и l. Поскольку допустимое размещение не должно содержать перекрытий, второй член целевой функции масштабируется с большим штрафным коэффициентом λ_0. Уравнения (1.3) гарантируют, что каждому компоненту присваивается ровно одна ячейка сетки.

Эта квадратичная оптимизационная формулировка допускает следующее расширение: когда некоторые компоненты разрешено поворачивать на 90°, существуют четыре разных реализации для размещения каждого из этих компонентов. В этом случае двоичные переменные x_{ik}^a определены для компонента i в ячейке сетки $k \in Z(i)$ и для реализации $a \in A(i)$, так что x_{ik}^a равно единице тогда и только тогда, когда назначается компонент i к допустимой ячейке сетки k в реализации a.

Задача расширенного квадратичного размещения формулируется следующим образом [7]:

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} \sum_{a_1 \in A(i)} \sum_{a_2 \in A(j)} c_{ij} d(i,k,a_1,j,l,a_2) x_{ik}^{a_1} x_{jl}^{a_2} +$$

$$+ \lambda_0 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k \in Z(i)} \sum_{l \in Z(j)} \sum_{a_1 \in A(i)} \sum_{a_2 \in A(i)} o'(i,k,a_1,j,l,a_2) x_{ik}^{a_1} x_{jl}^{a_2}$$

при условии $\sum_{k \in Z(i)} \sum_{a \in A(i)} x_{ik}^a = 1$ для всех $i = 1,...n$, $x_{ik}^a \in \{0,1\}$ для всех $i = 1,...n, k \in Z(i), a \in A(i)$, где $d(i,k,a_1,j,l,a_2)$ обозначает кратчайшее манхэттенское расстояние между компонентами *i* и *j*,

когда они помещаются на ячейки сетки k и l в реализации a_1 и a_2 и $o'(i,k,a_1,j,l,a_2)$ подсчитывает количество перекрывающихся компонентов i и j при размещении на ячейках сетки k и l в реализации a_1 и a_2.

Квадратичная модель размещения компонентов, описанная выше, относится к классу *NP*-трудных задач.

Приведенные математические модели относятся к известным классам задач комбинаторной оптимизации, методы решения которых достаточно исследованы. Однако они не учитывают всей специфики задач конструкторского проектирования микроэлектронных устройств. Изложению общего подхода к математическому моделированию и решению комбинаторных оптимизационных задач конструкторского проектирования микроэлектронных устройств посвящены следующие параграфы.

1.2. ОБЩАЯ ПОСТАНОВКА ЗАДАЧ КОНСТРУКТОРСКОГО ПРОЕКТИРОВАНИЯ МИКРОЭЛЕКТРОННЫХ УСТРОЙСТВ

Общая формальная постановка для задач конструкторского проектирования может быть представлена в следующем виде. Пусть даны m множеств объектов произвольной природы A_1, $A_2,\ldots,A_m$, возможно, с непустым пересечением, $A_i = \left\{a_1^i, a_2^i, \ldots, a_{n_i}^i\right\}$, $n_i = Card\, A_i$, $i = \overline{1,m}$, $\sum_{i=1}^{m} n_i \geq Card \bigcup_{i=1}^{m} A_i$. Необходимо, с учетом ограничений, выбрать по одному элементу из каждого множества таким образом, чтобы функционал, оценивающий качество совместного выбора элементов, принял минимальное значение:

$$a^* = \arg \min_{a \in K \subseteq A} \Phi(a), \tag{1.5}$$

$$A = A_1 \times A_2 \times \ldots \times A_m, \tag{1.6}$$

$$K : a \in K \Leftrightarrow g_j(a) \leq b_j, \quad j = \overline{1,M}. \tag{1.7}$$

Множество A представляет собой прямое произведение множеств $A_1, A_2, \ldots, A_m$, допустимое множество K задается системой M ограничений. В конкретных задачах A_i представляет собой: множество коммутационных полей, на которые может быть на-

значен *i*-й модуль в задаче разбиения, множество посадочных мест для *i*-го модуля в задаче размещения, группы контактов модулей для *i*-го инвариантного объекта (логического элемента, группы логических элементов, отдельных контактов) в задаче распределения инвариантных объектов, множество возможных физических реализаций (трасс) для *i*-й цепи в задачах расслоения и трассировки. Соответствующим образом интерпретируются и ограничения. Не теряя общности, можно рассматривать эквивалентную задаче (1.5)—(1.7) задачу комбинаторной оптимизации на прямом произведении m числовых множеств:

$$\pi^* = \arg \min_{\pi \in P \subseteq \Pi} \Phi(\pi), \tag{1.8}$$

$$\Pi = I_{n_1} \times I_{n_2} \times \ldots \times I_{n_m}, \tag{1.9}$$

$$P : \pi \in P \Leftrightarrow g_j(\pi) \le b_j, \quad j = \overline{1, M}, \tag{1.10}$$

где $\Phi(\pi) \equiv \Phi(a_\pi)$, $g_j(\pi) \equiv g_j(a_\pi)$, $j = \overline{1, M}$, $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, $a_\pi = \left(a^1_{\pi_1}, a^2_{\pi_2}, \ldots, a^m_{\pi_m}\right)$, $I_n = \{1, 2, \ldots, n\}$.

Далее покажем, как основные задачи конструкторского проектирования могут быть записаны в форме (1.8)—(1.10).

Для этого сформулируем необходимые определения.

Пусть имеется множество $V = \{v_i\}_{i=1}^{N_V}$ объектов, называемых в дальнейшем логическими контактами $(N_V \ge 0)$. Пусть имеется также множество цепей $U = \{u_i\}_{i=1}^{N_c}$, $N_C \ge 0$. Цепь представляет собой неупорядоченное подмножество логических контактов $u_i = \{v_{j_1^i}, v_{j_2^i}, \ldots, v_{j_{m_i}^i}\}$, $m_i = Card\, u_i$, $i = \overline{1, N_C}$. Число контактов в цепи u_i будем называть размерностью цепи u_i.

Множество U характеризуется следующими свойствами:

1) все логические контакты принадлежат цепям

$$\bigcup_{i=1}^{N_C} u_i = V; \tag{1.11}$$

2) цепи не имеют общих логических контактов

$$\bigcup_{i=1}^{N_C - 1} \bigcup_{j=i+1}^{N_C} u_i \cap u_j = \varnothing; \tag{1.12}$$

3) цепи содержат не менее двух контактов

$$m_i \geq 2\,,\ i = \overline{1, N_C}\,. \tag{1.13}$$

Пусть также имеется множество инвариантных объектов $E = \{e_i\}_{i=1}^{N_E}$, $N_E \geq 0$. Инвариантный объект представляет собой упорядоченное подмножество логических контактов $e_i = \left(v_{\tilde{j}_1^i}, v_{\tilde{j}_2^i}, \ldots, v_{\tilde{j}_{r_i}^i} \right)$, где r_i — число логических контактов в инвариантном объекте e_i. Обозначим $e_i' = \left\{ v_{\tilde{j}_1^i}, v_{\tilde{j}_2^i}, \ldots, v_{\tilde{j}_{r_i}^i} \right\}$ — соответствующее e_i неупорядоченное подмножество логических контактов, $r_i = Card\, e_i'$, $i = \overline{1, N_E}$.

Множество E характеризуется такими свойствами:

1) все логические контакты принадлежат инвариантным объектам

$$\bigcup_{i=1}^{N_E} e_i' = V\,; \tag{1.14}$$

2) инвариантные объекты не имеют общих контактов

$$\bigcup_{i=1}^{N_E - 1} \bigcup_{j=i+1}^{N_E} e_i' \cap e_j' = \varnothing\,; \tag{1.15}$$

3) инвариантные объекты содержат не менее одного контакта

$$r_i \geq 1\,,\ i = \overline{1, N_E}\,. \tag{1.16}$$

Тройку множеств $\{V,\ U, E\}$, удовлетворяющих свойствам (1.11)—(1.16), будем называть схемой. Множество цепей задает гиперграф $G(V,U)$, которому соответствует матрица инцидении $C = \|C_{ij}\|_{N_c \times N_V}$, $C_{ij} = \begin{cases} 1, & v_j \in U_i \\ 0, & v_j \notin U_i \end{cases}$. Для каждого логического контакта известен номер инвариантного объекта $\tilde{K}_j$, которому он принадлежит $\left(j = \overline{1, N_V},\quad \tilde{K}_j \in I_{N_E} \right)$, и порядковый номер $\tilde{n}_j$ этого контакта в инвариантном объекте $\left(j = \overline{1, N_V},\ \tilde{n}_j \in I_{r_{\tilde{K}_j}} \right)$. Введение инвариантных объектов структурирует множество ло-

гических контактов. Инвариантный объект представляет собой некоторую неделимую единицу, ему соответствует элемент логической схемы устройства.

Заданы также следующие множества.

Множество $\Omega = \{\Omega_i\}_{i=1}^{N_\Omega}$ коммутационных полей (конструктивов) — ограниченных замкнутых областей в двумерном евклидовом пространстве $\Omega_i \subset R^2$ $\left(i = \overline{1, N_\Omega}\right)$, представляющих собой φ-объекты [22]. На каждом конструктиве задана точка O_i^Ω — начало системы координат (полюс). Известны значения площадей конструктивов $S_i^\Omega \left(i = \overline{1, N_\Omega}\right)$.

Множество $H = \{H_i\}_{i=1}^{N_H}$ модулей — φ-объектов в R^2. Для каждого модуля известны положение на нем полюса O_i^H и его площадь S_i^H.

Множество $W = \{w_i\}_{i=1}^{N_W}$ объектов, называемых в дальнейшем физическими контактами $\left(N_W \geq N_V\right)$. Для физического контакта с номером j $\left(j = \overline{1, N_W}\right)$ известны номер модуля K_j, которому данный контакт принадлежит $\left(K_j \in I_{N_H}\right)$, и координаты контакта $(\tilde{x}_j,\ \tilde{y}_j)$ на K_j-м модуле в системе координат $X_{K_j} O_{K_j}^H Y_{K_j}$.

Множества $T^i = \{T_j^i\}_{j=1}^{P_i}$, $i = \overline{1, N_E}$ назначений для инвариантных объектов. Назначение для *i*-го инвариантного объекта представляет собой упорядоченное множество r_i попарно различных физических контактов $T_j^i = (w_{\gamma_1^{ij}}, w_{\gamma_2^{ij}}, \ldots, w_{\gamma_{r_i}^{ij}})$,

$$\forall i \in N_E \quad \forall_j \in I_{P_i} \quad \forall \alpha_1, \alpha_2 \in I_{r_i} \quad \alpha_1 \neq \alpha_2 \Rightarrow \gamma_{\alpha_1}^{ij} \neq \gamma_{\alpha_2}^{ij}. \quad (1.17)$$

С учетом введенных определений рассмотрим постановки основных задач конструкторского проектирования в форме (1.8)—(1.10), содержательную интерпретацию элементов множества П, способ записи критериев и ограничений.

Задача разбиения. Считаем, что задано исходное распределение инвариантных объектов, т. е. некоторый набор $\Psi = (\Psi_1, \Psi_2, \ldots, \Psi_{N_E})$, где $\Psi_i \in I_{n_i}$, $n_i \leq P_i$, $i = \overline{1, N_E}$, определяет номер $\tilde{\Psi}_i$ назначения

для инвариантного объекта e_i. Набор Ψ задает вариант электрической схемы устройства, т. е. схемы соединения физических контактов.

Множества A_i в данном случае представляют собой множества коммутационных полей, доступных для назначения i-го модуля, $m = N_H$. Для i-го модуля доступно $n_i \le N_\Omega$ полей. Вариант разбиения описывается вектором $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_{N_H})$, где $\kappa_i \in I_{n_i}$. Компоненту κ_i соответствует номер $\tilde{\kappa}_i$ коммутационного поля. Пусть $\Theta(\Psi, i)$ — оператор, определяющий номер модуля, к которому при данном распределении Ψ отнесен логический контакт v_i:

$$\Theta(\Psi, i) = K_{\gamma_{\tilde{n}_i}^{\tilde{K}_i \Psi \tilde{K}_i}}. \tag{1.18}$$

Тогда критерий качества, наиболее часто используемый в задачах разбиения, — суммарное число внешних связей блоков (конструктивов) — может быть записан в виде

$$\Phi(\kappa) = \sum_{t=1}^{N_\Omega} f_t(\kappa), \tag{1.19}$$

где

$$f_t(\kappa) = \sum_{j=1}^{N_C} \operatorname{sign}\left[\sum_{i=1}^{N_V} c_{ji} (1 - \delta(\tilde{\kappa}_{\Theta(\Psi,i)}, t)) \cdot \sum_{i=1}^{N_V} c_{ij} \delta(\tilde{\kappa}_{\Theta(\Psi,i)}, t) \right] \tag{1.20}$$

— число внешних связей t-го блока, $\delta(i, j) = \begin{cases} 1, i = j \\ 0, i \ne j \end{cases}$.

Ограничения в задаче разбиения накладываются обычно на число внешних связей каждого блока:

$$f_t(\kappa) \le N_t, \; t = \overline{1, N_\Omega}, \tag{1.21}$$

где N_t — максимально допустимое число внешних связей t-го блока, и на суммарную площадь модулей в блоках:

$$\sum_{i=1}^{N_H} S_i^H \delta(\tilde{\kappa}_i, t) \le \alpha_t S_t^\Omega, \tag{1.22}$$

где α_t — предельный коэффициент заполнения блока, $0 \le \alpha_t \le 1$.

Частным случаем задачи разбиения является задача компоновки, заключающаяся в распределении инвариантных объектов до размещения модулей. Критерием качества распределения

$\Psi = \left(\Psi_1, \Psi_2, \ldots, \Psi_{N_E}\right)$ служит, как правило, суммарное число внешних связей модулей:

$$\Phi(\Psi) = \sum_{t=1}^{N_H} f_t(\Psi), \tag{1.23}$$

где

$$f_t(\Psi) = \sum_{j=1}^{N_C} \operatorname{sign}\left[\sum_{i=1}^{N_V} c_{ji}\left(1 - \delta\left(\Theta(\Psi, i), t\right)\right) \cdot \sum_{i=1}^{N_V} c_{ji}\left(\delta\left(\Theta(\Psi, i), t\right)\right)\right] \tag{1.24}$$

— число внешних связей t-го модуля. Ограничениями компоновки являются ограничения на непересечение назначений инвариантных объектов (два логических контакта не могут быть назначены на один физический):

$$\delta\left(\gamma_{\tilde{n}_i}^{\tilde{K}_i \tilde{\Psi} \tilde{K}_i}, \gamma_{\tilde{n}_j}^{\tilde{K}_j \tilde{\Psi} \tilde{K}_j}\right) = 0, \ i, j = \overline{1, N_V}, \ i \neq j. \tag{1.25}$$

Задача размещения. Будем считать, что задано исходное распределение инвариантных объектов — вектор Ψ. Модули H_i, $i = \overline{1, N_H}$, необходимо разместить в области Ω_t, $t \in I_{N_\Omega}$. При неизменной ориентации модулей положение любой области H_i в области Ω_t можно однозначно описать двумя параметрами — координатами (x_i, y_i) точки O_i^H в системе координат $X O_t^\Omega Y$. Размещение всех N_H модулей, следовательно, может быть описано вектором параметров размерности $2N_H$: $\left(x_1, \ldots, x_{N_H}, y_1, \ldots, y_{N_H}\right)$ [22]. В дискретном варианте задачи для каждой области H_i задается набор координат разрешенных для нее q_i посадочных мест $\left(x_j^{i,t}, y_j^{i,t}\right)$, $j = \overline{1, q_i}$. Введение в рассмотрение нескольких разрешенных ориентаций для каждого модуля добавляет в список координат посадочных мест соответствующий угол поворота и принципиально не изменяет последующих рассуждений.

Множества A_i представляют собой, таким образом, множества доступных для i-го модуля посадочных мест, $m = N_H$. Для i-го модуля доступно $n_i \leq q_i$ посадочных мест. Вариант размещения описывается вектором $\varphi = \left(\varphi_1, \varphi_2, \ldots, \varphi_{N_H}\right)$, где $\varphi_i \in I_{n_i}$. Компоненту φ_i соответствует номер $\tilde{\varphi}_i$ посадочного места.

Традиционный критерий качества размещения — суммарная длина полупериметров минимальных охватывающих прямоугольников цепей — записывается в виде

$$\Phi(\varphi) = \sum_{j=1}^{N_c}\left(\max_{i:c_{ji}=1}\left(\tilde{x}_n + x_{\tilde{\varphi}_K}^{K,t}\right) - \min_{i:c_{ji}=1}\left(\tilde{x}_n + x_{\tilde{\varphi}_K}^{K,t}\right)\right) + \\ + \left(\max_{i:c_{ji}=1}\left(\tilde{y}_n + y_{\tilde{\varphi}_K}^{K,t}\right) - \min_{i:c_{ji}=1}\left(\tilde{y}_n + y_{\tilde{\varphi}_K}^{K,t}\right)\right), \quad (1.26)$$

где $n = \gamma_{\tilde{n}_i}^{\tilde{K}_i\tilde{\Psi}_{\tilde{K}_i}}$, $K = \Theta(\Psi, i)$.

Ограничения размещения включают требования непересечения областей и расположения внутри области Ω_t:

$$S\left(\left[\bigcup_{i=1}^{N_H} H_i\left(x_{\tilde{\varphi}_i}^{i,t}, y_{\tilde{\varphi}_i}^{i,t}\right)\right] \cap \Omega_t\right) = \sum_{i=1}^{N_H} S_i^H, \quad (1.27)$$

где $S(\cdot)$ — площадь соответствующей области, $H_i(x, y)$ — область H_i, расположенная в Ω_t таким образом, что полюс O_i^H находится в точке с координатами (x, y) в системе координат $X O_t^{\Omega} Y$.

Задача распределения инвариантных объектов. Считаем, что размещение модулей в области Ω_t проведено, задан вектор $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_{N_H})$.

Множества A_i представляют собой множества вариантов назначения для *i*-го инвариантного объекта, $m = N_E$. Для *i*-го инвариантного объекта доступно $n_i \le p_i$ назначений. Вариант распределения инвариантных объектов описывается вектором $\Psi = (\Psi_1, \Psi_2, \ldots, \Psi_{N_E})$, где $\Psi_i \in I_{n_i}$. Компоненту Ψ_i соответствует номер назначения $\tilde{\Psi}_i$.

Критерием качества распределения инвариантных объектов может быть критерий размещения — суммарная длина полупериметров минимальных описанных прямоугольников цепей:

$$\Phi(\Psi) = \sum_{j=1}^{N_c}\left(\max_{i:c_{ji}=1}\left(\tilde{x}_n + x_{\tilde{\varphi}_K}^{K,t}\right) - \min_{i:c_{ji}=1}\left(\tilde{x}_n + x_{\tilde{\varphi}_K}^{K,t}\right)\right) + \\ + \left(\max_{i:c_{ji}=1}\left(\tilde{y}_n + y_{\tilde{\varphi}_K}^{K,t}\right) - \min_{i:c_{ji}=1}\left(\tilde{y}_n + y_{\tilde{\varphi}_K}^{K,t}\right)\right), \quad (1.28)$$

где $n = \gamma_{\tilde{n}_i}^{\tilde{K}_i\tilde{\Psi}_{\tilde{K}_i}}$, $K = \Theta(\Psi, i)$.

В качестве ограничений выступают требования непересечения инвариантных объектов (два логических контакта не могут быть назначены на один физический):

$$\delta\left(\gamma_{\tilde{n}_i}^{\tilde{K}_i\tilde{\Psi}_{\tilde{K}_i}},\gamma_{\tilde{n}_j}^{\tilde{K}_j\tilde{\Psi}_{\tilde{K}_j}}\right)=0\ ,\ i,j=\overline{1,N_V}\ ,\ i\neq j\ . \qquad (1.29)$$

Задача расслоения. Будем считать, что заданы распределение инвариантных объектов и размещение модулей, т. е. известны векторы Ψ и φ.

В задаче расслоения множества A_i представляют собой множества трасс для i-й цепи (моделируемых, например, описанными прямоугольниками или трассами известной конфигурации), расположенные на различных слоях, $m = N_c$. Всего в конструкции задано N_S слоев. Для i-й цепи доступно $n_i \le N_S$ слоев. Вариант расслоения задается вектором $\sigma=\left(\sigma_1,\sigma_2,\ldots,\sigma_{N_c}\right)$, где $\sigma_i \in I_{n_i}$. Компоненту σ_i соответствует номер слоя $\tilde{y}_i$.

Критерием качества чаще всего выступает сумма коэффициентов конфликтности цепей:

$$\Phi(\sigma)=\sum_{i=1}^{N_c-1}\sum_{j=i+1}^{N_c} f_{ij}\delta\left(\tilde{\sigma}_i,\tilde{\sigma}_j\right), \qquad (1.30)$$

где f_{ij} — коэффициент конфликтности цепей с номерами i и j.

Коэффициенты конфликтности могут быть пропорциональны, например, площади пересечения описанных прямоугольников цепей. Ограничения в задаче расслоения обычно не накладываются.

Задача трассировки. Будем считать, что все предыдущие задачи решены. Положение всех логических контактов на коммутационном поле известно.

Множества A_i представляют собой множества трасс (физических реализаций) для i-й цепи, $m = N_C$. Трасса рассматривается, как связанное точечное множество, φ — объект в пространстве R^3 [22] и включает в себя все соответствующие физические контакты. Для i-й цепи рассматривается h_i возможных трасс. Вариант трассировки задается вектором $\zeta=\left(\zeta_1,\zeta_2,\ldots,\zeta_{N_C}\right)$, где $\zeta_i \in I_{n_i}$,

$n_i \le h_i$. Компоненту ζ_i соответствует номер трассы $\tilde{\zeta}_i$. Процесс перечисления трасс может быть организован с использованием некоторых эвристических приемов [23—25] и в общем случае имеет значительную вычислительную сложность.

Наиболее распространенный критерий качества трассировки — суммарная длина выбранных трасс:

$$\Phi(\zeta) = \sum_{i=1}^{N_C} l\left(a^i_{\tilde{\zeta}_i}\right). \qquad (1.31)$$

Среди ограничений выделяются геометрические (ограничения непересечения):

$$S\left(a^i_{\tilde{\zeta}_i} \cap a^j_{\tilde{\zeta}_j}\right) = 0, \quad i, j = \overline{1, N_C}, \quad i \ne j, \qquad (1.32)$$

и функциональные (взаимные емкости трасс):

$$c\left(a^i_{\tilde{\zeta}_i}, a^j_{\tilde{\zeta}_j}\right) \le \overline{c}, \quad i, j = \overline{1, N_C}, \quad i \ne j. \qquad (1.33)$$

Далее для простоты изложения рассматриваем задачу трассировки в канале [26]. Канальная модель позволяет эффективно организовывать перечисление трасс, подсчет критерия и ограничений. Тем не менее специальный вид модели не ограничивает общности получаемых результатов. Содержание множеств A_i качественно остается тем же, изменяются лишь способы перечисления трасс, подсчета ограничений и критерия.

Считаем канал горизонтальным, общее число магистралей канала — N_m. В любом сечении канала известны номера допустимых для трассировки магистралей. Физические контакты располагаются вдоль верхней и нижней границ канала и характеризуются своими горизонтальными координатами относительно некоторого условного левого края канала. Для верхнего края канала координаты считаются положительными, для нижнего — отрицательными, магистрали в канале нумеруются сверху вниз. В канале необходимо развести N цепей. Цепи с номером i соответствует список координат ее физических контактов $\left\{x^i_1, x^i_2, \ldots, x^i_{m_i}\right\}$, где m_i — количество контактов данной цепи в канале, причем $\left|x^i_j\right| \le \left|x^i_{j+1}\right|$, $j = \overline{1, m_i - 1}$. Цепь может занимать не более одной магистрали, поэтому вариант трассы полностью определяется заданием номера магистрали цепи $\tilde{\zeta}_i$. Таким обра-

зом, организуется перечисление трасс. Вектор $\zeta = (\zeta_1, \zeta_2, \ldots, \zeta_N)$, где $\zeta_i \in I_{n_i}$, $n_i \le N_m$, определяет вариант трассировки цепей в канале, горизонтальный участок трассы *i*-й цепи проходит по магистрали с номером $\tilde{\zeta}_i$.

Критерий (1.31) для канальной модели записывается в виде

$$\Phi(\zeta) = \sum_{i=1}^{N} \left[\left| x^i_{m_i} \right| - \left| x^i_1 \right| + \sum_{k=1}^{m_i} \left(\frac{1 - \operatorname{sign}(x^i_k)}{2} N + \operatorname{sign}\left(x^i_k\right) \tilde{\xi}_i \right) \right]. \qquad (1.34)$$

Ограничения непересечения (1.32) принимают вид

$$\sum_{k=1}^{m_i} \sum_{s=1}^{m_j} \delta\left(x^i_k, -x^j_s\right) \max\left\{0, \quad \operatorname{sign}\left(x^i_k\right)\tilde{\xi}_i + \operatorname{sign}\left(x^j_s\right)\tilde{\xi}_i + 1\right\} +$$

$$+ \delta\left(\tilde{\xi}_i, \tilde{\xi}_j\right) \max\left\{0, \min\left\{\left|x^i_{m_i}\right|, \left|x^j_{m_j}\right|\right\} - \max\left\{\left|x^i_1\right|, \left|x^j_1\right|\right\} + 1\right\} \le 0, \qquad (1.35)$$

$$i, j = \overline{1, N}, \quad i \ne j.$$

Таким образом, основные задачи конструкторского проектирования МЭУ могут быть формально записаны в виде выражений (1.8)—(1.10). Отметим преимущества такой постановки.

Модель (1.9) предпочтительнее традиционных перестановочных моделей для задач конструкторского проектирования. Перестановка m чисел не позволяет непосредственно описывать выбор элементов множеств $A_i \left(i = \overline{1, m}\right)$ в случае частичного несовпадения множеств, разной их мощности, возможности выбора из нескольких множеств элементов с одним и тем же номером. Использование для этих целей перестановочных моделей требует дополнительных алгоритмических приемов [27]. Модель (1.9) допускает частичное несовпадение, пустое пересечение либо даже вообще различную природу множеств A_i. Это позволяет использовать ее не только при формальной постановке всех основных задач конструкторского проектирования, но и при совместной постановке нескольких задач.

Так как выбор конкретной оптимизационной модели во многом определяет структуру алгоритма, то модель (1.9) порождает новые алгоритмы конструкторского проектирования, ранее для решения соответствующих задач не использовавшиеся.

Модель (1.9) позволяет использовать для метризации множества П метрики множеств A_i. Такой подход больше соответствует

конкретному содержанию задачи, чем использование традиционных метрик комбинаторных множеств.

Постановка (1.8)—(1.10) дает возможность, в отличие от перестановочных моделей, эффективно описывать алгоритмы последовательного типа, объединять начальные алгоритмы и алгоритмы оптимизации начального решения. Постановка (1.8)—(1.10), в отличие от постановки на множестве перестановок, позволяет эффективно моделировать ослабление ограничений (например, возможность установки нескольких модулей на одно посадочное место), агрегацию элементов множеств A_i.

Альтернативой множеству перестановок в математических моделях задач конструкторского проектирования МЭУ могут быть комбинаторные множества, обладающие заданными комбинаторными свойствами, соответствующими решаемой задаче. Конструктивным средствам описания, генерации и перечисления комбинаторных множеств, обладающих заданными свойствами, посвящена глава 2.

Специфической особенностью задач конструкторского проектирования является относительно малая мощность области допустимых решений P: $Card\ P << Card\ \Pi$. Это объясняется наличием большого количества ограничений, прежде всего геометрических. Поэтому на практике часто даже получение допустимого решения $\pi \in P$ является трудноразрешимой задачей. Допустимое решение может быть получено в результате оптимизации некоторого недопустимого решения. Кроме того, в некоторых случаях искусственное выведение процесса оптимизации из области допустимых решений может существенно улучшить качество окончательного решения, так как область P, как правило, многосвязна. Необходимость сравнения качества искусственных решений требует доопределения критерия $\Phi(\pi)$ (функции полезности) для соответствующих элементов множества Π. Поэтому предлагается осуществить переход от задачи оптимизации (1.8)—(1.10) к задаче безусловной оптимизации с помощью введения ограничений в критерий и построения штрафной функции:

$$F(\pi) = \Phi(\pi) + \sum_{j=1}^{M} \lambda_j \cdot (g_j(\pi) - b_j)_+ , \qquad (1.36)$$

где $\lambda_j \geq 0$ — штрафные коэффициенты, $(\alpha)_+ = \begin{cases} \alpha, \alpha \geq 0 \\ 0, \alpha < 0 \end{cases}$.

Соответствующая преобразованная оптимизационная задача записывается в виде

$$\pi^* = \arg\min_{\pi \in \Pi} F(\pi), \tag{1.37}$$

$$\Pi = I_{n_1} \times I_{n_2} \times \ldots \times I_{n_m}. \tag{1.38}$$

Переход к преобразованной задаче оптимизации позволяет существенно повысить эффективность оптимизационных методов при синтезе конструкций МЭУ. При расчете штрафных коэффициентов могут быть использованы различные известные подходы [28].

1.3. ОБЩАЯ СХЕМА АЛГОРИТМА РЕШЕНИЯ ОПТИМИЗАЦИОННЫХ ЗАДАЧ КОНСТРУКТОРСКОГО ПРОЕКТИРОВАНИЯ

В этом параграфе рассмотрен формальный процесс построения оптимизационных алгоритмов конструкторского проектирования. Введен способ метризации комбинаторного пространства, учитывающий содержание конкретной задачи. Предложена общая схема итерационно-последовательного алгоритма для задач конструкторского проектирования. Рассмотрены критерии выбора множеств в схеме последовательного алгоритма, использующие элементы теории статистических решений. Рассмотрен выбор множеств в схеме итерационного алгоритма. Исследованы способы повышения глобальных свойств алгоритмов.

1.3.1. Метризация комбинаторного пространства

Любой алгоритм комбинаторной оптимизации так или иначе структурирует процесс перебора элементов комбинаторного множества. Алгоритм, как правило, заканчивает работу либо по истечении заданного времени, либо при выполнении условий локальной оптимальности. Будем строить процесс поиска решения задачи (1.37)—(1.38) по общей схеме алгоритмов локальной оптимизации:

$$\pi^{*t+1} = \arg \min_{\pi \in R_{t+1} \subset \Pi,\ \pi^{*t} \in R_{t+1}} F(\pi), \quad t = 0,1,\ldots \tag{1.39}$$

Способ формирования подмножеств R_{t+1} может быть произвольным. Традиционное использование для этих целей метрики

множества П не обязательно: определение локального экстремума не требует введения метрики на соответствующем множестве. Подмножества R_i могут формироваться, например, случайно, либо задаваться некоторой топологической структурой на множестве П. Способ формирования подмножеств может оказать существенное влияние на глобальные свойства алгоритма оптимизации.

Рассмотрим возможность введения метрики на множестве П. Данный вопрос является важным, так как успех оптимизации во многом определяется соответствием метрики содержанию задачи.

Метрика

$$\rho'\left(\pi^1,\ \pi^2\right)=\sum_{i=1}^{m}\left(1-\delta\left(\pi_i^1,\ \pi_i^2\right)\right), \tag{1.40}$$

где $\delta(i,j)=\begin{cases}1, i=j\\0, i\neq j\end{cases}$, является во многом традиционной для комбинаторных задач и родственна транспозиционной метрике [22] множества перестановок.

Расстояние между двумя элементами множества П определяется количеством попарно различных компонентов соответствующих векторов. Несложно показать, что функционал (1.40) является метрикой. Выполнение аксиом тождества и симметрии очевидно.

Докажем выполнение аксиомы треугольника.

Пусть $\rho'\left(\pi^1,\ \pi^2\right)=r_1$, $\rho'\left(\pi^2,\ \pi^3\right)=r_2$. Тогда $\rho\left(\pi^1,\ \pi^3\right)=$

$$=\sum_{i=1}^{m}\left(1-\delta\left(\pi_i^1,\ \pi_i^3\right)\right)=\sum_{i:\delta\left(\pi_i^1,\pi_i^2\right)\cdot\delta\left(\pi_i^3,\pi_i^2\right)=0}\left(1-\delta\left(\pi_i^1,\ \pi_i^3\right)\right)\le m-\sum_{i=1}^{m}\delta\left(\pi_i^1,\pi_i^2\right)\delta(\pi_i^3,$$

$$\pi_i^2)\le m-\left(m-r_1-r_2\right)=r_1+r_2.$$

Утверждение доказано.

Метрика (1.40) не связана с содержанием конкретной задачи. Данное обстоятельство — одновременно и преимущество, и недостаток. Преимуществом является общность получаемого алгоритма оптимизации, недостатком — относительно низкая эффективность алгоритма при решении каждой конкретной задачи. Для учета содержательных особенностей задачи множество П можно метризовать, используя метрики исходных множеств A_i:

$$\rho''\left(\pi^1,\ \pi^2\right)=\max_{1\le i\le m}\alpha_i\rho_i\left(a^i_{\pi_i^1},\ a^i_{\pi_i^2}\right), \tag{1.41}$$

где $\rho_i(.)$ — метрика множества $A_i\left(i=\overline{1,m}\right)$, $\alpha_i > 0$ — весовой коэффициент. Тогда близость элементов множества П в смысле метрики (1.41) будет соответствовать «близости» соответствующих элементов множеств A_i (посадочных мест, трасс и т. п.).

Покажем, что функционал (1.41) является метрикой. Аксиомы тождества и симметрии выполняются по построению.

Докажем выполнение аксиомы треугольника: $\rho''\left(\pi^1,\ \pi^3\right) \le$ $\le \rho''\left(\pi^1,\ \pi^2\right)+\rho''\left(\pi^3,\ \pi^2\right)$. Пусть $i_0 = \arg\max\limits_{1\le i\le m} \alpha_i\rho_i\left(a^i_{\pi^1_i},\ a^i_{\pi^3_i}\right)$. Тогда

$$\rho''\left(\pi^1,\ \pi^3\right)=\alpha_{i_0}\rho_{i_0}\left(a^{i_0}_{\pi^1_{i_0}},\ a^{i_0}_{\pi^3_{i_0}}\right)\le \alpha_{i_0}\rho_{i_0}\left(a^{i_0}_{\pi^1_{i_0}},\ a^{i_0}_{\pi^2_{i_0}}\right)+\alpha_{i_0}\rho_{i_0}\left(a^{i_0}_{\pi^3_{i_0}},\ a^{i_0}_{\pi^2_{i_0}}\right)\le$$

$$\le \max_{1\le i\le m}\alpha_i\rho_i\left(a^i_{\pi^1_i},\ a^i_{\pi^2_i}\right)+\max_{1\le i\le m}\alpha_i\rho_i\left(a^i_{\pi^3_i},\ a^i_{\pi^2_i}\right)=\rho''\left(\pi_1,\pi_2\right)+\rho''\left(\pi_3,\pi_2\right).$$

Утверждение доказано.

Необходимо доопределить метрики множеств A_i для элементов a^i_0 $\left(i=\overline{1,m}\right)$, соответствующих отсутствию выбора. Сделаем это следующим образом:

$$\forall j\in I_{n_i}\qquad \rho_i\left(a^i_0,a^i_0\right)=0,\ \ \rho_i\left(a^i_0,a^i_j\right)=\rho_i\left(a^i_j,a^i_0\right)=\varepsilon,\ \ i=\overline{1,m},$$

где $\varepsilon=\dfrac{1}{2}\min\limits_{1\le i\le m}\min\limits_{n\ne k}\rho_i\left(a^i_n,\ a^i_k\right)$.

Функционалы $\rho_i(\cdot)$, $i=\overline{1,m}$ (следовательно, и $\rho''(\cdot)$), доопределенные таким образом, перестают быть метриками и становятся симметриками, так как для элементов a^i_0 не выполняется неравенство треугольника. Однако это не мешает использованию функционала $\rho''(\cdot)$ для формирования подмножеств R_t в алгоритме оптимизации.

Строим подмножества R_t как пересечения соответствующих окрестностей:

$$R_t = R'_t\cap R''_t,\quad R'_t=\left\{\pi\mid\rho'\left(\pi,\ \pi^{*t-1}\right)\le r'_t\right\},$$

$$R''_t=\left\{\pi\mid\rho''\left(\pi,\ \pi^{*t-1}\right)\le r''_t\right\},\ \ t=1,2,\ldots.$$

При этом радиус окрестности r'_t в метрике $\rho'(.)$ ограничивает количество одновременно рассматриваемых в процессе оптими-

зации множеств A_i, а радиус окрестности r_t'' в симметрике $\rho''(.)$ ограничивает «перемещения» модулей, цепей, инвариантных объектов. Весовые коэффициенты в (1.41) задают, по сути, радиусы окрестностей в симметриках каждого из множеств A_i:

$$\rho''(\pi^1, \pi^2) = \max_{1 \le i \le m} \alpha_i \rho_i \left(a^i_{\pi^1_i}, a^i_{\pi^2_i}\right) \le r_t'' \Rightarrow \rho_i \left(a^i_{\pi^1_i}, a^i_{\pi^2_i}\right) \le \frac{1}{\alpha_i} r_t'', \quad i = \overline{1,m}.$$

При $\alpha_i = 1,\ i = \overline{1,m}$ в ε-окрестности каждого из элементов a^i_j, $j = \overline{1,n_i}$, находится элемент a^i_0, соответствующий отсутствию выбора, а в ε-окрестности элемента a^i_0 находятся все элементы $a^i_j,\ j = \overline{1,n_i}$.

Введение симметрики (1.41), позволяющее более адекватно содержанию задачи упорядочить элементы абстрактного комбинаторного множества П, является эффективным средством ускорения процесса оптимизации за счет отделения менее перспективных вариантов. Такой прием позволяет естественным образом описывать, например, алгоритмы перестановок соседних модулей, обобщать данные алгоритмы на другие задачи конструирования. Действительно за один проход в алгоритме парных перестановок соседних (в смысле близости их посадочных мест) модулей [29] исследуется подмножество решений R_t, образованное пересечением окрестностей текущего варианта решения, ограниченных в метрике $\rho'(.)$ радиусом $r_t' = 2$ (парные перестановки), а в симметрике $\rho''(.)$ — радиусом $r_t'' = 1$ (соседние модули). Метрика множеств посадочных мест при этом обычно берется ортогональной или евклидовой.

При $r_t' = 4,\ r_t'' = 1$ получаем вариант алгоритмов сканирования [30]. В отсутствие ограничений на радиус окрестности в симметрике $\rho''(.)$ при $r_t' = 2, 3,\ldots$ получаем варианты алгоритма парных перестановок, полного троичного перебора и т. д. [31]. Ограничения на радиус окрестности в симметрике $\rho''(.)$ качественно снижают вычислительную сложность алгоритма. Если сложность обычного алгоритма парных перестановок можно оценить как $0(m^2)$, то сложность алгоритма парных перестановок соседних модулей оценивается как $0(m)$.

Отметим, что алгоритмы перебора решений в подмножествах R_t при $r_t' = 2$ тождественны алгоритмам парных перестановок лишь для узкого класса задач размещения. Алгоритмы перебора элементов подмножеств R_t при $r_t' = 2$ являются более общими и могут быть названы алгоритмами парных переназначений.

1.3.2. Общая схема итерационно-последовательного алгоритма

В качестве общей схемы при построении алгоритма решения задач (1.35) предлагается схема итерационно-последовательного алгоритма. Схема обобщает опыт разработки алгоритмов конструирования, а также совместно с введенным способом формирования подмножеств R_t позволяет моделировать широкий набор известных оптимизационных алгоритмов и предложить принципиально новые. Схема включает два алгоритма, названных условно итерационными и последовательными. Итерационный алгоритм на каждом шаге выделяет некоторый набор множеств A_i, $i \in I \subseteq I_m$, для которых решает соответствующую оптимизационную задачу последовательным алгоритмом. Последовательный алгоритм в процессе работы может использовать итерационный алгоритм для оптимизации промежуточных частичных решений, для которых значение критерия превысило предварительную оценку.

Логика создания итерационно-последовательной схемы основывается на замене понятия перестановки понятием переназначения при переходе от перестановочной модели к прямому произведению числовых множеств, замене полного исследования подмножества R_t приближенным решением. При этом использованы идеи методов сечений (покоординатного спуска) — выбор итерационным алгоритмом сечения R_t размерности p и последующий перебор последовательным алгоритмом сечений единичной размерности. Управление размерностью выбираемых итерационным алгоритмом сечений обеспечивается возможностью вызова итерационного алгоритма последовательным. Итерационный алгоритм определяет подмножество модулей, цепей, инвариантных объектов, а последовательный выполняет последовательное назначение этому множеству посадочных мест, трасс, физических контактов.

Итерационный алгоритм реализуется оператором B:

$$\pi^* = B\left(\pi^0, R, I\right), \tag{1.42}$$

где π^0 — начальное решение, $R \subseteq \Pi$, $I \subseteq I_m$,

$$R = J_1 \times J_1 \times \ldots \times J_m,\ J_i = \begin{cases} \pi_i^0, & i \notin I \\ J_i' \subseteq \{0,1,\ldots,n_i\}, & i \in I,\ \pi_i^0 \in J_i' \end{cases}.$$

На n-м шаге алгоритма оператором выбора $\tilde{W}$ выбирается подмножество индексов $I^n \subset I$, $I^n = \tilde{W}\left(I, n\right)$. Соответствующее решение на данном шаге имеет вид

$$\pi^n = \begin{cases} D\left(\pi^{n-1}, R, I^n\right), & F\left(D\left(\pi^{n-1}, R, I^n\right)\right) < F\left(\pi^{n-1}\right) \\ \pi^{n-1}, F\left(D\left(\pi^{n-1}, R, I^n\right)\right) \geq F\left(\pi^{n-1}\right) \end{cases}, \tag{1.43}$$

где $D(.)$ — оператор, реализующий последовательный алгоритм.

Решение задачи полагают равным $\pi^* = \pi^h$, где h — номер шага, на котором выполнилось условие останова алгоритма. Последовательный алгоритм реализуется оператором D:

$$\pi^* = D\left(\pi^0, R, I\right) \tag{1.44}$$

и представляет собой p-шаговую процедуру, $p = Card\, I$. На k-м шаге оператором выбора $\tilde{V}$ выбирается индекс $j_k = \tilde{V}\left(I, \{j_1, j_2, \ldots, j_{k-1}\}\right)$, $j_k \in I \setminus \{j_1, j_2, \ldots, j_{k-1}\}$.

Соответствующее решение на данном шаге равно

$$\pi^k = \begin{cases} \tilde{\pi}^k, F\left(\pi^{k-1}\right) - F\left(\tilde{\pi}^k\right) \geq \tau_k \\ B\left(\pi^{k-1}, R, \{j_1, j_2, \ldots, j_k\}\right), F\left(\pi^{k-1}\right) - F\left(\tilde{\pi}^k\right) < \tau_k \end{cases}, \tag{1.45}$$

где

$$\tilde{\pi}^k = \arg \min_{\pi \in \Pi_1^k \times \Pi_2^k \times \ldots \times \Pi_m^k} F(\pi), \Pi_i^k = \begin{cases} J_i', & i = j_k \\ 0, & i \in I \setminus \{j_1, j_2, \ldots, j_k\} \\ \pi_i^{k-1}, i \in \left(I_m \setminus I\right) \cup \{j_1, j_2, \ldots, j_{k-1}\} \end{cases}. \tag{1.46}$$

Решение задачи полагают равным $\pi^* = \pi^p$. Параметр τ_k представляет собой предварительную оценку уменьшения критерия. Реальное уменьшение критерия может быть меньше τ_k, например,

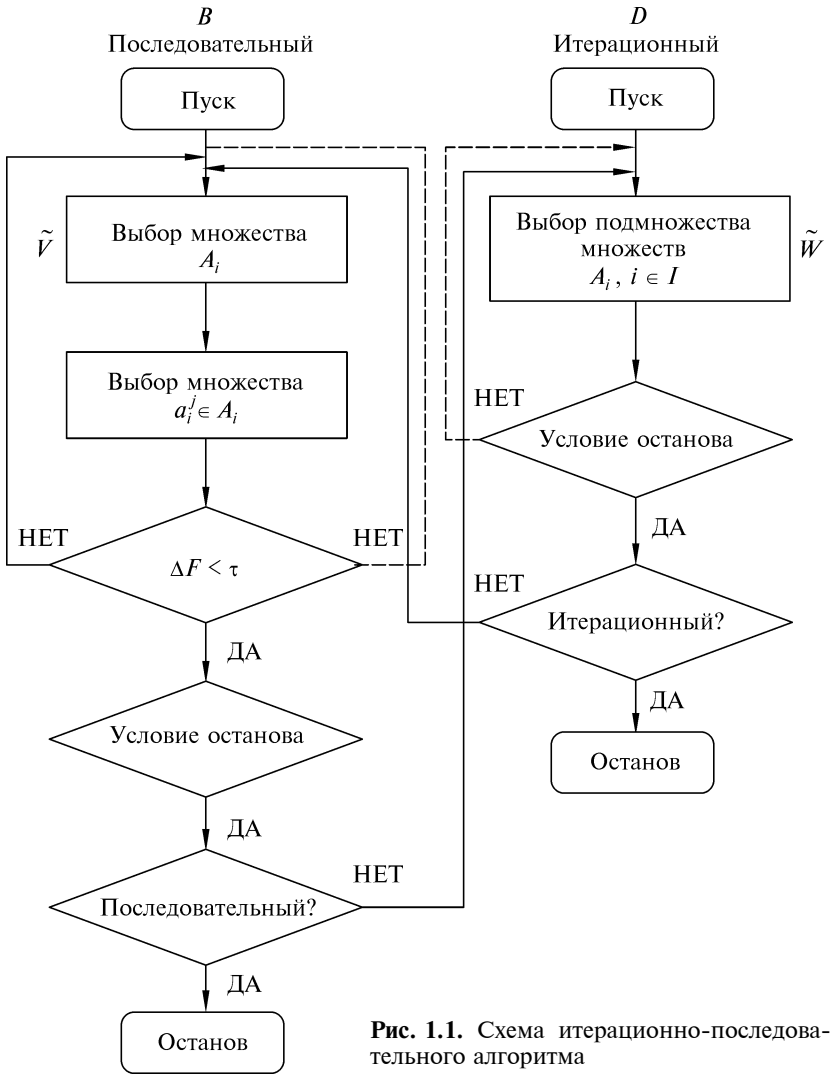

Рис. 1.1. Схема итерационно-последовательного алгоритма

в случае нарушения ограничений при выборе $a^{j_k}_{\pi^k_{jk}}$. Тогда проводится повторный выбор части элементов некоторых множеств, т. е. моделируется процесс снятия-переразмещения, снятия-перетрассировки и т. п. Аналогичный процесс может иметь место, например, при превышении трассой допустимой длины.

Графически схема итерационно-последовательного алгоритма изображена на рис. 1.1. Схема имеет два входа: алгоритм может быть инициализирован либо как последовательный, либо как итерационный. В первом случае последовательный алгоритм применяется ко всей задаче (1.39), во втором — к последовательности подзадач, генерируемых итерационным алгоритмом. Количество вложений вызовов последовательным алгоритмом итерационного (процедура снятия переназначения) может быть ограничено либо заданием максимально допустимого числа вложений, либо постепенным уменьшением оценок τ_K.

Задание радиуса r_t' окрестности R_t' ограничивает размерность p генерируемых итерационным алгоритмом подзадач.

Задание радиуса r_t'' ограничивает «перемещения» модулей, инвариантных объектов, цепей. Если модуль (инвариантный объект, цепь) уже размещен, то при $\frac{1}{\alpha_i} r_t'' > \varepsilon$ он может быть либо перемещен на расстояние, не превышающее $\frac{1}{\alpha_i} r_t''$, либо снят. Если модуль (инвариантный объект, цепь) еще не размещен, то он может быть размещен на любое доступное ему посадочное место (в силу доопределения симметрики (1.41)). При $\frac{1}{\alpha_i} r_t'' > \varepsilon$ модуль не может быть ни снят (если он размещен), ни размещен (если он снят). «Свобода» перемещения модулей, инвариантных объектов, цепей при заданном r_t'' определяется значениями весовых коэффициентов $\alpha_i \quad \left(i = \overline{1,m}\right)$.

Таким образом, итерационно-последовательная схема практически стирает различия между традиционно относимыми к различным классам алгоритмами итерационного и последовательного типов, алгоритмами получения начального решения и алгоритмами улучшения начального решения.

1.3.3. Критерии выбора множеств в схеме последовательного алгоритма

Последовательный алгоритм включает две основные операции — выбор множества из списка I и выбор элемента данного множества (см. рис. 1.1). В большинстве алгоритмов, использу-

мых на практике, эти две операции отделены друг от друга и выполняются независимо. На этом принципе построены алгоритмы разбиения [32], размещения [26], распределения инвариантных объектов [29], расслоения [33], трассировки [26]. Выбор элементов осуществляется, как правило, полным перебором либо оценкой всех элементов множества.

В простейшем случае множества списка I могут выбираться либо случайно, либо в естественном порядке (в порядке нумерации). Однако такой способ выбора малоэффективен. Создание более сложных схем выбора имеет целью максимально компенсировать основной недостаток последовательных алгоритмов: невозможность полностью учесть вклад в конечное значение критерия тех множеств, элементы из которых еще не выбраны. Иными словами, при выборе очередного модуля (цепи) слабо учитывается его влияние на еще не размещенные модули (цепи).

Попыткой придать критерию выбора максимальную достоверность стало введение принципа «максимальной конъюнкции—минимальной дизъюнкции» [34], максимизирующего «меру связности» выбираемого множества A_{j_t} с множествами, из которых уже произведен выбор элементов $(A_n : n \in I_m, \pi_n > 0)$, и минимизирующего «меру связности» с множествами, из которых выбор элементов еще не произведен $(A_n : n \in I_m, \pi_n = 0)$. В качестве «меры связности» μ в задачах разбиения, размещения, распределения инвариантных объектов используется количество цепей, связывающих модуль j_t с соответствующим подмножеством модулей [26], а в задачах расслоения и трассировки — степень конфликтности цепи j_t с соответствующим подмножеством цепей [33], [35]. Действительно, чем меньше «связность» множества A_{j_k} с множествами $A_n : n \in I_m$, $\pi_n = 0$, тем меньше «неопределенность» при выборе элементов из данного множества. Формально один из вариантов критерия максимальной конъюнкции—минимальной дизъюнкции может быть записан в виде

$$\tilde{V}_1 : j_k = \arg \max_{j \in I \setminus I^{k-1}} \left(\mu\left(j, \{n \mid n \in I_m, \pi_n > 0\}\right) - \mu\left(j, \{n \mid n \in I_m, \pi_n = 0\}\right)\right), \tag{1.47}$$

где $I^k = \{j_1, j_2, \ldots, j_k\}$, $I^0 = \varnothing$.

Дальнейшее развитие критериев выбора привело к частичному совмещению операций выбора множества и элемента, к более

полному учету влияния множеств списка $I \setminus I^t$ и выразилось в создании матричных схем размещения модулей [26]. В данных алгоритмах реализованы принципы выбора стратегий в условиях неопределенности, известные в теории статистических решений [36]. Поскольку матричные алгоритмы зарекомендовали себя как один из наиболее эффективных среди алгоритмов последовательного типа [26], представляет интерес рассмотреть приложения некоторых результатов теории статистических решений применительно к формальной постановке задачи (1.37)—(1.38) и исследовать сравнительную эффективность полученных критериев на различных задачах конструкторского проектирования.

Критерии теории статистических решений предполагают наличие оценок эффективности выбора элементов множества списка I:

$$\alpha_{ji}^{k}, \quad i = 0,1,\ldots,n_j, \quad j \in I \setminus I^{k-1}. \tag{1.48}$$

По аналогии с матричными алгоритмами размещения [26] для заданного варианта решения π' будем строить оценки (1.48) таким образом, чтобы они не превышали соответствующей нижней границы критерия:

$$\alpha_{ji}^{k} \le \min_{\pi \in \Pi'} F(\pi), \quad \pi \in \Pi' \Rightarrow \begin{cases} \pi_S = \pi'_S, & s \notin I \setminus I^{k-1} \\ \pi_S = i, & s = j \\ \pi_s \in I_{n_S}^{0}, & s \in I \setminus \left(j \cup I^{k-1}\right) \end{cases}, \tag{1.49}$$

где $I_{n_s}^{0} = \{0,1,\ldots,n_s\}$. В простейшем случае оценки могут быть равными [37]

$$\alpha_{ji}^{k} = \Phi(\pi''), \quad \pi''_s = \begin{cases} \pi'_s, & s \notin I \setminus I^{k-1} \\ i, & s = j \\ 0, & s \in I \setminus (I^{k-1} \cup j) \end{cases}. \tag{1.50}$$

Особенность интерпретации результатов теории статистических решений применительно к постановке (1.37)—(1.38) заключается в том, что оценки α_{ji}^{k} формируют не матрицу, а множество векторов различной длины. Поэтому критерии выбора должны быть соответствующим образом скорректированы. Согласно [26], [36] определим следующие критерии выбора:

- критерий взвешенного среднего [36]:

$$\tilde{V}_2 : j_k = \arg \min_{j \in I \setminus I^{k-1}} \frac{1}{n_j} \sum_{i=1}^{n_j} \alpha_{ji}^k ; \qquad (1.51)$$

- максиминный критерий (критерий Вальда) [36]:

$$\tilde{V}_3 : j_k = \arg \max_{j \in I \setminus I^{k-1}} \min_i \alpha_{ji}^k ; \qquad (1.52)$$

- критерий Джилмора [26]:

$$\tilde{V}_4 : j_k = \arg \min_{j \in I \setminus I^{k-1}} \max_i \alpha_{ji}^k ; \qquad (1.53)$$

- критерий пессимизма-оптимизма (критерий Гурвица) [36]:

$$\tilde{V}_5 : j_k = \arg \max_{j \in I \setminus I^{k-1}} \left\{ \beta \min_i \alpha_{ji}^k + (1-\beta) \max_i \alpha_{ji}^k \right\}, 0 \le \beta \le 1; \qquad (1.54)$$

- критерий Хиллера [26]:

$$\tilde{V}_6 : j_k = \arg \max_{j \in I \setminus I^{k-1}} r_j^k , \qquad (1.55)$$

где $r_j^k = \min_{i \ne i_0} \alpha_{ji}^k - \alpha_{ji_0}^k$, $i_0 = \arg \min_i \alpha_{ji}^K$. Выбор элемента из множества A_{j_k} может выполняться независимо, однако целесообразно для этой цели использовать построенные оценки $\alpha_{j_k i}^k \left(i \in I_{n_{j_k}}^0 \right)$ и выбирать, согласно [26], элемент $a_{i_0}^{j_k}$, где $i_0 = \arg \min_i \alpha_{j_k i}^k$.

Если положить $n_1 = n_2 = \ldots = n_m = n$, то вычислительная сложность последовательного алгоритма, основанного на критериях выбора $\tilde{V}_2 — \tilde{V}_6$, вычисленная в операциях построения оценок α_{ji}^k, равна $0\left(p^2 n\right)$, где $p = Card\, I$. В задачах размещения n (число посадочных мест) может быть слишком велико. В этом случае представляет интерес использование в критериях выбора модулей макродискретных моделей. Выбрав соответствующий размер макродискрета, можно ограничить n любым заданным числом. При этом α_{ji}^k будет представлять собой оценку эффективности размещения j-го модуля по некоторым областям коммутационного поля. Выбор посадочного места для модуля j_K может осуществляться с применением обычных микродискретных моделей в области с номером i_0.

Проведено экспериментальное исследование эффективности критериев выбора $\tilde{V}_1 - \tilde{V}_6$ в задачах размещения. Решалась задача квадратичного назначения [26] 34 модулей на 36 посадочных мест, расположенных в 4 ряда по 9 посадочных мест в ряду. В табл. 1.1 приведены результаты решения (значения критерия) 6 задач последовательным алгоритмом с критериями выбора модулей $\tilde{V}_1 - \tilde{V}_6$. Первая схема соединений представляет собой известный тест Штейнберга [26], схемы 2—6 сгенерированы случайно. Расстояние между позициями оценивалось в ортогональной метрике, в критерии $\tilde{V}_5$ полагали, что $\beta = 0,5$.

Результаты исследований показали, что наиболее эффективными являются максиминный критерий и критерий Хиллера, причем критерий минимального риска эффективнее максиминного, что согласуется с данными, приведенными в работе [37].

В табл. 1.2 приведены результаты решения 5 задач трассировки в канале с четырьмя модификациями интервального алгоритма [29] и последовательными алгоритмами, построенными на принципах максимина и минимального риска. Результаты позволяют сделать вывод об эффективности предложенных игровых схем выбора цепей, причем относительная эффективность алгоритма минимального риска увеличивается с увеличением размерности задачи. Критерий трассировки вычислялся по (1.34), ограничения учитывались с помощью штрафных функций.

В табл. 1.3 приведены результаты решения 5 задач распределения соединений схемы по 7 слоям. Сравнивались тривиальный алгоритм расслоения цепей в порядке нумерации [33], алгоритм расслоения цепей на основе формирования семейства внутренне устойчивых множеств (ВУМ) [33] и алгоритмы расслоения, основанные на принципах максимина и минимального риска.

ТАБЛИЦА 1.1. **Результаты сравнения эффективности критериев выбора $\tilde{V}_1 - \tilde{V}_6$ на задачах размещения**

Номер схемы	Критерий выбора					
	$\tilde{V}_1$	$\tilde{V}_2$	$\tilde{V}_3$	$\tilde{V}_4$	$\tilde{V}_5$	$\tilde{V}_6$
1	6962	8784	8959	5425	8828	5189
2	26394	30022	27584	24644	29677	23110
3	27641	32876	31624	27846	30071	26096
4	26813	27277	29698	25223	27277	24019
5	28313	32294	31438	26526	32464	26458
6	31814	36041	35137	29524	34691	26141

ТАБЛИЦА 1.2. **Результаты сравнения эффективности последовательных алгоритмов канальной трассировки**

Номер задачи	Количество цепей	Интервальные алгоритмы				$\tilde{V}_4$	$\tilde{V}_6$
		1	2	3	4		
1	4	38	12	14	38	12	14
2	5	36	56	58	36	14	14
3	9	185	101	187	181	79	79
4	25	579	751	579	781	411	345
5	25	604	776	604	806	400	320

ТАБЛИЦА 1.3. **Результаты сравнения эффективности последовательных алгоритмов расслоения**

Номер схемы	Количество цепей	Тривиальный алгоритм	Алгоритм формирования ВУМ	$\tilde{V}_4$	$\tilde{V}_6$
1	222	384,5	415,8	441,8	297,3
2	343	618,7	536,2	513,3	386,0
3	513	917,3	1135,8	899,3	736,5
4	187	353,8	327,5	343,8	236,9
5	255	397,0	402,6	376,3	317,3

ТАБЛИЦА 1.4. **Результаты сравнения эффективности последовательных алгоритмов разбиения**

Номер схемы	Количество модулей	Алгоритм из [32]	Алгоритм макс. конъюнкции — мин. дизъюнкции	$\tilde{V}_4$	$\tilde{V}_6$
1	37	60	46	52	30
2	112	96	68	73	52
3	87	83	62	85	45
4	25	44	37	35	17
5	57	76	58	72	

Результаты расчетов подтверждают сравнительную эффективность алгоритма минимального риска.

В табл. 1.4 приведены результаты решения 5 тестовых задач разбиения. Задача 1 представляет собой известный тест К.К. Морозова [32], задачи 2—5 сгенерированы случайным образом. Сравнивались последовательный алгоритм разбиения, предложенный в [32], алгоритм максимальной конъюнкции—минимальной дизъюнкции [26] и последовательные алгоритмы разбиения, построенные на принципах максимина и минимального риска. Критерий рассчитывался по формуле (1.19).

На основании результатов исследований, приведенных в табл. 1.1—1.4, можно сделать общий вывод о сравнительной эффективности алгоритма минимального риска. Рассмотренное обобщение данного критерия выбора на постановку задачи (1.37)—(1.38) позволяет построить соответствующие последовательные алгоритмы решения основных задач конструкторского проектирования.

1.3.4. Выбор множеств в схеме итерационного алгоритма

Количество одновременно выбираемых множеств $p = Card\, I$ (размерность подзадач) в схеме итерационного алгоритма (см. рис. 1.1) определяется радиусом r_t' окрестности в метрике (1.40). При $r_t' = 1$ или $r_t' = 2$ вычислительные ресурсы позволяют, как правило, организовать полный перебор всех вариантов выбора множеств, порождающий алгоритмы оптимизации соответственно единичными или парными переназначениями модулей, цепей, инвариантных объектов. Схемы данных алгоритмов могут быть записаны следующим образом:

Алгоритм 1.1 (алгоритм одиночных переназначений):

Шаг 1. Положить $\pi^* = \pi^0$.

Шаг 2. Положить $S = 0$.

Шаг 3. Положить $j = 0$.

Шаг 4. Положить $j = j + 1$.

Шаг 5. Положить $\pi^1 = D\left(\pi^*, R, j\right)$.

Шаг 6. Положить $S = S + 1$.

Шаг 7. Если $F\left(\pi^1\right) \geq F\left(\pi^*\right)$, то идти к шагу 10.

Шаг 8. Положить $\pi^* = \pi^1$.

Шаг 9. Положить $S = 0$.

Шаг 10. Если $S > m$, то останов.

Шаг 11. Если $j < m$, то идти к шагу 4.

Шаг 12. Идти к шагу 3.

Алгоритм 1.2 (алгоритм парных переназначений):

Шаг 1. Положить $\pi^* = \pi^0$.

Шаг 2. Положить $S = 0$.

Шаг 3. Положить $i_1 = 0$.

Шаг 4. Положить $i_1 = i_1 + 1$.

Шаг 5. Положить $i_2 = i_1$.

Шаг 6. Положить $i_2 = i_2 + 1$.

Шаг 7. Положить $\pi^1 = D\left(\pi^*, R, \{i_1, i_2\}\right)$.

Шаг 8. Положить $S = S + 1$.

Шаг 9. Если $F\left(\pi^1\right) \geq F\left(\pi^*\right)$, то идти к шагу 12.

Шаг 10. Положить $\pi^* = \pi^1$.

Шаг 11. $S = 0$.

Шаг 12. Если $S > \dfrac{m(m-1)}{2}$, то останов.

Шаг 13. Если $i_2 < m$, то идти к шагу 6.

Шаг 14. Если $i_1 < m - 1$, то идти к шагу 4.

Шаг 15. Идти к шагу 3.

Алгоритм 1.1 имеет, таким образом, линейную, а алгоритм 1.2 — квадратичную сложность. Можно ожидать, что алгоритмы одиночных переназначений будут иметь высокое быстродействие, но обладать при этом более сильными локальными свойствами. Их применение эффективно в задачах со слабыми ограничениями. Алгоритмы парных переназначений, проигрывая в быстродействии, являются более мощным средством оптимизации, так как исследуют окрестности большего радиуса. Они представляют гораздо более широкий класс алгоритмов, чем алгоритмы парных перестановок.

При $r_t' > 2$ перебрать все варианты выбора множеств практически невозможно. Рассмотрим схему алгоритма групповых переназначений, осуществляющего частичный перебор вариантов. Зададим для каждого элемента π систему N подмножеств мощности p $\left\{I^1(\pi),\ I^2(\pi),\ldots, I^N(\pi)\right\}$ номеров множеств $I^j(\pi) = \left\{i_1^j(\pi),\ i_2^j(\pi),\ldots,\ i_p^j(\pi)\right\}$, $j = \overline{1,N}$ и будем осуществлять перебор подмножеств в определенном порядке. Порядок перебора может быть либо задан заранее, либо изменяться в зависимости от хода процесса оптимизации. Схему алгоритма записываем следующим образом.

Алгоритм 1.3 (алгоритм групповых переназначений):

Шаг 1. Положить $\pi^* = \pi^0$.

Шаг 2. Положить $n = 0$.

Шаг 3. Положить $S = 0$.

Шаг 4. Задать систему $\left\{I^1(\pi^*),\ \ I^2(\pi^*),\ldots,I^N(\pi^*)\right\}$.

Шаг 5. Положить $n = n+1$.

Шаг 6. Положить $j_n = f(j_{n-1}, S)$.

Шаг 7. Положить $\pi^1 = D\left(\pi^*, R, I^{j_n}(\pi^*)\right)$.

Шаг 8. Положить $S = S+1$.

Шаг 9. Если $F(\pi^1) \geq F(\pi^*)$, то идти к шагу 12.

Шаг 10. Положить $\pi^* = \pi^1$.

Шаг 11. Идти к шагу 3.

Шаг 12. Если $S \geq N$, то останов.

Шаг 13. Идти к шагу 5.

Отметим, что алгоритмы 1.1 и 1.2 также могут быть описаны схемой алгоритма 1.3. В данном случае система $\left\{I^1(\pi),\ldots,I^N(\pi)\right\}$ включает все возможные варианты выбора множеств, порядок перебора подмножеств системы задан заранее, содержимое подмножеств не зависит от конкретного элемента π. В алгоритмах оптимизации размещения сканированием [30], например, содержимое подмножеств $I^j(\pi)$ зависит от текущего варианта размещения, сама же система подмножеств включает только часть из возможных вариантов выбора модулей. Порядок перебора подмножеств в алгоритмах сканирования задан заранее.

1.3.5. Усиление глобальных свойств алгоритмов

Рассматриваемые алгоритмы оптимизации, построенные по принципу (1.39), являются алгоритмами локального поиска, т. е. исследуют множество П в окрестности текущего наилучшего найденного решения π^{*t}. Алгоритмы прекращают работу, если в заданной окрестности π^{*t} не найдено лучших вариантов. Решение, найденное с помощью таких алгоритмов, является локально-оптимальным относительно исследовавшейся окрестности.

Рассмотрим два основных пути усиления глобальных свойств локальных алгоритмов — увеличение радиуса окрестности поиска и изменение решающего правила при оценке перебираемых вариантов решений. В последнем случае алгоритм получает возможность «подниматься вверх по склону», т. е. в качестве π^{*t}

выбирать не наилучший вариант. Изменение решающего правила позволяет также оказывать влияние на способ формирования окрестности.

Увеличение радиуса окрестности r_t' влечет за собой увеличение мощности подмножеств $I^j(\pi)$, $j = \overline{1,N}$ и, следовательно, размерности решаемых последовательным алгоритмом подзадач. Увеличение радиуса окрестности r_t'' увеличивает количество просматриваемых элементов множеств A_i, $i \in I^j(\pi)$, $j = \overline{1,N}$. Если время работы последовательного алгоритма возрастает нелинейно с увеличением этих параметров, то увеличение радиуса окрестности может привести к недопустимому с практической точки зрения увеличению времени работы итерационного алгоритма. Одним из возможных путей снижения временных затрат при работе итерационного алгоритма является использование для решения подзадач последовательного алгоритма меньшей вычислительной сложности и, как правило, более низкой точности. Такой подход может дать лучшие результаты, чем использование более точного последовательного алгоритма, но для решения подзадач меньшей размерности. Потеря точности последовательного алгоритма компенсируется увеличением радиуса окрестности.

В табл. 1.5 приведены результаты решения тестовой задачи размещения Штейнберга [26] одним из вариантов алгоритма групповых переназначений для нескольких начальных размещений. Для двух значений размерности решаемых подзадач использованы два варианта последовательного алгоритма: алгоритм размещения по связности (критерий выбора $\tilde{V}_1$) и алгоритм минимального риска (критерий выбора $\tilde{V}_6$). Указано значение критерия размещения (в ортогональной метрике) и времени решения задач. На основании данных табл. 1.5 можно сделать некоторые выводы. С увеличением радиуса окрестности качество решения улучшается. Применение для решения подзадач менее эффективного алгоритма последовательного размещения по связности позволяет при $r_t' = 10$ значительно сократить время решения по сравнению с использованием алгоритма минимального риска. При большом радиусе окрестности качество конечного решения практически не зависит от качества начального, начальное размещение может вообще отсутствовать. Таким образом, увеличение окрестности поиска существенно влияет на улучшение

ТАБЛИЦА 1.5. **Результаты решения тестовой задачи Штейнберга с использованием различных последовательных алгоритмов**

Начальное решение	Алгоритм решения подзадач			
	$\tilde{V}_6$		$\tilde{V}_1$	
	$r_t' = 4$	$r_t' = 10$	$r_t' = 4$	$r_t' = 10$
5451	5368 2 с	4775 297 с	5405 1 с	4971 20 с
12691	5588 2 с	4791 214 с	6090 1 с	4951 17 с
Без начального решения	5843 3 с	4856 192 с	5954 1 с	4911 35 с

решения даже при использовании последовательных алгоритмов невысокой точности в итерационно-последовательной схеме.

Глобальные свойства алгоритма могут быть усилены изменением решающего правила, внесением в него элемента случайности. Такой подход был предложен в [38], назван принципом «моделируемого отжига» и стал впоследствии популярен [31]. Если в детерминированном алгоритме оптимизации новое состояние принимается в качестве текущего наилучшего состояния при

$$\Delta F = F\left(\pi^*\right) - F\left(\pi^1\right) > 0 \qquad (1.56)$$

(см. п. 1.3.4), то в алгоритме, построенном на принципе «моделируемого отжига», состояние π^1 принимается при

$$\xi < e^{\frac{\Delta F}{T}}. \qquad (1.57)$$

Здесь ξ — случайное число, равномерно распределенное в интервале $(0,1)$, T — параметр. Данный принцип основан на физической аналогии с процессом отжига металла [38], параметр T моделирует текущую температуру металла. Показано, что при бесконечно медленном уменьшении температуры до нуля вероятность отыскания глобального экстремума равна единице.

Схема «моделируемого отжига» придает алгоритму глобальные свойства, позволяя «подниматься вверх по склону». Если решающее правило (1.56) позволяет принимать только состояния, уменьшающие критерий, то правило (1.57), принимая все улучшающие состояния, позволяет принимать в зависимости от

конкретных значений ξ, ΔF, T и некоторые состояния, увеличивающие критерий.

Многочисленные работы, посвященные использованию принципа «моделируемого отжига» и последовавшие за [38], не внесли качественных дополнений в теоретические результаты, полученные в [38]. Это отчасти можно объяснить неудачной формой записи решающего правила (1.57). Покажем, что правило (1.57) может быть записано в эквивалентной форме:

$$\Delta F + \eta > 0\,, \tag{1.58}$$

где η — случайная величина с определенным законом распределения. В форме (1.58) это решающее правило больше походит на (1.56), и его можно интерпретировать как наложение на приращение критерия случайной аддитивной помехи, подчиняющейся определенному закону распределения. Выражение (1.57) эквивалентно выражению

$$\Delta F - T \ln \xi > 0\,. \tag{1.59}$$

Таким образом, случайная помеха η получается из равномерно распределенной случайной величины ξ по правилу:

$$\eta = -T \ln \xi\,. \tag{1.60}$$

Докажем, что величина η распределена по экспоненциальному закону: $p(\eta_1 \le \eta \le \eta_2) = p(\xi_2 \le \xi \le \xi_1) = \xi_1 - \xi_2 = e^{-\frac{\eta_1}{T}} - e^{-\frac{\eta_2}{T}} = \int_{\eta_2}^{\eta_1} \frac{1}{T} e^{-\frac{1}{T}x} dx$. Следовательно, случайная величина η имеет плотность вероятности $f(x) = \frac{1}{T} e^{-\frac{1}{T}x}$, что соответствует экспоненциальному закону распределения с параметром $\lambda = \frac{1}{T}$. При этом математическое ожидание η равно $M[\eta] = \frac{1}{\lambda} = T$, а дисперсия $D[\eta] = \frac{1}{\lambda^2} = T^2$. Данное решающее правило в форме (1.60) было известно гораздо раньше в [39], чем в [38], с соответствующим обоснованием сходимости к глобальному экстремуму. Однако в [39] рассматривался не экспоненциальный, а нормальный закон распределения помехи.

Рассмотрим вопрос об эффективности экспоненциального закона распределения. Исследуем решения одних и тех же задач при экспоненциальном, нормальном и равномерном законах распределения помехи с одним и тем же правилом изменения параметра T. При экспоненциальном законе распределения получение величины η из равномерно распределенной на интервале $(0,1)$ случайной величины ξ описывается выражением (1.60), величина η изменяется в интервале $(0,+\infty)$. При равномерном законе распределения η с параметрами $M[\eta]=T$ и $D[\eta]=T^2$ величина η получается из ξ по правилу:

$$\eta \approx (3,464\xi - 0,732)T, \tag{1.61}$$

изменяется в интервале $(-0,732T; 2,732T)$, а решающее правило в форме, аналогичной (1.57), записывается так:

$$\xi > 0,268 - \frac{\Delta F}{T}. \tag{1.62}$$

Нормальная случайная величина с соответствующими параметрами распределения может быть промоделирована [36]:

$$\eta \approx \left(\sum_{i=1}^{12}\xi_i - 5\right)T, \tag{1.63}$$

изменяется в интервале $(-5T; 7T)$ с решающим правилом

$$\sum_{i=1}^{12}\xi_i > 5 - \frac{\Delta F}{T}. \tag{1.64}$$

Сравнение решающих правил (1.57), (1.62), (1.64) показывает, что данная форма записи гораздо менее удобна и наглядна, чем (1.60).

В табл. 1.6 приведены результаты решения 10 тестовых задач квадратичного назначения 34 модулей на 36 посадочных мест, расположенных в 4 ряда по 9 посадочных мест в ряду. Использовался вариант алгоритма парных переназначений. Приведены значения критерия в начальной (случайной) точке, а также значения критерия после оптимизации начального решения с использованием различных законов распределения помехи и без наложения помехи.

ТАБЛИЦА 1.6. **Результаты исследования эффективности различных законов распределения случайной помехи**

Номер схемы	Начальная точка	Экспоненциальный закон	Нормальный закон	Равномерный закон	Без помехи
1	33133	21105	20708	20741	23123
2	36910	22339	22731	22307	24010
3	33798	22769	22764	22132	22582
4	36126	23074	22418	22187	25415
5	36442	23085	21713	22643	24745
6	31830	19079	19238	19017	20715
7	34728	21238	20898	21325	23010
8	33518	20429	19688	19133	20278
9	39177	25114	24548	24912	26567
10	30427	18208	18185	18672	20251

На основании полученных результатов можно сделать следующие выводы.

1. Использование в алгоритме принципа «моделируемого отжига» позволяет повысить качество получаемых решений.

2. Использование экспоненциального закона распределения помехи не исчерпывает возможности подхода.

3. Для определенного класса задач использование нормального и равномерного законов дает в среднем лучшие результаты. Строго говоря, в данном случае схема решающего правила не является уже схемой «отжига», так как не описывает соответствующий физический процесс. Отличительной особенностью нормального и равномерного законов распределения помехи является то, что в этом случае появляется возможность не только принять «худшее» решение, как это показано в (1.57), но и отвергнуть «лучшее».

Форма записи решающего правила (1.58) позволяет предложить еще один оригинальный способ повышения глобальных свойств алгоритма. Формирование исследуемой алгоритмом оптимизации окрестности с помощью метрики (симметрики) можно рассматривать как самостоятельную оптимизационную задачу. Решение о включении элемента π в формируемую окрестность принимается на основании правила

$$\Delta\rho = r - \rho\left(\pi^*, \pi\right) \geq 0 . \qquad (1.65)$$

При наложении на метрику случайной аддитивной помехи получаем

$$\Delta\rho + \eta \geq 0 . \qquad (1.66)$$

ТАБЛИЦА 1.7. **Результаты решения тестовых задач размещения с наложением помехи на метрику исходного пространства**

Начальное решение	Оптимизация с наложением помехи на метрику	Оптимизация без наложения помехи на метрику
11176	5144	5443
10341	4978	5325
13256	5232	5628
9714	5117	5363
11336	5159	5418

Наложение помехи «размывает» окрестность. В метрике $\rho'(.)$ это означает «размывание» размеров подзадач, в метрике $\rho''(.)$ — возможность «больших перемещений» модулей, инвариантных объектов, цепей при соответствующей амплитуде помехи.

В качестве примера рассмотрим оптимизацию пяти случайных начальных решений тестовой задачи квадратичного назначения Штейнберга [26] вариантом алгоритма групповых переназначений. Было принято $r' = 4$ (выбирали подмножества из четырех модулей), $r'' = 1$ (в ортогональной метрике). При этом на метрику $\rho''(.)$ накладывалась случайная аддитивная экспоненциально распределенная помеха. Амплитуда помехи в процессе оптимизации постепенно уменьшалась. В табл. 1.7 приведены результаты решения задач рассмотренным способом, а также без наложения помехи на метрику $\rho''(.)$.

Анализ результатов позволяет сделать вывод о целесообразности предлагаемого подхода. Его логическое обоснование состоит в том, что чем «хуже» вариант решения с точки зрения критерия, тем дальше от своих оптимальных положений находятся модули, цепи, инвариантные объекты и тем большие «перемещения» им необходимо совершать. По мере уменьшения критерия амплитуда «перемещений» уменьшается.

Рассмотренные способы повышения глобальных свойств алгоритмов позволяют улучшить окончательное решение, однако требуют при этом некоторого увеличения времени счета.

ВЫВОДЫ К ГЛАВЕ 1

1. Выполнена общая формальная постановка для задач конструкторского проектирования микроэлектронных устройств (МЭУ). Рассмотрены преимущества данной постановки.

2. На основании предложенной общей формальной постановки осуществлены формальные постановки основных конструкторских задач.

3. Сформулирована общая схема итерационно-последовательного алгоритма решения оптимизационных задач конструкторского проектирования МЭУ.

4. Введен способ метризации комбинаторного пространства, учитывающий конкретное содержание задачи.

5. Приведены критерии выбора в схеме последовательного алгоритма. Предложена модификация критерия минимального риска, ориентированная на общую постановку задачи.

6. Предложены схемы выбора множеств в итерационном алгоритме оптимизации.

7. Проанализированы способы повышения глобальных свойств оптимизационных алгоритмов.

ГЛАВА 2

КОМБИНАТОРНЫЕ КОНФИГУРАЦИИ СО СПЕЦИАЛЬНЫМИ СВОЙСТВАМИ: ОПИСАНИЕ, ГЕНЕРАЦИЯ, ПЕРЕЧИСЛЕНИЕ, ОПТИМИЗАЦИЯ

Как следует из приведенных в главе 1 постановок задач, основные задачи конструкторского проектирования МЭУ имеют сложную комбинаторную природу. Подобными свойствами обладают и многие другие задачи проектирования, управления, транспортной маршрутизации, размещения геометрических объектов и т. д. [1—4].

Для моделирования задач, имеющих сложную комбинаторную природу, необходимо использование соответствующих классов комбинаторных множеств, адекватно описывающих области допустимых решений указанных задач. При построении математических моделей требуются определение и конструктивное описание комбинаторных множеств, обладающих необходимыми комбинаторными свойствами. Во многих случаях для этого нужно построить комбинаторные множества, выходящие за рамки известных классов комбинаторных множеств. Следовательно, необходимо создание конструктивных средств описания комбинаторных множеств с заданными свойствами и их классификация.

В главе изложен единый подход к формализации описаний, генерации и перечислению классов комбинаторных множеств, обладающих заданными свойствами, для математического моделирования и решения задач, имеющих сложную комбинаторную структуру.

2.1. ПРОБЛЕМЫ ОПИСАНИЯ, ГЕНЕРАЦИИ И ПЕРЕЧИСЛЕНИЯ КОМБИНАТОРНЫХ КОНФИГУРАЦИЙ

Многие научные и прикладные проблемы в разных областях человеческой деятельности имеют комбинаторную структуру. Традиционно для моделирования и решения этих задач используют-

ся математические модели, основанные на комбинаторных конфигурациях. В результате такого моделирования возникает возможность анализа и решения комбинаторных задач упаковки и раскроя, теории графов, планирования, проектирования, управления и т. д. [1—5].

Основой математических моделей комбинаторных задач являются комбинаторные множества. Мы рассматриваем комбинаторное множество (или комбинаторную структуру) как набор кортежей, построенных из конечного множества произвольных элементов (так называемых порождающих элементов) в соответствии с определенными правилами. Перестановки, сочетания, конечные двоичные последовательности являются примерами классических комбинаторных множеств.

Существуют два класса задач комбинаторной структуры.

Основа математических моделей комбинаторных задач, принадлежащих первому классу, формируется следующими классическими комбинаторными множествами: перестановками, размещениями, сочетаниями, булевыми последовательностями. Описания и характеристики классических комбинаторных множеств хорошо известны (см., например, [6, 7]). Задача коммивояжера, задача о ранце, некоторые проблемы упаковки, раскроя и теории кодирования относятся к данному классу комбинаторных задач.

Задачи, математические модели которых невозможно адекватно описать классическими комбинаторными множествами, относятся ко второму классу комбинаторных задач. К этому классу принадлежат некоторые задачи упаковки и раскроя, проблемы обслуживания, планирования и другие. Для построения адекватных математических моделей этих комбинаторных задач необходимо ввести новые типы комбинаторных множеств, обладающих специальными характеристиками.

Еще во второй половине XVII века Г. Лейбниц выдвинул идею создания «всеобъемлющей комбинаторной схемы», которая позволяет описать различные комбинаторные множества [8]. Идея была исследована С. Бержем [9], который предложил концепцию конфигурации, т. е. отображения специального типа. Другой подход основан на применении теории перечисления Г. Пойа [10], в рамках которой проводится математическое описание конфигураций и построение соответствующих производящих функций, результатом чего является общая комбинаторная схема. Общая комбинаторная схема позволяет с единых позиций описывать многие комбинаторные множества.

Являясь универсальным средством описания широкого класса комбинаторных множеств, общая комбинаторная схема дает эффективные решения только для простых классов комбинаторных множеств. Применение данной схемы для описания конфигураций, имеющих сложную комбинаторную структуру, приводит к громоздким построениям, которые неприменимы при математическом моделировании реальных комбинаторных задач [11, 12].

Современное состояние проблемы описания классов комбинаторных структур, их анализа и генерации приведено в [13—22]. Общий подход к анализу перечислительных задач комбинаторики, основанный на производящих функциях, содержится в [13]. Для разных классов комбинаторных структур указаны методы построения и анализа генерирующих функций. Много публикаций посвящено генерации комбинаторных множеств [14—18]. Генерацию в них понимают как построение всех комбинаторных структур определенного типа [16]. В основном рассматриваются вопросы создания достаточно простых объектов — перестановок, сочетаний, деревьев и двоичных последовательностей. Генерация более сложных комбинаторных объектов затруднена из-за отсутствия эффективных конструктивных средств.

Некоторые проблемы, озаглавленные «комбинаторная генерация», описаны в [18]. Основные вопросы, рассматриваемые в работе, связаны с алгоритмическими аспектами построения различных комбинаторных структур. Алгоритмический подход к перечислению классических комбинаторных конфигураций, таких как перестановки, сочетания, разбиения, коды Грея, деревья и т. д., приведен в [19].

Комбинаторная теория видов, предложенная А. Жоялем (A. Joyal) [20, 21], формирует общий взгляд на описание, построение и анализ различных дискретных структур. Теория видов использует производящие ряды для спецификации и анализа структур. Первичный вид структур может быть объединен посредством введенных комбинаторных операций. Результирующий производящий ряд построен из соответствующих порождающих рядов первичных видов. Инструменты комбинаторной теории видов могут быть эффективно использованы для перечисления различных комбинаторных множеств. Но они неэффективны при построении и перечислении некоторых классов комбинаторных множеств со специальными свойствами.

Работа [23] посвящена общему описанию и анализу комбинаторных свойств больших комбинаторных конфигураций. Пред-

ставленный подход основан на аналитических методах и, прежде всего, на методе производящих функций. Авторы выдвигают идею построения описаний сложных комбинаторных конфигураций на основе более простых (атомных) структур, обладающих известными свойствами. Целью приведенных результатов является решение перечислительных задач для различных комбинаторных структур — множеств, разбиений, деревьев и т. д.

Решение многих научных и прикладных задач, имеющих комбинаторную природу, относится к проблематике комбинаторной оптимизации. Задачам комбинаторной оптимизации и методам их решения посвящены многие работы отечественных и зарубежных авторов, среди которых [1, 3, 5, 24—29].

Интенсивные исследования комбинаторных множеств в течение последних десятилетий проводятся в научной школе, руководимой чл.-кор. НАН Украины, профессором Ю.Г. Стояном. Введенное им в 1980 г. в работе [30] понятие евклидова комбинаторного множества создало основу для разработки и развития математических моделей и методов комбинаторной оптимизации. Исследования евклидовых комбинаторных множеств проводятся на основе их отображения в евклидово пространство, при котором каждому элементу комбинаторного множества ставится во взаимно однозначное соответствие точка арифметического евклидова пространства [31]. При таком отображении, называемом погружением, комбинаторные множества приобретают многие свойства. Эти свойства, с одной стороны, дают возможность разрабатывать новые подходы к решению задач комбинаторной оптимизации. С другой стороны, использование свойств погруженных множеств позволяет повысить эффективность известных методов решения оптимизационных комбинаторных задач [2, 32].

Центральным понятием описываемого подхода к моделированию задач комбинаторной оптимизации является понятие евклидова комбинаторного множества как класса комбинаторных множеств, обладающих специальными свойствами. Примерами евклидовых комбинаторных множеств служат множества перестановок, размещений с повторениями и без повторений, упорядоченных разбиений и др. [30].

Погружение комбинаторных множеств в евклидово пространство позволяет привлечь к исследованию их свойств весь арсенал средств анализа, алгебры, геометрии. Построение и исследование выпуклых оболочек погруженных комбинаторных множеств, распределения их элементов по семействам различных гиперплоскостей и граням многогранников, симметрии множеств

составляет основу для решения широких классов задач оптимизации на евклидовых комбинаторных множествах. Результаты такого рода для различных классов евклидовых комбинаторных множеств приведены в работах [2, 32, 33—36]. Современному состоянию теории евклидовых комбинаторных множеств посвящена монография [37].

Развитию методов ветвей и границ для решения задач комбинаторной оптимизации способствовали исследования экстремальных свойств выпуклых функций на евклидовых комбинаторных множествах, начатые в работах Ю.Г. Стояна и С.В. Яковлева [38, 39]. В этих работах предложены оценки и достаточные условия минимума на множествах перестановок для выпуклых функций на выпуклых множествах, содержащих перестановочные многогранники. Полученные результаты положили начало исследованиям экстремальных свойств выпуклых и сильно выпуклых функций на классах евклидовых комбинаторных множеств и методов решения соответствующих задач комбинаторной оптимизации. Основные результаты исследований приведены в работах Ю.Г. Стояна, С.В. Яковлева, О.А. Емца и их учеников [32, 34, 35, 40—49].

В статье проф. С.В. Яковлева [50] сформулированы основы теории выпуклых продолжений функций, заданных на вершинах выпуклых многогранников. Методы построения выпуклых продолжений для функций, заданных в точках евклидовых комбинаторных множеств, позволяют переходить к эквивалентным задачам оптимизации выпуклых и сильно выпуклых функций на евклидовых комбинаторных множествах. Для решения эквивалентных задач могут быть использованы методы ветвей и границ с оценками минимума выпуклых функций на комбинаторных множествах.

Развитию теории выпуклых продолжений функций, заданных на евклидовых комбинаторных множествах, посвящены работы [40, 41, 51—56]. В этих работах предложены конструктивные подходы к построению выпуклых продолжений для классов функций, заданных на различных классах евклидовых комбинаторных множеств.

Многие оптимизационные задачи геометрического проектирования [2], к которым относятся задачи упаковки, раскроя (Packing&Cutting) и покрытия (Covering), имеют комбинаторную структуру.

Комбинаторная структура задач геометрического проектирования во многих случаях определяется тем, что в процессе их

решения необходимо упорядочить размещаемые объекты в заданной области. При этом порядок размещения оказывает определяющее влияние на значение критерия эффективности, а с каждым размещаемым объектом связан набор метрических характеристик и параметров размещения. Исходя из этого, области допустимых решений задач геометрического проектирования описываются комбинаторными множествами, относящимися к различным классам [57, 58].

Математические модели многих комбинаторных экстремальных задач геометрического проектирования могут быть построены на основе известных и достаточно исследованных комбинаторных множеств, таких как перестановки, размещения, сочетания и другие [1, 2, 32, 59]. В то же время структура некоторых задач геометрического проектирования достаточно сложна и не может быть адекватно описана на основе комбинаторных множеств, принадлежащих известным классам [60—63].

Для моделирования задач, имеющих сложную комбинаторную природу, необходимо использование соответствующих классов комбинаторных множеств, адекватно описывающих области допустимых решений указанных задач. При построении математических моделей требуется определение и конструктивное описание комбинаторных множеств, обладающих необходимыми комбинаторными свойствами. Во многих случаях для этого нужно построить комбинаторные множества, выходящие за рамки известных классов. Следовательно, необходимо создание конструктивных средств описания комбинаторных множеств с заданными свойствами и их классификации.

Построение комбинаторных множеств, обладающих специальными характеристиками, основано на решениях следующих трех классических комбинаторных задач:

1. Описание комбинаторного множества, имеющего специальные свойства.

2. Генерация элементов комбинаторного множества.

3. Перечисление элементов комбинаторного множества.

Первая проблема может быть реализована классическими подходами. Но их применение для описания множеств, имеющих сложную комбинаторную структуру, приводит к громоздким выражениям, которые неприменимы в реальных комбинаторных задачах. Основой решения проблемы генерации являются методы и алгоритмы генерации классических комбинаторных множеств. Эти методы и алгоритмы могут быть использованы при построении комбинаторных множеств, обладающих специальными

свойствами. Перечислительная комбинаторика является основой решения проблемы перечисления.

Как отмечает Р. Стэнли [12], «перечислительная комбинаторика связана с подсчетом числа элементов конечного множества *S*». Главное — оценка их количества, но не поиск какого-то определенного элемента. При этом поиск конкретного элемента комбинаторного множества является важной частью реальных комбинаторных задач [16].

Основная цель исследования — создание и развитие конструктивных средств формального описания, генерации и перечисления комбинаторных множеств, обладающих специальными свойствами, на основе базовых комбинаторных множеств. Описания не должны быть громоздкими. Результаты должны быть применимы в математическом моделировании различных классов комбинаторных задач.

2.2. БАЗОВЫЕ КОМБИНАТОРНЫЕ МНОЖЕСТВА. БАЗОВЫЕ ОТОБРАЖЕНИЯ

Для описания комбинаторных множеств используем понятие конфигурации, введенное К. Бержем [9].

***Определение* 1** [7]. Пусть $X = \{1, 2, ..., m\}$, $Y = \{y_1, y_2, ..., y_n\}$ и пусть определен линейный порядок на Y, такой что $y_1 < y_2 < < ... < y_n$. Отображение $\varphi : X \to Y$, удовлетворяющее некоторому множеству ограничений Λ, называется *конфигурацией*. Множество ограничений Λ, которому удовлетворяет отображение φ, определяет некоторый класс конфигураций, соответствующий условиям комбинаторных структур в заданной задаче.

Пусть $Z = \{z_1, z_2, ..., z_n\}$ — множество произвольных элементов. Построим комбинаторное множество *Y*, порожденное кортежем $z = (z_1, z_2, ..., z_n)$, где $y = (z_{j_1}, z_{j_2}, ..., z_{j_m}) \in Y(z)$, $m \le n$, $\{j_1, j_2, ..., j_n\} \subseteq J_n = \{1,2,...,n\}$. Y — образ при отображении $\Gamma_Y : Z \to Y$, $Y = \Gamma_Y(Z)$.

Множество Y назовем *базовым комбинаторным множеством*, а соответствующее отображение Γ_Y — базовым отображением.

Семейство базовых комбинаторных множеств может быть сформировано на основе известных конструктивных средств комбинаторики [11, 12]. Включим в него классы конфигураций,

соответствующие наиболее распространенным комбинаторным множествам.

Исходя из этого, рассмотрим конфигурации, соответствующие перестановкам, размещениям из n объектов по η, сочетаниям и n-кортежам. Отметим, что построение конфигураций, включая определение отображений φ и ограничений Λ, реализовано, например, в [7].

Таким образом, семейство базовых комбинаторных множеств состоит из множества перестановок $P_{n\eta}$ из n элементов, η из которых различны; общего множества размещений $\tilde{A}_n^{\eta}$ из n элементов по η; множества сочетаний $\bar{C}_n^{\eta}$ из n элементов по η; кортежа T_{nm} из n элементов, m из которых различны [32].

Будем рассматривать семейство базовых комбинаторных множеств как открытую систему, позволяющую ее расширение при построении новых комбинаторных множеств.

2.3. КОМПОЗИЦИОННЫЕ *k*-ОБРАЗЫ КОМБИНАТОРНЫХ МНОЖЕСТВ (*k*-МНОЖЕСТВА)

Определим операцию n-композиции комбинаторных множеств, используя операцию композиции, описанную в [11]. Пусть комбинаторное множество Y_0 порождено элементами z_1^0, $z_2^0, ..., z_n^0$, а комбинаторные множества $Y_1, Y_2, ..., Y_n$ порождены элементами $z^1, z^2, ..., z^n$,

$$z^i = \{z_1^i, z_2^i, ..., z_{n_i}^i\} \in Z, \ i \in J_n. \qquad (2.1)$$

Введем операцию n-замещения комбинаторных множеств Y_0, Y_1, Y_2, ..., Y_n, которая состоит в замене каждого порождающего элемента $z_1^0, z_2^0, ..., z_n^0$ множества Y_0 произвольным элементом множества $Y_1, Y_2, ..., Y_n$ соответственно. Операция n-замещения представляется отображением $F(Y_0(z^0), \Gamma_{Y_i}(z^i), i \in J_n)$.

В результате применения операции формируется элемент множества W_z. Множество W_z состоит из элементов вида $w = (z_1^1, z_2^1, ..., z_{n_1}^1, z_1^2, z_2^2, ..., z_{n_2}^2, ..., z_1^n, z_2^n, ..., z_{n_n}^n) \in W_z$, где $y = (z_{t_1}^0, z_{t_2}^0, ..., z_{t_n}^0) \in$

$\in Y_0$, $z^0_{t_j} \in Y_i(z^i)$, $t_j \in J_n$, $i \in J_n$, $j \in J_n$. Операция n-композиции множеств определяется как композиция отображений на основе базовых отображений и операции n-замещения следующим образом:

$$W_z = \Gamma_W \circ \Gamma_{Y_0}(x) = F(Y_0(z^0), \Gamma_{Y_i}(z^i), i \in J_n), \qquad (2.2)$$

где $\Gamma_{Y_0} : Z \to Y_0$ — базовое отображение; Y_0 — базовое комбинаторное множество; $\Gamma_W : Y_0 \times \{z^1, z^2, ..., z^n\} \to W$, $W = \{W_z,\ z \in Z\}$. Для любого Y_0 справедливо: $\Gamma_W(Y_0) = W_z \subset W$.

Осуществим операцию n-композиции комбинаторного множества W_z, заменяя его порождающие элементы элементами других базовых комбинаторных множеств. Выполним k таких шагов. В результате получим обобщение W_z вида (2.2).

Пусть $z^\beta = \{z^\beta_1, z^\beta_2, ..., z^\beta_{n_\beta}\} \subseteq Z_{\delta_i}$, Z_{δ_i} — множества произвольных элементов, $\beta \in \delta_i$, $i \in J^0_k = \{0,1,...,k\}$, где $\delta_0 = \{0\}$,

$$\delta_i = \{\beta_j, j = 1,2..., \eta_i\},\ \beta_j = (\alpha_1, \alpha_2, ..., \alpha_i),$$
$$\alpha_1 \in J_n,\ \alpha_2 \in J_{n_{\alpha_1}}, ..., \alpha_i \in J_{n_{\alpha_1 ... \alpha_{i-1}}}, \qquad (2.3)$$

$$\eta_1 = n,\ \eta_2 = \sum_{j=1}^{n} n_j,\ \eta_i = \sum_{\alpha_1=1}^{n} \sum_{\alpha_2=1}^{n_1} ... \sum_{\alpha_{i-1}=1}^{n_{1...i-2}} n_{\alpha_1 ... \alpha_{i-1}},\ i = 3,4,...,k. \qquad (2.4)$$

Пусть Γ_{δ_0}, Γ_{δ_i} являются отображениями вида

$$\Gamma_{\delta_0} : Z_{\delta_0} \to Y^0,\ \Gamma_{\delta_i} : Y^{i-1} \times Z_{\delta_i} \to Y^i, \qquad (2.5)$$

где $Y^0 = \{Y_{\delta_0}(z^\beta),\ \beta \in \delta_0\}$, $Y^i = \{Y_{\delta_i}(Y_{\delta_{i-1}},\ z^\beta),\ \beta \in \delta_i\}$, $Y_{\delta_0} = \Gamma_{\delta_0}(z^0)$, $Y_{\delta_i} = \Gamma_{\delta_i}(Y_{\delta_{i-1}}, z^{\delta_i}) = F(Y_{\delta_{i-1}}, \tilde{\Gamma}_{\delta_i}(z^{\delta_i}))$, $i \in J_k$, $J_t = \{1,2,...,t\}$, $F(Y_{\delta_{i-1}}, \tilde{\Gamma}_{\delta_i}(z^{\delta_i}))$ — отображение, реализующее операцию n-замещения, которая состоит в замене каждого порождающего элемента множества $Y_{\delta_{i-1}}$ элементами базовых комбинаторных множеств $Y_\beta = \tilde{\Gamma}_\beta(z^\beta)$, $\beta \in \delta_i$, соответственно, $\tilde{\Gamma}_{\delta_i}(z^{\delta_i}) = (\tilde{\Gamma}_\beta(z^\beta), \beta \in \delta_i)$, $z^{\delta_i} = (z^\beta, \beta \in \delta_i)$, $\tilde{\Gamma}_\beta(z^\beta)$, $\beta \in \delta_i$, базовые отображения. Тогда $(z^\beta_{l_1}, z^\beta_{l_2}, ..., z^\beta_{l_\beta}) \in Y_\beta$, $z^\beta_{l_t} \in Y_\sigma$, $l_t \in J_{l_\beta}$, $\beta \in \delta_{i-1}$, $\sigma \in \delta_i$, $i \in J_k$.

Обозначим $\Gamma_i^* = \{\Gamma_{\delta_i}\}$, $i \in J_k$, δ_i определено в (2.3).

***Определение* 2.** Комбинаторное множество W_z называется *композиционным k-образом комбинаторных множеств* Y_0, Y_1, Y_2, ..., Y_n, $Y_{11}, Y_{12}, ..., Y_{1n_1}$, ..., $Y_{\underbrace{1...1}_{k}}, ..., Y_{nn_1...n_{k-1}}$ (*k-множеством*), порожденным множествами z^{β_k}, $\beta_k \in \delta_k$, если

$$W_z = \Gamma_k \circ \Gamma_{k-1} \circ ... \circ \Gamma_0(z). \qquad (2.6)$$

Множество Y_0 называется множеством нулевого порядка, Y_{δ_i} — множества *i*-го порядка. Отображения $\Gamma_i \in \Gamma_i^*$, $i \in J_k$, построены с использованием операции *n*-композиции. Множество W_z вида (2.6) является базовым комбинаторным множеством при $k = 0$.

***Теорема* 1.** Отображения $\Gamma_i \in \Gamma_i^*$, $i \in J_k$, в (2.6) являются сюръективными.

Доказательство. Отображение $\Gamma_i = \Gamma_{\delta_i}$, которое определяет соответствие между множествами $Y_{\delta_{i-1}} \in Y^{i-1}$ и $Y_{\delta_i} \in Y^i$, $i \in J_k$, есть сюръективным, потому что Γ_i-образом множества $Y_{\delta_{i-1}}$ является все множество Y_{δ_i}. Это следует из способа построения множеств $Y_{\delta_{i-1}}$, Y_{δ_i}.

***Теорема* 2.** Множество

$$Card\ W_z = \sum_{\{\gamma_1,\gamma_2,...,\gamma_r\} \subset J_n} \alpha_{\gamma_1,\gamma_2,...,\gamma_r} \cdot \prod_{i=1}^{k} \prod_{\beta_i = (\alpha_1\alpha_2...\alpha_i)} Card\ Y_{\beta_i}, \qquad (2.7)$$

где β_i удовлетворяет (2.3), $\alpha_{\gamma_1,\gamma_2,...,\gamma_r}$ — количество различных элементов $y \in Y_0$, сгенерированных *n*-кортежем $z = \{z_1^0, z_2^0, ..., z_n^0\}$.

Доказательство теоремы следует из правила произведения и из способа построения множества W_z и может быть получено математической индукцией.

Пример 2.1. Рассмотрим *k-множество* P_2 (нулевого порядка); P_2, P_2 (первого порядка); A_3^2, T_3, T_2, T_2 (второго порядка), порожденные множествами $\{0,1,2\}$, $\{3,4,5\}$, $\{6,7\}$, $\{8,9\}$. Здесь $k = 2$,

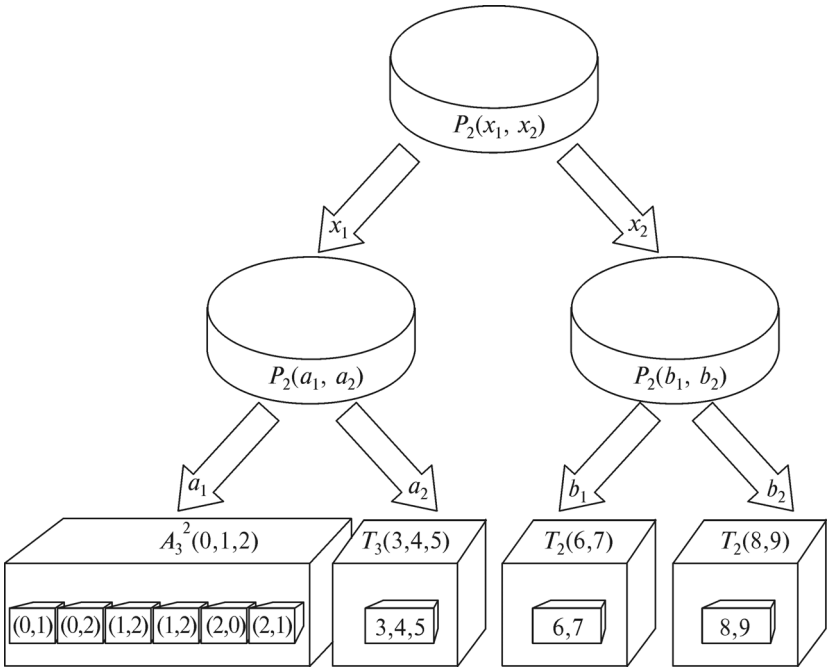

Рис. 2.1. Представление *k*-множества из примера 2.1 с помощью дерева

$P_n = P_{nn}$ — множество перестановок n различных элементов, $T_n = T_{nn}$ — n-кортеж, A_n^k — множество размещений без повторений.

Пусть $Z_{\delta_0} = \{x_1, x_2, ..., x_{10}\}$, $z^0 = \{x_1, x_2\} \subset Z_{\delta_0}$. Определим отображение вида $\Gamma_{\delta_0} : Z_{\delta_0} \to Y^0$, $Y^0 = \{Y_{\delta_0}\}$, $\delta_0 = \{0\}$, в результате получим множество нулевого порядка $Y_0 = \Gamma_{Y_0}(z^0)$. В соответствии с постановкой задачи

$$Y_0 = \Gamma_{Y_0}(x_1, x_2) = P_2(x_1, x_2), \; P_2(x_1, x_2) = \{(x_1, x_2), (x_2, x_1)\}. \quad (2.8)$$

Здесь $\Gamma_{Y_0} = \tilde{\Gamma}_0$ — базовое отображение, $P_2(x_1, x_2)$ — базовое комбинаторное множество перестановок, $Y_0 \in Y^0$.

Сформируем множества первого порядка $(i = 1)$. Пусть $Z_{\delta_1} = \{a_1, a_2, b_1, b_2, c_1, c_2, ..., z_1, z_2\}$, $z^1 = \{a_1, a_2\} \subset Z_{\delta_1}$, $z^2 = \{b_1, b_2\} \subset Z_{\delta_1}$,

где $\delta_1 = \{\beta_1\}$, $\beta_1 \in \{1,2\}$. Определим отображение $\Gamma_{\delta_1}: Y^0 \times Z_{\delta_1} \to$ $\to Y^1$ следующим образом:

$$\Gamma_{\delta_1}(Y_{\beta_0}, z^{\beta_1}) = F(Y_{\beta_0}, \tilde{\Gamma}_1(z^1), \tilde{\Gamma}_2(z^2)),\ Y_{\beta_1} = \Gamma_{\delta_1}(Y_{\beta_0}, z^{\beta_1}).$$

Здесь $Y_1(z^1) = \tilde{\Gamma}_1(z^1) = P_2(a_1, a_2) = \{(a_1, a_2), (a_2, a_1)\}$, $Y_2(z^2) = \tilde{\Gamma}_2(z^2) =$ $= P_2(b_1, b_2) = \{(b_1, b_2), (b_2, b_1)\}$ — базовые комбинаторные множества перестановок, $\tilde{\Gamma}_1$, $\tilde{\Gamma}_2$ — базовые отображения, отображение F реализует операцию n-замещения. В соответствии с операцией n-замещения при $n = 2$ каждый порождающий элемент x_i множества Y_0 вида (2.8) заменяется элементом базового множества Y_i, $i = 1,2$: x_1 элементом $y \in Y_1$, x_2 элементом $y \in Y_2$. В результате имеем

$$Y_{\beta_1} = F(Y_0(x_1, x_2),\ \tilde{\Gamma}_1(a_1, a_2), \tilde{\Gamma}_2(b_1, b_2)),$$

$$Y_{\beta_1} = \{(a_1, a_2, b_1, b_2), (a_2, a_1, b_1, b_2), (a_1, a_2, b_2, b_1), (a_2, a_1, b_2, b_1),$$
$$(b_1, b_2, a_1, a_2), (b_1, b_2, a_2, a_1), (b_2, b_1, a_1, a_2), (b_2, b_1, a_2, a_1)\},$$

$$Y_{\beta_1} \in Y^1 = \{Y_{\delta_1}(Y_{\delta_0}, z^{\beta_1}), \beta_1 \in \delta_1\}.$$

Аналогичным образом реализуется построение множеств второго порядка. В качестве Z_{δ_2} выберем $Z_{\delta_2} = \{1,2,...,10\}$, $\delta_2 = \{\beta_j,$ $j \in J_4\}$, $\beta_j \in \{(1,1),(1,2),(2,1),(2,2)\}$. Пусть $z^{11} = \{0,1,2\}$, $z^{12} = \{3,$ $4,5\}$, $z^{21} = \{6,7\}$, $z^{22} = \{8,9\}$. Определим комбинаторные множества второго порядка с помощью базовых отображений вида

$$Y_{11} = \tilde{\Gamma}_{11}(z^{11}) = A_3^2(z^{11}) = A_3^2(0,1,2) = \{(0,1),(0,2),(1,0),(2,0),(1,2),(2,1)\},$$

$$Y_{12} = \tilde{\Gamma}_{12}(z^{12}) = T_3(z^{12}) = T_3(3,4,5) = \{(3,4,5)\},$$

$$Y_{21} = \tilde{\Gamma}_{21}(z^{21}) = T_2(z^{21}) = T_2(6,7) = \{(6,7)\},$$

$$Y_{22} = \tilde{\Gamma}_{22}(z^{22}) = T_2(z^{22}) = T_2(8,9) = \{(8,9)\}.$$

Определим отображение Γ_{δ_2} следующим образом:

$$\Gamma_{\delta_2}(Y_{\delta_1},\ z^{\beta_2}) = F(Y_{\delta_1}(x_1, x_2), \tilde{\Gamma}_{11}(z^{11}),\ \tilde{\Gamma}_{12}(z^{12}), \tilde{\Gamma}_{21}(z^{21}), \tilde{\Gamma}_{22}(z^{22})),$$

$$Y_{\delta_2} = \Gamma_{\delta_2}(Y_{\delta_1}, z^{\beta_2}),\ \beta_2 \in \delta_2.$$

В результате получим композиционный k-образ комбинаторных множеств W_z. Он совпадает с комбинаторным множеством

Y_{δ_2} и его можно представить с помощью композиции отображений $Y_{\delta_2} = W_z = \Gamma_{\delta_2} \circ \Gamma_{\delta_1} \circ \Gamma_{\delta_0}(z^0)$ или в соответствии с обозначениями $\Gamma_0 = \Gamma_{\delta_0}$, $\Gamma_1 = \Gamma_{\delta_1}$, $\Gamma_2 = \Gamma_{\delta_2}$, $W_z = \Gamma_2 \circ \Gamma_1 \circ \Gamma_0(z^0)$:

$W_z = \{(0,1,3,4,5,6,7,8,9),(0,2,3,4,5,6,7,8,9),(1,2,3,4,5,6,7,8,9),$
$(1,0,3,4,5,6,7,8,9),\ (2,0,3,4,5,6,7,8,9),\ (2,1,3,4,5,6,7,8,9),$
$(3,4,5,0,1,6,7,8,9),\ (3,4,5,0,2,6,7,8,9),\ (3,4,5,1,2,6,7,8,9),$
$(3,4,5,1,0,6,7,8,9),\ (3,4,5,2,0,6,7,8,9),\ (3,4,5,2,1,6,7,8,9),$
$(0,1,3,4,5,8,9,6,7),\ (0,2,3,4,5,8,9,6,7),\ (1,2,3,4,5,8,9,6,7),$
$(1,0,3,4,5,8,9,6,7),\ (2,0,3,4,5,8,9,6,7),\ (2,1,3,4,5,8,9,6,7),$
$(3,4,5,0,1,8,9,6,7),\ (3,4,5,0,2,8,9,6,7),\ (3,4,5,1,2,8,9,6,7),$
$(3,4,5,1,0,8,9,6,7),\ (3,4,5,2,0,8,9,6,7),\ (3,4,5,2,1,8,9,6,7),$
$(6,7,8,9,0,1,3,4,5),\ (6,7,8,9,0,2,3,4,5),\ (6,7,8,9,1,2,3,4,5),$
$(6,7,8,9,1,0,3,4,5),\ (6,7,8,9,2,0,3,4,5),\ (6,7,8,9,2,1,3,4,5),$
$(6,7,8,9,3,4,5,0,1),\ (6,7,8,9,3,4,5,0,2),\ (6,7,8,9,3,4,5,1,2),$
$(6,7,8,9,3,4,5,1,0),\ (6,7,8,9,3,4,5,2,0),\ (6,7,8,9,3,4,5,2,1),$
$(8,9,6,7,0,1,3,4,5),\ (8,9,6,7,0,2,3,4,5),\ (8,9,6,7,1,2,3,4,5),$
$(8,9,6,7,1,0,3,4,5),\ (8,9,6,7,2,0,3,4,5),\ (8,9,6,7,2,1,3,4,5),$
$(8,9,6,7,3,4,5,0,1),\ (8,9,6,7,3,4,5,0,2),\ (8,9,6,7,3,4,5,1,2),$
$(8,9,6,7,3,4,5,1,0),\ (8,9,6,7,3,4,5,2,0),\ (8,9,6,7,3,4,5,2,1)\}$.

Согласно (2.7) число элементов множества W_z равно

$$Card\ W_z = \alpha_{12} \cdot Card\ P_2 \cdot Card\ P_2 \cdot Card\ A_3^2 \cdot Card\ T_3 \cdot Card\ T_2 \cdot Card\ T_2 = \\ = 2! \cdot 2! \cdot 2! \cdot 3! \cdot 1 \cdot 1 \cdot 1 = 48 .$$

2.4. ГЕНЕРАЦИЯ *k*-МНОЖЕСТВ

Семейство базовых комбинаторных множеств, сформированных выше, включает в себя множества с известными описаниями и комбинаторными свойствами. При этом предполагается, что мы имеем приемлемые алгоритмы генерации для базовых комбинаторных множеств. Для множеств перестановок, размещений, сочетаний и кортежей такие алгоритмы существуют и приведены, например, в [14—16, 18]. Использование известных алгоритмов генерации для композиционных *k*-образов комбина-

торных множеств (k-множеств) одновременно с алгоритмической реализацией отображений Γ_i из выражения (2.6) на основе операций n-композиции и n-замещения обусловливает большие возможности для решения задач генерации различных k-множеств. Примеры использования k-множеств при решении различных задач генерации перестановок, содержащих циклы, приведены в [22], перестановок с частично заданной сигнатурой — в [64]. Общая стратегия генерации k-множеств, алгоритмы генерации с оценками вычислительной сложности изложены в [65].

2.5. ПЕРЕЧИСЛЕНИЕ k-МНОЖЕСТВ НА ОСНОВЕ КОМБИНАТОРНЫХ ВИДОВ

Проанализируем связь между предложенной концепцией композиционных k-образов комбинаторных множеств (k-множеств) и теорией комбинаторных видов, изложенной в [20]. Используем основные элементы и терминологию теории, следуя [20, 21]. Построим комбинаторные виды, соответствующие базовым комбинаторным множествам и композиционным k-образам комбинаторных множеств. Результат построения является основой для решения задач перечисления k-множеств.

Пусть $U = (U_1, U_2, ..., U_n)$ — многосортное множество [20], где $U_i = z^i = \{z_1^i, z_2^i, ..., z_{n_i}^i\}$, $i \in J_n$. Определим многосортный вид [20] $S^0(U)$, где U — основное множество, и односортные виды [20] $S^i(U_i)$, основными множествами которых являются U_i, $i \in J_n$.

Пусть

$$\bar{S}^i(U) = S^i(U_i), \ i \in J_n, \tag{2.9}$$

где $\bar{S}^i(U)$ — многосортный вид, U — его основное множество.

Рассмотрим многосортный вид $SW_N(U)$, $N = \sum_{i=1}^{n} n_i$, построенный из $S^0(U)$ и $\bar{S}^i(U)$ типа (2.9) с помощью операции функториальной композиции [20, 21]:

$$SW_N(U) = S^0 \square(\bar{S}^1, \bar{S}^2, ..., \bar{S}^n)[U_1, U_2, ..., U_n] = \\ = S^0 \square(\bar{S}^1(U), \bar{S}^2(U), ..., \bar{S}^n(U)). \tag{2.10}$$

Вид $SW_N(U)$ типа (2.10) представляет собой вид композиционного k-образа комбинаторных множеств при $k = 1$.

Получим обобщение комбинаторного вида $SW_N(U) = SW_N^1(U)$ типа (2.10) для произвольного конечного натурального $k > 1$.

Рассмотрим виды $S^{11}, S^{12}, ..., S^{1n_1}; S^{21}, S^{22}, ..., S^{2n_2}; ...; S^{n1}, S^{n2}, ..., S^{nn_n}$. Основными множествами этих видов являются $U_{ij} = z^{ij} : U_{11} = \{z_1^{11}, z_2^{11}, ..., z_{n_{11}}^{11}\}$, $U_{12} = \{z_1^{12}, z_2^{12}, ..., z_{n_{12}}^{12}\}$, ..., $U_{nn_n} = \{z_1^{nn_n}, z_2^{nn_n}, ..., z_{n_{nn_n}}^{nn_n}\}$.

Аналогично видам $\bar{S}^i(U)$ определим комбинаторные виды $\bar{S}^{ij}(U_i)$ следующим образом: $\bar{S}^{ij}(U_i) = S^{ij}(U_{ij})$, $j \in J_{n_i}$, где $U_i = (U_{i1}, U_{i2}, ..., U_{in_i})$, $i \in J_n$. Используя операцию функториальной композиции, построим виды $SW_N^2(U)$: $SW_N^2 = SW_N^1 \square(\bar{S}^{11}, \bar{S}^{12}, ..., \bar{S}^{nn_n})[U_{11}, U_{12}, ..., U_{nn_n}]$.

$SW_N^2(U)$ являются видами комбинаторных множеств W_z типа (2.10) при $k = 2$.

Продолжая процесс построения комбинаторных видов для произвольного *k*, определим комбинаторные виды $SW_N^k(U)$ следующим образом:

$$SW_N^k(U) = SW_N^{k-1} \square(\bar{S}^{\underbrace{11...1}_{k}}, ..., \bar{S}^{nn_{\alpha_1} n_{\alpha_1\alpha_2} ... n_{\alpha_1...\alpha_{k-1}}})\left[U_{\underbrace{11...1}_{k}}, ..., U_{nn_{\alpha_1} n_{\alpha_1\alpha_2} ... n_{\alpha_1...\alpha_{k-1}}}\right], \quad (2.11)$$

где $\bar{S}^{\underbrace{11...1}_{k}}, ..., \bar{S}^{nn_{\alpha_1} n_{\alpha_1\alpha_2} ... n_{\alpha_1...\alpha_{k-1}}}$ — многосортные виды, которые определяем как $\bar{S}^{\alpha_1...\alpha_k}(U_{\alpha_1...\alpha_{k-1}}) = S^{\alpha_1...\alpha_k}(U_{\alpha_1...\alpha_k})$, $U_{\alpha_1...\alpha_{k-1}} = (U_{\alpha_1...\alpha_{k-11}}, U_{\alpha_1...\alpha_{k-12}}, ..., U_{\alpha_1...\alpha_{k-1} n_{\alpha_1...\alpha_{k-1}}})$. Здесь $S^{\alpha_1...\alpha_k}$ — заданные односортные виды, $U_{\alpha_1...\alpha_k} = z^{\alpha_1...\alpha_k}$ определяем из (2.3)—(2.5).

Таким образом, $SW_N^k(U)$ — вид комбинаторных множеств W_z типа (2.6) при произвольном натуральном k.

Построенные комбинаторные виды $SW_N^k(U)$ позволяют решать перечислительные задачи для различных классов композиционных *k*-образов комбинаторных множеств. С этой целью используем производящие ряды видов структур [20, 21] для базовых комбинаторных множеств и производящие ряды, связанные с результатами операций над базовыми комбинаторными множествами.

Для построения производящего ряда, связанного с комбинаторным видом $SW_N^k(U)$, используем следующие выражения. Производящие ряды, связанные с многосортными видами $S^0(U)$ и $\bar{S}^i(U) = S^i(U_i)$, $i \in J_n$, типа (2.9), можно записать так [20]:

$$F_{S^0}(x_1, x_2, ..., x_n) = \sum_{n_1, n_2, ..., n_n \geq 0} \left| F_{S^0}(n_1, n_2, ..., n_n) \right| \frac{x_1^{n_1} \cdot x_2^{n_2} \cdot ... \cdot x_n^{n_n}}{n_1 ! \cdot n_2 ! \cdot ... \cdot n_n !}, \quad (2.12)$$

$$F_{\bar{S}^i}(y_1^i, y_2^i, ..., y_n^i) = \sum_{n_1, n_2, ..., n_n \geq 0} \left| F_{\bar{S}^i}(n_1, n_2, ..., n_n) \right| \frac{(y_1^i)^{n_1} \cdot (y_2^i)^{n_2} \cdot ... \cdot (y_n^i)^{n_n}}{n_1 ! \cdot n_2 ! \cdot ... \cdot n_n !}, \quad (2.13)$$

где $\left| F_{S^0}(n_1, n_2, ..., n_n) \right|$ и $\left| F_{\bar{S}^i}(n_1, n_2, ..., n_n) \right|$ — количества $S^0(U)$- и $\bar{S}^i(U)$-структур, $n_i = |U_i|$, $i \in J_n$ соответственно.

Производящий ряд, связанный с комбинаторным видом $SW_N^1(U)$ как функториальной композицией n-сортных видов $S^0(U)$ и $\bar{S}^i(U)$, $i \in J_n$, определяем следующим образом [20]:

$$F_{SW_N^1}(x_1, x_2, ..., x_n) = \sum_{n_1, n_2, ..., n_n \geq 0} \left| F_{SW_N}(n_1, n_2, ..., n_n) \right| \frac{x_1^{n_1} \cdot x_2^{n_2} \cdot ... \cdot x_n^{n_n}}{n_1 ! \cdot n_2 ! \cdot ... \cdot n_n !}, \quad (2.14)$$

где $\left| F_{SW_N^1}(n_1, n_2, ..., n_n) \right| = \left| S^0 \square (\bar{S}^1, \bar{S}^2, ..., \bar{S}^n)[U_1, U_2, ..., U_n] \right| = \left| S^0(\bar{S}^1, \bar{S}^2, ..., \bar{S}^n) \right|$ — количество $SW_N^1(U)$-структур.

Количество $SW_N^1(U)$-структур зависит от комбинаторных видов $S^0(U)$, $\bar{S}^i(U)$ и мощности их основных множеств. Его можно вычислить для конкретных реализаций комбинаторного вида $SW_N^1(U)$. Построение производящего ряда, связанного с комбинаторным видом $SW_N^k(U)$ типа (2.11), состоит в рекуррентном применении выражений (2.12), (2.13) с учетом мощности основных множеств комбинаторных видов.

2.6. КЛАССИФИКАЦИЯ КОМПОЗИЦИОННЫХ *k*-ОБРАЗОВ КОМБИНАТОРНЫХ МНОЖЕСТВ

Классы базовых комбинаторных множеств являются основой классификации *k*-множеств. *k*-Множество, в котором базовые множества порядка *i* принадлежат одному классу (например, пе-

ТАБЛИЦА 2.1. **Классы *k*-множеств при *k* = 1**

Множества первого порядка	Множества нулевого порядка			
	Перестановки $P_{n\eta}$	**Размещения $\tilde{A}_n^{\eta}$**	**Сочетания $\bar{C}_n^{\eta}$**	***n*-кортежи T_{nm}**
Перестановки $P_{n\eta}$	Композиции перестановок PW_N	Размещения перестановок $AP(n,\eta,N,K)$	Сочетания перестановок $CP(n,\eta,N,K)$	*n*-кортеж перестановок $P_{\eta k}^{s}(G,H)$
Размещения $\tilde{A}_n^{\eta}$	Перестановки размещений $PA(n,\eta,N,K)$	Композиции размещений AW_N	Сочетания размещений $CA(n,\eta,N,K)$	*n*-кортеж размещений $A_{\eta n}^{ks}(G,H)$
Сочетания $\bar{C}_n^{\eta}$	Перестановки сочетаний $PC(n,\eta,N,K)$	Размещения сочетаний $AC(n,\eta,N,K)$	Композиции сочетаний CW_N	*n*-кортеж сочетаний $TC(n,\eta,N,K)$
n-кортежи T_{nm}	Перестановки *n*-кортежей $PT_{n\eta}^{m}$	Размещения *n*-кортежей $AT_{n\eta}^{m}$	Сочетания *n*-кортежей $CT_{n\eta}^{m}$	—
2-кортежи T_2	Парные перестановки $PI_{n\eta}^{m}$	Парные размещения $AI_{n\eta}^{m}$	Парные сочетания $CI_{n\eta}^{m}$	—

рестановкам), называется *однородным*. *k*-Множество, в котором базовые множества разных порядков принадлежат различным классам, называется *смешанным*.

Мы построили классификацию однородных *k*-множеств, порожденных различными базовыми комбинаторными множествами, при $k = 1$. Классы *k*-множеств определяются в зависимости от классов комбинаторных множеств нулевого и первого порядков. Результат классификации приведен в табл. 2.1.

2.7. НЕКОТОРЫЕ КЛАССЫ КОМПОЗИЦИОННЫХ *k*-ОБРАЗОВ КОМБИНАТОРНЫХ МНОЖЕСТВ И ИХ СВОЙСТВА

Рассмотрим примеры некоторых классов *k*-множеств, которые могут быть использованы для математического моделирования различных комбинаторных задач.

2.7.1. Множество перестановок кортежей

Пусть

$$T_i = \{(z_1^i, z_2^i, ..., z_m^i)\} \qquad (2.15)$$

— m-кортеж, составленный из элементов множества $U_i = \{z_1^i, z_2^i, ..., z_m^i\}$, $z_j^i \in R$, $i \in J_n$, $j \in J_m$. k Множеств среди n множеств T_i являются различными. Рассмотрим k-множество P_{nk}, $T_1, T_2, ..., T_n$, порожденное множествами $\{z_1^1, z_2^1, ..., z_m^1\}$, $\{z_1^2, z_2^2, ..., z_m^2\}$, ..., $\{z_1^n, z_2^n, ..., z_m^n\}$. Это k-множество обозначим PT_{nk}^m и назовем множеством перестановок кортежей, т. е. упорядоченных наборов вида $h \in PT_{nk}^m$, где $h = (z^{i_1}, z^{i_2}, ..., z^{i_n}) = (z_1^{i_1}, z_2^{i_1}, ..., z_m^{i_1}, z_1^{i_2}, z_2^{i_2}, ..., z_m^{i_2}, ..., z_1^{i_n}, z_2^{i_n}, ..., z_m^{i_n})$, $i_s, j_s \in J_n$, $i_s \neq j_s$, $s \in J_n$. Элементы PT_{nk}^m отличаются друг от друга последовательностью кортежей z^i в наборах.

Из способа построения множества PT_{nk}^m следует, что все его элементы принадлежат также множеству перестановок $P_{N k^0}$, порожденному множеством $\tilde{D} = \{z_1^1, z_2^1, ..., z_m^1, z_1^2, z_2^2, ..., z_m^2, ..., z_1^n, z_2^n, ..., z_m^n\}$. Здесь $N = mn$, $k^0 \geq k$, k^0 — количество различных элементов в $\tilde{D}$. Значит, справедливо соотношение $PT_{nk}^m \subset P_{N k^0}$.

Комбинаторный вид $SPT_n^m(U)$ перестановок кортежей при $n = k$ может быть построен в соответствии с подходом, изложенным в п. 2.5. Рассмотрим вид ST_m кортежей, построенных из элементов множеств вида (2.15). В соответствии с выражением (2.9) введем вид $\overline{ST}_m^i(U) = ST_m(U_i)$, $i \in J_n$, с основным множеством $U = (U_1, U_2, ..., U_n)$. Пусть $SP_n(U)$ — n-сортный вид перестановок, чьим основным множеством является U. Использовав формулу (2.10), получим вид $SPT_n^m(U)$: $SPT_n^m(U) = SP_n \square (\overline{ST}_m^1, \overline{ST}_m^2, ..., \overline{ST}_m^n)[U_1, U_2, ..., U_n]$.

Количество элементов множества перестановок кортежей можно вычислить согласно (2.14). Для этой цели используем порождающий ряд для перестановок и построим порождающий ряд для m-кортежей. Исходя из определения m-кортежа T_i как $T_i = \{(z_1^i, z_2^i, ..., z_m^i) \mid z_j^i \in U_i, j \in J_m\}$, $i \in J_n$, построим производящий ряд, связанный с видом $ST_m(U)$ как произведение m видов множеств [20]: $F_{ST_m}(x) = e^{mx} = 1 + mx + m^2 \frac{x^2}{2!} + ... + m^n \frac{x^n}{n!} + ...$

Производящий ряд для *n*-сортного вида перестановок $SP_n(U)$ [20] записываем следующим образом:

$$F_{SP_n}(x_1, x_2, ..., x_n) = \sum_{n_1, n_2, ..., n_n \geq 0} n! \cdot \frac{x_1^{n_1}}{n_1!} \cdot \frac{x_2^{n_2}}{n_2!} \cdot ... \cdot \frac{x_n^{n_n}}{n_n!}.$$

Согласно выражению (2.14) получаем производящий ряд, связанный с видом $SPT_n^m(U)$ перестановок *m*-кортежей T_i:

$$F_{SPT_n^m}(x_1, x_2, ..., x_n) = \sum_{m \geq 0} (m^{nm}) n! \cdot \left(\frac{x_1 \cdot x_2 \cdot ... \cdot x_n}{m!} \right)^m .$$

Этот производящий ряд соответствует ситуации, когда всевозможные *m*-кортежи, построенные из элементов множества $U_i = \{z_1^i, z_2^i, ..., z_m^i\}$, являются допустимыми. Если допустимым является только один заданный *m*-кортеж, составленный из элементов множества U_i, то соответствующий производящий ряд при $n = k$ приобретает вид

$$F_{SPT_n^m}(x_1, x_2, ..., x_n) = \sum_{m \geq 0} n! \cdot \left(\frac{x_1 \cdot x_2 \cdot ... \cdot x_n}{m!} \right)^m .$$

В этом случае $Card\ PT_{nk}^m = Card\ P_{nk} = n!$.

2.7.2. Композиция перестановок

Пусть P_{nt} — множество перестановок *n* элементов, *t* из которых различны. Композиционный *k*-образ комбинаторных множеств P_{nt}; $P_{m_1 k_1}$, $P_{m_2 k_2}$, ..., $P_{m_n k_n}$, порожденный множествами $U_1 = \{b_1^1, b_2^1, ..., b_{m_1}^1\}$, $U_2 = \{b_1^2, b_2^2, ..., b_{m_2}^2\}$, ..., $U_n = \{b_1^n, b_2^n, ..., b_{m_n}^n\}$, $U = (U_1, U_2, ..., U_n)$, называется композицией перестановок и обозначается $W_P = W_P(N, m_1, m_2, ..., m_n)$. W_P состоит из элементов $w = (\bar{w}_1, \bar{w}_2, ..., \bar{w}_n) \in W_P$, где $\bar{w}_i = (b_{s_1}^j, b_{s_2}^j, ..., b_{s_{m_j}}^j)$, $i \in J_{m_i}$, $j \in J_n$. В множестве $\{\bar{w}_1, \bar{w}_2, ..., \bar{w}_n\}$ *t* элементов различны, k_j элементов среди $b_{s_1}^j$, $b_{s_2}^j, ..., b_{s_{m_j}}^j$ различны. Последовательность $(s_1, s_2, ..., s_{m_j}) \in L_{m_j}$, где L_k — множество различных перестановок элементов индексного множества J_k.

Из способа построения множества W_P следует, что все его элементы являются также элементами множества P_{Nk^0}, порожденного множеством

$$G = \{a_1^1, a_2^1, ..., a_{m_1}^1, ..., a_1^n, a_2^n, ..., a_{m_n}^n\} = \{g_1, g_2, ..., g_N\}, \qquad (2.16)$$

содержащим k^0 различных элементов. Это значит, что справедливо включение $W_P \subset P_{Nk^0}$.

Комбинаторный вид композиции перестановок при $n = t$ обозначим $VW_P(U) = VW_P(U, N, n_1, n_2, ..., n_n)$. Построение $VW_P(U)$ и связанного с ним производящего ряда F_{VW_P} выполним на основе выражений (2.10) и (2.14). В результате получим: $VW_P(U) = SP_n \square(\overline{SP}_n^1, \overline{SP}_n^2, ..., \overline{SP}_n^n)[U_1, U_2, ..., U_n]$, где $\overline{SP}_n^i(U)$ — n-сортный вид перестановок, $\overline{SP}_n^i(U) = SP(U_i)$, $SP(U_i)$ — односортный вид перестановок, $i \in J_n$. Производящий ряд, связанный с видом $VW_P(U)$, записываем так:

$$F_{VW_P}(x_1, x_2, ..., x_n) = \sum_{m_1, m_2, ..., m_n \geq 0} m_1 ! \cdot m_2 ! \cdot ... \cdot m_n ! \cdot n ! \cdot \frac{x_1^{m_1} \cdot x_2^{m_2} \cdot ... \cdot x_n^{m_n}}{m_1 ! \cdot m_2 ! \cdot ... \cdot m_n !} =$$

$$= \sum_{m_1, m_2, ..., m_n \geq 0} n ! \cdot x_1^{m_1} \cdot x_2^{m_2} \cdot ... \cdot x_n^{m_n} \quad .$$

В соответствии с производящим рядом количество элементов множества W_P при $n = t$, $m_i = k_i$, $i \in J_n$, равно: $Card\, W_P = m_1 ! \cdot m_2 ! \cdot ... \cdot m_n ! \cdot n !$.

2.7.3. *k*-Композиция перестановок

Пусть k-множество

$$Y_0 = P_{nm}, \qquad (2.17)$$

$$Y_1 = P_{n_1 m_1}, \; Y_2 = P_{n_2 m_2}, ..., Y_n = P_{n_n m_n}, \qquad (2.18)$$

$$Y_{11} = P_{n_{11} m_{11}}, \; Y_{12} = P_{n_{12} m_{12}}, ..., Y_{1n_1} = P_{n_{1n_1} m_{1n_1}}, \; Y_{21} = P_{n_{21} m_{21}},$$
$$Y_{22} = P_{n_{22} m_{22}}, \; ..., \; Y_{2n_2} = P_{n_{2n_2} m_{2n_2}}, \; ..., Y_{nn_1} = P_{n_{n1} m_{n1}},$$
$$Y_{nn_2} = P_{n_{n2} m_{n2}}, ..., Y_{nn_n} = P_{n_{nn_n} m_{nn_n}}, \; ..., \qquad (2.19)$$

$$Y_{\underbrace{1\ldots1}_{k}} = P_{n_{\underbrace{1\ldots1}_{k}} m_{\underbrace{1\ldots1}_{k}}},\ldots,\ Y_{nn_1\ldots n_{k-1}} = P_{n_{nn_1\ldots n_{k-1}} m_{nn_1\ldots n_{k-1}}} \tag{2.20}$$

порождено множествами

$$B_1 = z^{k1} = \{a_1^1, a_2^1, \ldots, a_{n_{\underbrace{1\ldots1}_{k}}}^1\}, \ldots, B_\eta = z^{k\eta} = \{a_1^\eta, a_2^\eta, \ldots, a_{n_{nn_1\ldots n_{k-1}}}^\eta\}. \tag{2.21}$$

Здесь $a_i^j \in R^1$, $i \in J_{n_{\beta_k}}$; β_k, η определяется выражением (2.3). Данное k-множество называется *k-композицией перестановок* и обозначается W_P^k. Здесь P_{nm} вида (2.17) — множество нулевого порядка, множества (2.18)—(2.20) являются множествами первого, второго, ..., k-го порядка. W_P^k состоит из элементов $w = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_\eta) \in W_P^k$, где $\bar{w}_i = (a_{s_1}^j, a_{s_2}^j, \ldots, a_{s_{n_{\beta_k}}}^j)$, $i, j \in J_\eta$, $(s_1, s_2, \ldots, s_{n_{\beta_k}}) \in L_{n_{\beta_k}}$, L_k — множество всевозможных перестановок элементов индексного множества J_k. В соответствии с теоремой 2

$$V = Card\ W_P^k = Card\ P_{nm} \cdot \prod_{i=1}^{k} \prod_{\beta_i = (\alpha_1\alpha_2\ldots\alpha_i)} Card\ P_{n_{\beta_i} m_{\beta_i}}. \tag{2.22}$$

Если в выражении (2.22) $n = m$ и $n_{\beta_i} = m_{\beta_i}$, $i \in J_k$, то $P_{n_{\beta_i} m_{\beta_i}} = P_{n_{\beta_i}}$, $i \in J_k$, т. е. все множества $P_{n_{\beta_i} m_{\beta_i}}$ являются множествами перестановок без повторений. Следовательно,

$$V = Card\ W_P^k = n! \cdot \prod_{i=1}^{k} \prod_{\beta_i = (\alpha_1\alpha_2\ldots\alpha_i)} n_{\beta_i}!. \tag{2.23}$$

Из способа построения множества W_P^k следует, что все его элементы являются также элементами множества P_{Nm^0}, порожденного множеством $G = \{a_1^1, a_2^1, \ldots, a_{n_{\underbrace{1\ldots1}_{k}}}^1, \ldots, a_1^\eta, a_2^\eta, \ldots, a_{n_{nn_1\ldots n_{k-1}}}^\eta\} = \{g_1, g_2, \ldots, g_N\}$, т. е. $W_P^k \subset P_{Nm^0}$, где

$$N = \sum_{\alpha_1=1}^{n} \sum_{\alpha_2=1}^{n_1} \ldots \sum_{\alpha_k=1}^{n_{1\ldots k-1}} n_{\alpha_1\ldots\alpha_k}. \tag{2.24}$$

Комбинаторный вид $VW_P^k(U)$ k-композиции перестановок может быть построен с помощью рекурсивной процедуры, осно-

ванной на выражении (2.11). В результате получаем: $VW_P^k(U) = VW_P^{k-1} \square (\overline{SP}_N^1, \overline{SP}_N^2, ..., \overline{SP}_N^N)[U_1, U_2, ..., U_N]$, где N определяется соотношением (2.24), $\overline{SP}_N^i(U)$ — N-сортный вид, построенный аналогично виду $\bar{S}^{\alpha_1 ... \alpha_k}(U_{\alpha_1 ... \alpha_{k-1}})$ в (2.11), $i \in J_N$.

Построим производящий ряд, связанный с видом $VW_P^k(U)$, используя выражение (2.14):

$$F_{VW_P^k}(x_1, x_2, ..., x_N) = \left| F_{VW_P^k}(n_{\beta_1}, n_{\beta_2}, ..., n_{\beta_k}) \right| \prod_{n_{\alpha_1\alpha_2...\alpha_k}=n_{11...1}}^{n_{nn_1...n_{k-1}}} \frac{x_{\alpha_1\alpha_2...\alpha_k}^{n_{\alpha_1\alpha_2...\alpha_k}}}{n_{\alpha_1\alpha_2...\alpha_k}!},$$

где $\left| F_{VW_P^k}(n_{\beta_1}, n_{\beta_2}, ..., n_{\beta_k}) \right| = V$ определяется формулой (2.23). В результате

$$F_{VW_P^k}(x_1, x_2, ..., x_N) = \sum_{\alpha_1, \alpha_2, ..., \alpha_k \geq 0} n! \cdot \prod_{i=1}^{k} \prod_{\beta_i = (\alpha_1\alpha_2...\alpha_i)} n_{\beta_i}! \prod_{n_{\alpha_1\alpha_2...\alpha_k}=n_{11...1}}^{n_{nn_1...n_{k-1}}} \frac{x_{\alpha_1\alpha_2...\alpha_k}^{n_{\alpha_1\alpha_2...\alpha_k}}}{n_{\alpha_1\alpha_2...\alpha_k}!}.$$

2.8. ЗАДАЧИ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ НА *k*-МНОЖЕСТВАХ

Композиционные k-образы комбинаторных множеств (k-множества) и их подмножества могут выступать в качестве областей допустимых решений задач комбинаторной оптимизации.

Рассмотрим задачу комбинаторной оптимизации:

$$\kappa(\alpha) \to \min, \tag{2.25}$$

$$\alpha \in A \subseteq W_z, \tag{2.26}$$

где κ — функционал, заданный на подмножестве A композиционного образа комбинаторных множеств W_z вида (2.6).

Осуществим погружение множества W_z в евклидово пространство R^N с помощью отображения f вида [2]: $f : W_z \to R^N$, $\forall w = (w_1, w_2, ..., w_N) \in W_z$, $x = f(w) = (x_1, x_2, ..., x_N) \in E \subset R^N$, $x_i = w_i$, $i \in J_N$.

На этой основе сформулируем задачу комбинаторной оптимизации в евклидовом пространстве, эквивалентную задаче (2.25)—(2.26):

$$\varphi(x) \to \min, \tag{2.27}$$

$$x \in X \subseteq E_z \subset R^N, \tag{2.28}$$

где $x = f(\alpha)$, $\varphi(x) = \kappa(\alpha)$ $\forall \alpha \in W_z$, X, E_z — образы соответственно множеств A, W_z в пространстве R^N при отображении f.

Осуществим переход от задачи оптимизации (2.27)—(2.28) к эквивалентной задаче оптимизации с выпуклой (сильно выпуклой с параметром $\rho > 0$) на выпуклом замкнутом множестве $V \supset \operatorname{conv} E_z$ целевой функцией. Здесь эквивалентность двух задач оптимизации на множестве E_z будем понимать в том смысле, что функция цели исходной задачи и выпуклая функция цели эквивалентной задачи совпадают в точках множества E_z. С этой целью используем методы построения выпуклых и сильно выпуклых продолжений функций, заданных на комбинаторных множествах, предложенные в работах [38, 41, 50—53, 56].

2.9. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И РЕШЕНИЕ НЕКОТОРЫХ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ РАЗМЕЩЕНИЯ ПАРАЛЛЕЛЕПИПЕДОВ

Введенные композиционные *k*-образы комбинаторных множеств (*k*-множества) могут быть использованы в математических моделях и методах решения задач, имеющих комбинаторную структуру. Комбинаторные свойства *k*-множеств, заданные при их построении, позволяют более адекватно описывать соответствующие свойства комбинаторных задач. Это повышает эффективность применения известных методов комбинаторной оптимизации и дает возможность предлагать новые подходы к анализу математических моделей задач. Ниже приведены примеры задач геометрического проектирования, математические модели и методы анализа которых используют *k*-множества.

Задачи раскроя и размещения (Packing and Cutting problems [66]) элементов в некоторой области возникают во многих сферах человеческой деятельности. Примерами таких задач могут быть размещение контейнеров в трюме корабля, размещение товара на складах, формирование карт раскроя при подготовке полиграфической продукции или проектирование и размещение элементов на электронных платах [2]. Во многих работах такие задачи рассматриваются как однокритериальные оптимизационные задачи. Примерами критериев оптимизации могут служить:

заполнение области размещения, остаток материала, плотность размещения и другие. Однако, структура многих практических задач такова, что требует учета нескольких критериев одновременно. Так, в [67] рассматривается задача доставки грузов в различные пункты. Для решения задачи необходимо не только максимально заполнить грузовой отсек, а также, в зависимости от выбранного маршрута, обеспечить удобство выгрузки груза в пунктах разгрузки.

В данном параграфе выполнено математическое моделирование и решение многокритериальных задач размещения элементов, имеющих форму параллелепипеда, в области размещения, также имеющей форму параллелепипеда.

2.9.1. Общая постановка задачи

Рассмотрим задачу упаковки n-мерных параллелепипедов в параллелепипеде, которая является обобщением и развитием задач упаковки, исследованной в [68, 69], и состоит в следующем. Имеется N прямоугольных n-мерных параллелепипедов (n-параллелепипедов) $P_i \subset R^n$ вида

$$P_i = \{x \in R^n : x = (x_1, x_2, ..., x_n) \,|\, 0 \le x_k \le a_{ik}, k \in J_n = \{1, 2, ..., n\}\}, \quad (2.29)$$

где $a_i = (a_{i1}, a_{i2}, ..., a_{in})$ — вектор метрических характеристик (линейных размеров) n-параллелепипеда P_i, $i \in J_N$, $J_N = \{1, 2, ..., N\}$. Каждый n-параллелепипед P_i характеризуется векторами параметров размещения и числовых параметров $\theta_i \in R^s$, $i \in J_N$, среди которых могут быть координаты его центра тяжести, ценность, вес и т. д. Полагаем, что n-параллелепипеды P_i могут поворачиваться на углы $\pi / 2$ относительно координатных плоскостей пространства R^n. Задана область размещения в виде n-параллелепипеда D_0, который определяется как

$$D_0 = \{x \in R^n, x = (x_1, x_2, ..., x_n) \,|\, 0 \le x_k \le b_{0k}, k \in J_n\}, \quad (2.30)$$

где $b_{01} = d$ — переменная величина, $b_0 = (b_{01}, b_{02}, ..., b_{0n})$ — вектор метрических характеристик n-параллелепипеда D_0.

При этом грани n-параллелепипедов P_i, $i \in J_N$, параллельны соответствующим граням n-параллелепипеда D_0. Размещение n-

параллелепипедов P_i, $i \in J_N$, в области D_0 характеризуется векторным критерием $F = (F_1, F_2, ..., F_m)$.

Необходимо упаковать n-параллелепипеды P_i из множества $\{P_1, P_2, ..., P_N\}$ с учетом возможности их поворотов на углы $\pi / 2$ в n-параллелепипеде D_0 так, чтобы параллелепипеды попарно не пересекались, выполнялись заданные ограничения на значения параметров θ_i и векторный критерий F достигал экстремального значения.

2.9.2. Особенности решения общей задачи

Для решения задачи в приведенной постановке могут быть использованы различные комбинации методов многокритериальной оптимизации и схем перебора локальных экстремумов скалярных задач оптимизации. Сложность сформулированной задачи, большое количество вариантов размещения параллелепипедов не позволяют использовать точные методы оптимизации, ориентируя на применение приближенных и эвристических методов.

В частности, анализ многокритериальных задач можно проводить следующими способами [70, 71]:

1. Выделение парето-оптимальных решений — различных вариантов размещения параллелепипедов.

2. Формирование многофакторных скалярных оценок различных вариантов размещения n-параллелепипедов на основе свертки критериев с последующей оптимизацией скалярного обобщенного критерия.

3. Схема последовательно применяемых критериев.

Решение скалярных задач оптимизации локальных критериев или обобщенного критерия можно проводить на основе метода сужающихся окрестностей (МСО) [72].

Рассмотрим некоторые реализации общей задачи размещения параллелепипедов.

Задача 1. Имеются N прямоугольных n-параллелепипедов $P_i \subset R^n$ вида (2.29) и область размещения D_0 вида (2.30). Грани n-параллелепипедов P_i, $i \in J_N$, параллельны соответствующим граням n-параллелепипеда D_0. Для каждого параллелепипеда P_i известны координаты его центра тяжести в собственной системе

координат $m_{0i} = (m_1^{0i}, m_2^{0i}, ..., m_n^{0i}) \in P_i$, соответствующие его начальной ориентации, и масса $M_i, i \in J_N$. Полагаем, что:

1. Кроме поворотов параллелепипедов P_i на углы $\pi/2$ допустимы их зеркальные отражения относительно координатных плоскостей.

2. Линейные размеры области D_0 позволяют разместить все n-параллелепипеды P_i, $i \in J_N$. Для этого полагаем, что $b_{0k} \geq \geq \max_{i \in J_N, j \in J_n}$, $k = 2, 3, ..., n$.

3. Размещение n-параллелепипедов P_i, $i \in J_N$, в области D_0 характеризуется векторным критерием $F = (F_1, F_2)$, где F_1 — длина d ребра b_{01}, характеризующая занятую часть области D_0, F_2 — характеристика отклонения центра тяжести размещенных параллелепипедов от заданной точки области D_0.

Необходимо упаковать n-параллелепипеды P_i, $i \in J_N$, с учетом возможности их поворотов на угол $\pi/2$ и зеркальных отражений в n-параллелепипеде D_0 так, чтобы параллелепипеды попарно не пересекались и критерии F_1 и F_2 достигали минимального значения.

Задача 2. В условиях предыдущей задачи необходимо так упаковать n-параллелепипеды P_i, $i \in J_N$, чтобы центр тяжести размещенных параллелепипедов находился в некоторой окрестности заданной точки области D_0, описываемой системой линейных неравенств.

2.9.3. Построение математических моделей задач

Для построения математических моделей задач 1 и 2 используем математический аппарат Ф-функций [68, 73, 74].

С каждым n-параллелепипедом P_i, $i \in J_N$, свяжем ортогональную подвижную систему координат $Ox_1x_2...x_n$, а с областью D_0 — неподвижную систему координат $Ox_1x_2...x_n$. Начало собственной (подвижной) системы координат n-параллелепипеда P_i — точку O_i, находящуюся в геометрическом центре параллелепипеда, примем в качестве его полюса, $i \in J_N$. Координаты $u_i = (u_{i1}, u_{i2}, ..., u_{in})$ полюса n-параллелепипеда P_i, $i \in J_N$, относи-

тельно неподвижной системы координат $Ox_1x_2...x_n$ являются его параметрами размещения и определяют положение n-параллелепипеда в пространстве R^n. Тогда вектор $u = (u_1, u_2, ..., u_N) \in R^{Nn}$ определяет размещение параллелепипедов P_1, P_2, ..., P_N в пространстве R^n.

Для математического моделирования поворотов n-параллелепипедов P_i, $i \in J_N$, на угол $\pi/2$ относительно координатных плоскостей пространства R^n каждому n-параллелепипеду P_i поставим в соответствие вектор его линейных размеров $a_i = (a_{i1}, a_{i2}, ..., a_{in})$, $i \in J_N$. Различные повороты каждого n-параллелепипеда P_i на угол $\pi/2$ опишем с помощью упорядочения линейных размеров P_i. Перестановка элементов вектора $a_i = (a_{i1}, a_{i2}, ..., a_{in})$ вида

$$a_{it} = (a_{ik_{i1}}, a_{ik_{i2}}, ..., a_{ik_{in}}), \qquad (2.31)$$

где $v_i = (k_{i1}, k_{i2}, ..., k_{in})$ — перестановка элементов индексного множества J_n, $t \in J_M$, $M = n!$, соответствует всевозможным поворотам n-параллелепипеда P_i на угол $\pi/2$, $i \in J_N$. Перестановка вида (2.31) может быть получена из вектора $a_i = (a_{i1}, a_{i2}, ..., a_{in})$, соответствующего начальной ориентации n-параллелепипеда P_i, как $a_{it} = a_i \cdot Q_t$, где Q_t — перестановочная матрица. При этом поворот n-параллелепипеда P_i, задаваемый преобразованием Q_t, влечет за собой изменение его центра тяжести следующим образом: $m_i = m_{0i} \cdot Q_t$, $t \in J_M$, $M = n!$.

Центр тяжести n-параллелепипеда P_i — точка $m_i = (m_1^i, m_2^i, ..., m_n^i)$ — с учетом всевозможных поворотов на угол $\pi/2$ и зеркальных отражений находится в одной из 2^n вершин n-параллелепипеда $H_i \subseteq P_i$, у которого геометрический центр совпадает с геометрическим центром P_i, а одной из вершин является точка $m_{0i} = (m_1^{0i}, m_2^{0i}, ..., m_n^{0i}) \in P_i$, $i \in J_N$. Все вершины n-параллелепипеда $H_i \subseteq P_i$ могут быть получены по формуле

$$m_j^i = m_j^{0i} + 2\mu_j^i y_j^i, \ \mu_j^i = u_{ij} - m_j^{0i}, \qquad (2.32)$$

где $y_j^i \in \{0,1\}$, $j \in J_n$, $i \in J_N$.

Таким образом, положение n-параллелепипеда P_i определяется его параметрами размещения $u_i = (u_{i1}, u_{i2}, ..., u_{in})$, ориентацией, задаваемой поворотом на угол $\pi/2$ с помощью вектора $v_i = (k_{i1}, k_{i2}, ..., k_{in})$, и положением центра тяжести, который задается вектором булевых переменных $y_i = (y_{i1}, y_{i2}, ..., y_{in})$, $i \in J_N$. Тогда соответствующий ориентированный параллелепипед обозначим как $P_i(u_i, v_i, y_i)$, $i \in J_N$.

Математические модели задач 1 и 2 представим в следующем виде.

2.9.4. Модель задачи 1

В качестве характеристики отклонения центра тяжести размещенных параллелепипедов от заданной точки области D_0 — критерия F_2 — примем квадрат расстояния между центром тяжести размещенных параллелепипедов $X_c = (x_1^c, x_2^c, ..., x_n^c)$ и геометрическим центром C_d области размещения при фиксированной длине ребра b_{01}. Имеем

$$X_c = \sum_{i=1}^{N} \frac{M_i \cdot m_i(y_i)}{M},\ M = \sum_{i=1}^{N} M_i,\ C_d = (c_1^d, c_2^d, ..., c_n^d),$$

$$c_1^d = \frac{d}{2},\ c_j^d = \frac{b_{0j}}{2},\ j = 2,3,...,n,$$

$$F_2(X) = \sum_{i=1}^{n} (x_i^c(y) - c_i^d)^2, \qquad (2.33)$$

$$x_i^c(y) = \frac{1}{M} \sum_{j}^{N} M_j \cdot (m_i^{0j} + 2(U_{ji} - m_i^{0j}) \cdot y_i^j),$$

$$X^* = \arg\min_{X \in W \subset R^{3Nn+1}} (F_1(X), F_2(X)), \qquad (2.34)$$

где $F_1(X) = d$, $d = b_{01}$, $X = (u, v, y, d)$, $u = (u_1, u_2, ..., u_N)$, $v = (v_1, v_2, ..., v_N)$, $y = (y_1, y_2, ..., y_N)$, W — область допустимых решений, которая описывается системой неравенств [68]:

$$\begin{cases} \Phi_{ij}(u_i, u_j, v_i, v_j) \geq 0,\ i, j \in J_N,\ i < j, \\ \Phi_{0j}(u_0, u_j, v_j) \geq 0,\ j \in J_N. \end{cases} \qquad (2.35)$$

Для решения задачи (2.34) может быть использована схема последовательной оптимизации критериев. В этом случае необходимо решить последовательно две однокритериальные задачи:

$$X_1^* = \arg\min_{X \in W \subset R^{3Nn+1}} (F_1(X)), \tag{2.36}$$

$$X_2^* = \arg\min_{X \in W \subset R^{3Nn+1}} (F_2(X)). \tag{2.37}$$

Задача (2.36) заключается в упаковке параллелепипедов P_i в заданной области D_0 так, чтобы длина d ребра b_{01}, характеризующая занятую область, была минимальной. Для решения данной задачи может быть использован метод, предложенный в [68].

Задача (2.37) состоит в нахождении такой последовательности значений булевых переменных $y = (y_j, ..., y_{Nn})$, характеризующей размещение центров тяжести всех прямоугольников, которая бы минимизировала критерий $F_2(X)$.

2.9.5. Модель задачи 2

В терминах модели задачи 1 запишем математическую модель задачи 2:

$$X^* = \arg\min_{X \in W \subset R^{3Nn+1}} (F_1(X), F_2(X)), \tag{2.38}$$

$$X_1^* = \arg\min_{X \in W \subset R^{3Nn+1}} (F_1(X)), \tag{2.39}$$

$$X_2^* = \arg\min_{X \in W \subset R^{3Nn+1}} (F_2(X)), \tag{2.40}$$

$$L \cdot X \le h, \tag{2.41}$$

где $L = [L_{ij}]_{m \times (n \cdot N)}$, m — количество линейных ограничений, описывающих некоторую окрестность заданной точки области C_d, в пределах которой может находиться искомый центр тяжести размещенных параллелепипедов.

Для решения задачи (2.38) используем схему последовательного применения критериев. При этом решение скалярной задачи (2.39) выполним по схеме, предложенной в [68]. Для решения задачи (2.40) используем схему метода ветвей и границ в следующем виде.

2.9.6. Метод ветвей и границ

Решение задачи (2.40) заключается в нахождении такой последовательности значений булевых переменных $y = (y_j, ..., y_k)$ длины $k = n \cdot N$, с учетом системы линейных ограничений (2.41), которая бы минимизировала критерий $F_2(X)$. Отметим, что множество всех значений булевых переменных $y = (y_j, ..., y_k)$ представляет собой множество B_k k-мерных булевых векторов — вершин k-мерного гиперкуба, исследованного, например, в [41]. В [41] приведена схема метода ветвей и границ для решения задач оптимизации квадратичной функции на множестве B_k без дополнительных ограничений. Распространим указанный подход для решения задачи (2.40).

Опишем основные шаги предлагаемой модификации метода ветвей и границ для случая, когда $F_2(X)$ представляет собой квадратичную функцию.

Шаг 1. Используя известные методы [38,50], построим выпуклое (сильно выпуклое с параметром ρ) продолжение $\varphi(z)$ функции цели $F_2(X)$ на выпуклое множество $Z \supseteq \mathrm{conv} B_k$, где $\mathrm{conv} B_k$ — выпуклая оболочка множества B_k. В результате такого построения получим выпуклую или сильно выпуклую функцию $\varphi(z)$, которая в точках комбинаторного множества B_k принимает те же значения, что и исходная функция $F_2(X)$, т. е. $\varphi(z) = F_2(x)$ для $\forall z \in B_k$. Тогда $\varphi(z) = (Az, z) + Bz$, $A = [a_{ij}]_{k \times k}$, $a_{ij} \in R$, $b \in R^k$.

Перейдем к эквивалентной задаче:

$$\varphi(z) \to \min, \qquad (2.42)$$

$$L \cdot z \le h \; z \in B_k .$$

Шаг 2. Ветвление — выделение некоторого подмножества допустимых решений. Для этого зафиксируем $x_1 = 0$. Эффективность метода зависит во многом от выбора фиксируемой переменной, поэтому выбор такой переменной можно проводить, исходя из дополнительных соображений и особенностей задачи. Таким образом, преобразуем матрицы A и B, а также систему линейных неравенств:

$$\varphi_0^1(\tilde{z}) = (A_0^1 \tilde{z}, \tilde{z}) + B_0^1 \tilde{z} \;\; , A_0^1 = [a_{0ij}^1]_{k-1 \times k-1}, \; B_0^1 = [b_i]_{k-1}, \; L_0^1 \cdot \tilde{z} \le h_0^1,$$

$$\tilde{z} = (\tilde{z}_1, \tilde{z}_2, ..., \tilde{z}_{k-1}),\ \tilde{z}_i = z_{i+1},\ i = 1, ..., k-1.$$

Функция $\varphi_0^1(\tilde{z})$ также является квадратичной. Для оценки минимума полученной функции $\varphi_0^1(\tilde{x})$ на выделенном подмножестве воспользуемся нижними оценками минимума выпуклых и сильно выпуклых функций на евклидовых комбинаторных множествах, исследованных в [32, 39—42, 75, 76]:

$$\min \varphi(\tilde{z}) \geq \overline{e_1}(\tilde{z}) = \varphi(\tilde{z}) - (\nabla\varphi(\tilde{z}), \tilde{z}) + \min_{y \in P}(\nabla\varphi(\tilde{z}), y), \qquad (2.43)$$

$$\min \varphi(\tilde{z}) \geq \overline{e_2}(\rho) = \varphi(y^*) + \rho \cdot \min_{y \in P} \left\| y - y^* \right\|^2, \qquad (2.44)$$

$$\min \varphi(\tilde{z}) \geq \overline{e_3}(\tilde{z}, \rho) = \varphi(\tilde{z}) - \frac{1}{4\rho}\left\|\nabla\varphi(\tilde{z})\right\|^2 + \rho \min_{y \in P}\left\| y - \tilde{z} + \frac{1}{2\rho}\nabla\varphi(\tilde{z})\right\|^2, \qquad (2.45)$$

где $y^* = \arg\min_{y \in B_k} \varphi(y)$, $P = \{\tilde{z} \mid \tilde{z} \in B_k \subset R^n, L_0^1 \tilde{z} \leq h_0^1\}$.

Таким образом, вычислив одно из значений (2.42)—(2.44), определим оценку минимума Λ_0^1 функции $\varphi_0^1(\tilde{z})$.

Шаг 3. Аналогично шагу 2 зафиксируем $x_1 = 1$ и вычислим нижнюю оценку минимума Λ_1^1 функции $\varphi_1^1(\tilde{z})$.

Шаг 4. Сравним значения оценок в полученных на шагах 2 и 3 ветвлениях. Если значение $\Lambda_0^1 \leq \Lambda_1^1$, то в качестве новой исходной вершины выберем вершину с фиксированным значением $z_1 = 0$. Иначе новая исходная вершина — вершина с фиксированным значением $z_1 = 1$.

Шаг 5. В зависимости от выбора новой исходной вершины фиксируем значение z_1 и значение z_2. Аналогично шагам 2—4 построим оценки минимума и проведем новые ветвления при фиксированном значении $z_2 = 0$ и $z_2 = 1$.

Шаг 6. Ветвления продолжаются до тех пор, пока значения всех переменных не будут зафиксированы.

Шаг 7. В качестве начального решения выбираем вектор всех зафиксированных значений переменных.

Шаг 8. Поиск решения продолжения продолжается в тех вершинах, где значение оценки меньше или равно полученному начальному решению. Для таких вершин строим новые ветвления согласно описанному алгоритму. В результате выбираем наименьшее из всех найденных решений.

Т А Б Л И Ц А 2.2. **Исходные данные для решения задачи**

№	Размеры	№	Размеры	№	Размеры
2	[6, 1, 4]	9	[8, 9, 5]	16	[4, 3, 1]
3	[3, 4, 5]	10	[4, 7, 8]	17	[2, 8, 9]
4	[6, 4, 2]	11	[3, 3, 1]	18	[1, 1, 3]
5	[5, 2, 8]	12	[7, 7, 5]	19	[7, 8, 3]
6	[1, 2, 2]	13	[9, 1, 7]	20	[1, 2, 2]
7	[6, 4, 9]	14	[6, 3, 2]	21	[7, 1, 6]
8	[2, 9, 6]	15	[4, 6, 9]		

Отметим, что значения оценок (2.43)—(2.45) можно улучшить [50], решив дополнительные оптимизационные задачи вида

$$\overline{e_1}(\tilde{z}) \to \max\,,$$
$$\overline{e_2}(\rho) \to \max\,,\ \rho > \rho_0\,, \qquad (2.46)$$
$$\overline{e_3}(\tilde{z}, \rho) \to \max\,,\ \rho > \rho_0\,.$$

Рассмотрим пример решения задачи 2 [77]. Пусть заданы 20 параллелепипедов (табл. 2.2). Также задана область размещения (контейнер), в которую необходимо упаковать параллелепипеды (табл. 2.3).

Решение. На первом этапе разместим заданные параллелепипеды в область, используя метод, предложенный в [68]. В результате получим параметры размещения, приведенные в табл. 2.4. Первый столбец содержит номера параллелепипедов в порядке, определенном в результате решения задачи размещения на первом этапе. Зададим массы центра тяжести для каждого параллелепипеда.

Значение функции цели $F_1(X) = 12$. Полюс каждого параллелепипеда и центр тяжести задан в системах координат контейнера, в которых он размещен.

Т А Б Л И Ц А 2.3. **Область размещения**

№	Размеры
1	[15, 15, 15]

Пусть на координаты центров тяжести (2.32) наложены $t = 2$ ограничения вида

$$\sum_{j}^{N}\sum_{i}^{n} \alpha_{ti}^{\,j} m_i^j \le \beta_t\,,$$

где N — количество параллелепипедов; $N = 20$; n-размерность задачи $n = 3$. Коэффициенты α_t и β_t приведены в табл. 2.5.

ТАБЛИЦА 2.4. **Результаты размещения параллелепипедов и значения центров тяжестей**

№	Полюс	Размеры	Масса	Центр тяжести
12	[0, 0, 0]	[7, 7, 5]	36	[1,75; 1,75; 1,25]
19	[0, 0, 5]	[7, 8, 3]	75	[1,75; 2; 5,75]
18	[0, 0, 8]	[1, 1, 3]	77	[0,25; 0,25; 8,75]
15	[0, 8, 0]	[4, 6, 9]	22	[1; 9,5; 2,25]
21	[0, 1, 8]	[7, 1, 6]	24	[1,75; 1,25; 9,5]
8	[0, 2, 9]	[2, 9, 6]	20	[0,5; 4,25; 10,5]
4	[0, 11, 9]	[6, 4, 2]	4	[1,5; 12; 9,5]
9	[2, 2, 9]	[8, 9, 5]	17	[4; 4,25; 10,25]
20	[0, 11, 11]	[1, 2, 2]	55	[0,25; 11,5; 11,5]
7	[4, 8, 0]	[6, 4, 9]	51	[5,5; 9; 2,25]
3	[6, 11, 9]	[3, 4, 5]	47	[6,75; 12; 10,25]
2	[0, 0, 11]	[6, 1, 4]	63	[1,5; 0,25; 12]
5	[4, 12, 0]	[5, 2, 8]	29	[5,25; 12,5; 2]
14	[0, 11, 13]	[6, 3, 2]	49	[1,5; 11,75; 13,5]
16	[0, 2, 8]	[4, 3, 1]	4	[1; 2,75; 8,25]
13	[0, 14, 0]	[9, 1, 7]	7	[2,25; 14,25; 1,75]
11	[0, 5, 8]	[3, 3, 1]	78	[0,75; 5,75; 8,25]
6	[0, 13, 11]	[1, 2, 2]	42	[0,25; 13,5; 11,5]
10	[7, 0, 0]	[4, 7, 8]	15	[8; 1,75; 2]
17	[10, 7, 0]	[2, 8, 9]	86	[10,5; 9; 2,25]

ТАБЛИЦА 2.5. **Ограничения на центры тяжести параллелепипедов**

t_i	$\alpha_{t_i}^{j}$	β_t
$t = 1$	–2, –9, 4, 4, 7, 0, 0. 3, –4, –4, 9, –8, 5, 5, –2, 1, 1, –6, –3, –3, 10, 3, 3, 6, –1, –1, –8, –5, 8, 8, –9, 4, 4, –3, 0, 0, –7, 6, 6, 9, 2, 2, 5, 2, –9, –9, –6, 7, 7, 0, 3, 3, –4, –1, –1, –8, 5, 5, 8, 1	–7
$t = 2$	6, 6, 3, –10, –10, 7, –6, –6, 1, –2, –2, 5, –8, –8, 9, –4, –7, –7, 0, 7, 7, 4, –9, –9, –2, –5, –5, 2, –1, –1, 6, –7, –10, –10, –3, 4, 4, 1, 8, 8, 5, –8, –8, –1, –4, –4, 3, –9, 7, 7, –6, –9, –9, –2, 6, 6, 2, 9, 9, –3	7

Тогда на втором этапе решения выполним оптимизацию по критерию $F_2(X)$ (распределение весов), используя описанный метод ветвей и границ и оценку вида (2.42). Для сравнения результатов на данном этапе решим две задачи:

1. Распределение весов с решением задач (2.46).
2. Распределение весов без решения задач (2.46).

В табл. 2.6 приведены результаты решения задачи 1.

Решение задачи 2 оказалось хуже. Результаты решения приведены в табл. 2.7.

ТАБЛИЦА 2.6. **Результаты решения задачи распределения весов с улучшением оценок**

№	Масса	Центр тяжести
12	36	[1,75; 1,75; 1,25]
19	75	[1,75; 2; 5,75]
18	77	[0,25; 0,25; 8,75]
15	22	[1; 9,5; 2,25]
21	24	[1,75; 1,25; 9,5]
8	20	[0,5; 8,75; 10,5]
4	4	[4,5; 14; 10,5]
9	17	[8; 8,75; 12,75]
20	55	[0,75; 12,5; 12,5]
7	51	[8,5; 11; 6,75]
3	47	[8,25; 14; 12,75]
2	63	[4,5; 0,75; 12]
5	29	[7,75; 13,5; 6]
14	49	[4,5; 11,75; 14,5]
16	4	[3; 2,75; 8,25]
13	7	[6,75; 14,75; 5,25]
11	78	[2,25; 5,75; 8,25]
6	42	[0,75; 14,5; 11,5]
10	15	[8; 1,75; 2]
17	86	[10,5; 9; 2,25]
Значение функции цели	31,87	
Вектор булевых переменных	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0)	

ТАБЛИЦА 2.7. **Результаты решения задачи распределения весов без улучшения оценок**

Значение функции цели	37,25
Вектор булевых переменных	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0)

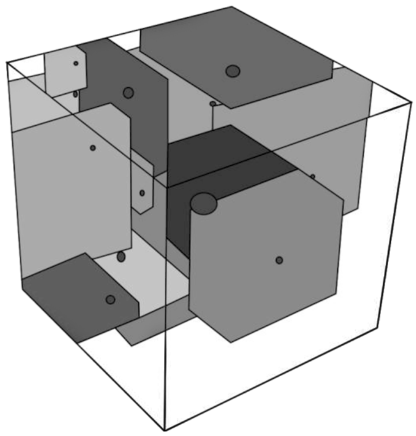

Рис. 2.2. Результаты решения задачи 2

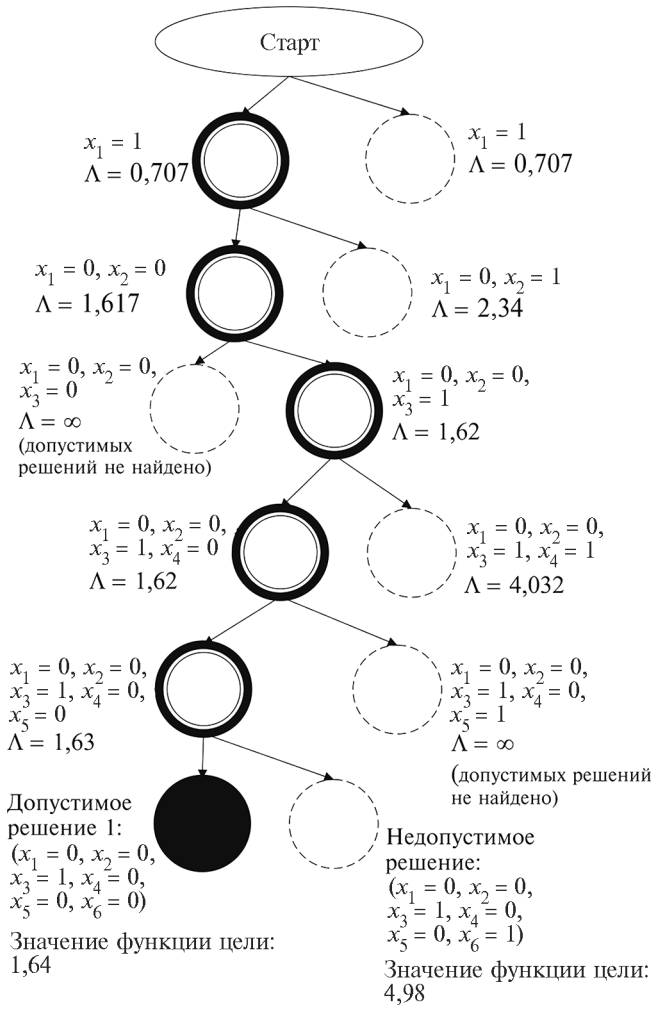

Рис. 2.3. Дерево решений

Графическое изображение полученного решения приведено на рис. 2.2. На рисунке точками отмечены полученные центры тяжести каждого из параллелепипедов и области размещения.

Для иллюстрации построения дерева решений предлагаемым методом рассмотрим пример. Пусть задана функция вида $\varphi(x) = (Ax, x) + Bx$, где

$$A = \begin{pmatrix} 1{,}343 & 0{,}199 & 0{,}071 & 0{,}571 & 0{,}069 & 0{,}227 \\ 0{,}199 & 1{,}797 & 0{,}173 & 0{,}226 & 0{,}056 & 0{,}218 \\ 0{,}071 & 0{,}173 & 1{,}199 & 0{,}192 & 0{,}406 & 0{,}471 \\ 0{,}571 & 0{,}226 & 0{,}192 & 1{,}392 & 0{,}187 & 0{,}544 \\ 0{,}069 & 0{,}056 & 0{,}406 & 0{,}187 & 1{,}908 & 0{,}83 \\ 0{,}227 & 0{,}218 & 0{,}471 & 0{,}544 & 0{,}83 & 1{,}957 \end{pmatrix}; \quad B = \begin{pmatrix} 0{,}736 \\ 0{,}962 \\ 1{,}066 \\ 1{,}559 \\ 1{,}95 \\ 3{,}323 \end{pmatrix}.$$

Пусть система линейных ограничений на булевые переменные x имеет вид

$$\begin{cases} 5x_1 - 10x_2 - 6x_3 - 1x_4 + 3x_5 + 8x_6 \le -6, \\ x_1 + 6x_2 - 10x_3 - 5x_4 - x_5 + 8x_6 \ge 5. \end{cases}$$

В качестве оценки выберем выражение (2.42). Используя предложенный алгоритм, построим дерево решений (рис. 2.3).

Таким образом, получено решение, которое совпадает с решением, найденным с помощью полного перебора. Для получения эффективных оценок решались дополнительные задачи оптимизации вида (2.46).

ВЫВОДЫ К ГЛАВЕ 2

1. Изложен единый подход к формализации описаний, генерации и перечислению классов комбинаторных множеств, обладающих заданными свойствами, для математического моделирования и решения задач, имеющих сложную комбинаторную структуру.

2. Введено понятие композиционного k-образа комбинаторных множеств (k-множества). Рассмотрены вопросы генерации k-множеств и перечисления k-множеств на основе теории комбинаторных видов. Выделены и исследованы классы k-множеств, которые могут быть использованы для математического моделирования различных комбинаторных задач.

3. Приведены примеры решения задач геометрического проектирования, математические модели и методы анализа которых используют k-множества.

4. Полученные результаты могут быть использованы при решении многих научных и прикладных задач, имеющих комбинаторные свойства. К ним относятся, в частности, задачи принятия решений и многофакторного оценивания объектов с дискретными параметрами [78, 79].

МОДЕЛИ И МЕТОДЫ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ В УПРАВЛЕНИИ РАСПРЕДЕЛЕННЫМИ ВЫЧИСЛЕНИЯМИ

Основная задача управления распределенными вычислительными средами — планирование выполнения заданий в условиях большого количества ресурсов, заданий и пользователей. При этом для выполнения заданий могут использоваться распределенные вычислительные системы (РВС), в которых необходимые для проведения вычислений ресурсы могут полностью (или частично) отчуждаться. Исходными данными для решения этой задачи являются выполняемые системой функции, которые могут быть формализованы в виде множества математических моделей. Для каждого из заданий могут быть заданы возможные варианты его решения, и каждое задание может состоять из нескольких подзадач. Связи между заданиями и этапами задают в виде графа, ребра которого характеризуют отношения следования (операционные и информационные), существующие между решаемыми заданиями и их подзадачами, и соответствуют направлениям информационных потоков. Кроме того, задают множество ресурсов системы и связей между ними и виды технических средств, которые используются в системе. Математические модели планирования вычислений должны отражать приведенные аспекты, влияющие на планирование как с целью повышения производительности непосредственно РВС, так и обеспечения качества обслуживания пользователей.

Временные ограничения для выполнения различных заданий требуют анализа работы различных ресурсов системы.

В качестве критериев оптимизации используют следующие частные критерии:

1. Минимизация затрат на выполнение заданий:

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} W_{ij} x_{ij}, \qquad (3.1)$$

где $i = (\overline{1, I})$ — множество заданий, реализуемых в системе; $j = (\overline{1, J})$ — множество обслуживающих ресурсов системы; W_{ij} — затраты на реализацию i-го задания в j-м ресурсе; $x_{ij} =$

$= \begin{cases} 1, & \text{если задание } i \text{ выполняется в } j\text{-м ресурсе,} \\ 0 & \text{в противном случае.} \end{cases}$

2. Минимизация общего времени выполнения всех заданий в системе:

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} t_{ij} x_{ij}, \tag{3.2}$$

где t_{ij} — время выполнения i-го задания j-м ресурсом.

3. Минимизация максимального времени выполнения задания в системе:

$$\min \left\{ \max_{j} \sum_{i=1}^{I} t_{ij} x_{ij} \right\}. \tag{3.3}$$

При формулировании *частных задач оптимизации* используются такие ограничения:

1. По связи между заданиями, т. е. задан граф G_F.
2. По связи между узлами, т. е. задан граф $G_м$.
3. Общие затраты на реализацию задания в системе:

$$\sum_{i=1}^{I} \sum_{j=1}^{J} W_{ij} x_{ij} \le W_{\text{доп}}. \tag{3.4}$$

4. На затраты на выполнение задания на ресурсах:

$$\sum_{i=1}^{I} W_{ij} x_{ij} \le W_{j\,\text{доп}}, \quad j = (\overline{1, J}). \tag{3.5}$$

5. На загрузку каждого из ресурсов системы:

$$\sum_{i=1}^{I} \lambda_i t_{ij} x_{ij} \le \rho_j, \quad j = (\overline{1, J}), \tag{3.6}$$

где λ_i — интенсивность поступления i-го задания на решение.

6. На общее время выполнения заданий:

$$\sum_{i=1}^{I} \sum_{j=1}^{J} t_{ij} x_{ij} \le T. \tag{3.7}$$

7. На время выполнения задания:

$$\sum_{j=1}^{J} t_{ij} x_{ij} \le \tau_i, \quad i = \overline{(1, I)}. \tag{3.8}$$

Рассмотрим основные типы частных задач оптимизации.

Первая частная задача оптимизации процесса распределения заданий

Необходимо распределить задания i, $i = \overline{(1, I)}$ между j ресурсами $j = \overline{(1, J)}$, чтобы обеспечить минимум общих затрат или минимум общего времени выполнения при выполнении ограничений на загрузку каждого из ресурсов (3.6) или затраты в каждом j-м ресурсе.

Математическую модель этой задачи можно записать так:

найти

$$\min \sum_{i=1}^{I} \alpha_{ij} x_{ij} \tag{3.9}$$

при ограничениях

$$\sum_{i=1}^{I} \beta_{ij} x_{ij} \le b_j, \quad j = \overline{(1, J)}; \tag{3.10}$$

$$\sum_{j=1}^{J} x_{ij} = 1, \quad i = \overline{(1, I)}, \quad x_{ij} \in \{0, 1\}, \tag{3.11}$$

где α_{ij} — затраты (или время выполнения) i-го задания в j-м ресурсе; b_j — допустимые затраты (загрузка) в j-м ресурсе;

$$x_{ij} = \begin{cases} 1, \text{ если задание } i \text{ выполняется в } j\text{-м ресурсе;} \\ 0 \text{ в противном случае.} \end{cases}$$

Ограничение (3.11) означает, что каждое задание выполняется только одним ресурсом.

Вторая частная задача оптимизации процесса распределения заданий

Необходимо так распределить задания i, $i = \overline{(1, I)}$, между j ресурсами, $j = \overline{(1, J)}$, чтобы обеспечить минимум времени общих затрат или минимум общего времени выполнения при выполнении ограничений на общее время выполнения или общие затраты.

Математическую модель этой задачи можно записать следующим образом.

Найти

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_{ij} x_{ij} \tag{3.12}$$

при ограничениях

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \beta_{ij} x_{ij} \le B, \tag{3.13}$$

$$\sum_{j=1}^{J} x_{ij} = 1, \quad i = (\overline{1, I}), \quad x_{ij} \in \{0, 1\}, \tag{3.14}$$

где α_{ij} — затраты (или время выполнения) i-го задания в j-м ресурсе; β_{ij} — время выполнения (или затраты) i-го задания в j-м ресурсе; B — общее время выполнения (или затраты) всех заданий.

Третья частная задача оптимизации процесса распределения заданий

Необходимо так распределить задания i, $i = (\overline{1, I})$, между j узлами, $j = (\overline{1, J})$, чтобы обеспечить минимум времени общих затрат или минимум общего времени выполнения при выполнении ограничений на общее время выполнения и загрузку ресурсов либо на общие затраты и загрузку ресурсов.

Математическая модель этой задачи имеет такой вид.

Найти

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_{ij} x_{ij} \tag{3.15}$$

при ограничениях

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} \beta_{ij} x_{ij} \le B, \tag{3.16}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{J} C_{ij} x_{ij} \le C_j, \quad j = (\overline{1, J}), \tag{3.17}$$

$$\sum_{j=1}^{J} x_{ij} = 1, \quad i = (\overline{1, I}), \quad x_{ij} \in \{0, 1\}. \tag{3.18}$$

Рассмотренные задачи являются основными при оптимизации процессов распределения заданий в РВС.

Далее рассмотрены математические постановки и методы решения задач планирования в РВС, сводящиеся к задачам дискретной оптимизации в виде линейного и нелинейного программирования.

3.1. МЕТОД РЕШЕНИЯ ПРОИЗВОЛЬНЫХ ЗАДАЧ БУЛЕВОГО ПРОГРАММИРОВАНИЯ: ОБЩИЙ ПОДХОД

Задачи линейного и нелинейного программирования с булевыми переменными являются моделями широкого круга прикладных задач в теории построения сложных систем и относятся к классу *NP*-полных трудно решаемых задач [1—9]. При этом эффективные методы решения задач нелинейного булевого программирования с произвольными нелинейностями практически отсутствуют. В настоящее время распространены подходы, в которых для каждой из задач данного типа разрабатывается свой метод решения. В этой связи представляется актуальным использовать единый подход к решению задач данного класса, обеспечивающий их решение с требуемой оперативностью (малой временной сложностью) и точностью. Далее рассматривается ранговый подход, позволяющий решать задачи линейного программирования и нелинейного программирования с булевыми переменными с произвольными нелинейностями одним и тем же алгоритмом [10—19].

Постановка задачи

Для описания всего множества аддитивных целочисленных функций с произвольными нелинейностями, определяемыми на множестве переменных $\{X_1, X_2 \ldots X_n\}$, введем понятие мета-функции $F(x)$:

$$F(x) = \sum_{j=1}^{p_1} C_{1j} S_1(C_n^1) + \sum_{j=1}^{p_2} C_{2j} S_2(C_n^2) + \ldots$$

$$\ldots + \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k) + \ldots + \sum_{j=1}^{p_n} C_{nj} S_n(C_n^n), \qquad (3.19)$$

где $Sr(C_n^r) = S_1 + S_2 + \ldots + S_{p_r}$ — сумма всех возможных сочетаний произведений переменных, содержащих в каждом произ-

ведении $Sr = X_p X_k ... X_m$ (нелинейности) r различных переменных; $p_r = \dfrac{n!}{r!(n-r)!}$; C_{rj} — целочисленные коэффициенты, стоящие в произведениях S_r, содержащих r переменных.

Обозначим H — множество всех функций, которое можно породить на основе $F(x)$, полагая равными нулю различные сочетания C_{lj} в (3.19). Множество H является полным в том смысле, что содержит в себе все возможные нелинейности, включающие все возможные сочетания переменных, их образующих, и которые можно построить на основе данного подмножества переменных $\{X_1, X_2 ... X_n\}$.

Нетрудно показать, что мощность данного множества велика, но конечна и равна 2^{p_Σ}, где

$$p_\Sigma = 1 + \frac{1}{2}\left[\frac{n!}{1!(n-1)!}\left(\frac{n!}{1!(n-1)!}+1\right) + \frac{n!}{2!(n-2)!}\left(\frac{n!}{2!(n-2)!}+1\right) + ... \right.$$

$$\left. ... + \frac{n!}{k!(n-k)!}\left(\frac{n!}{k!(n-k)!}+1\right) + ... + \frac{n!}{(n-1)!1!}\left(\frac{n!}{(n-1)!1!}+1\right)\right].$$

Отметим, что с помощью соотношения (3.19) можно определить класс задач дискретной оптимизации, в которых решение определяется только сочетанием переменных и не зависит от перестановки переменных в $S_r(C_n^r)$, т. е. в этих задачах значение C_{lj} зависит только от сочетания переменных в $S_r(C_n^r)$.

В общем случае задачу булевого программирования можно представить в виде

$$f(X_1, X_2, ..., X_n) \Rightarrow \max;$$

$$g_j(X_1, X_2, ..., X_n) \le b_j; \quad j = (\overline{1,m}), \tag{3.20}$$

где

$$f(X_1, X_2, ..., X_n) \in H, g_j(X_1, X_2, ..., X_n) \in H,$$

$$b_j \in Z;\ Z \text{ — множество целых чисел; } X_i \in \{0, 1\}.$$

Рассмотрим граф $G(X, E)$ (рис. 3.1), в котором вершины X_i и X_j соединены ребром (i, j), если они могут быть объединены в клику. В графе $G(X,E)$ каждой вершине X_i соответствует переменная X_i.

Выделим в графе G произвольную клику $Q = X_p X_r ... X_m$, состоящую из r вершин, где $r < n$, и рассмотрим ее пересечения — с $S_r(C_n^r) \in f(X_1, X_2, ..., X_n)$ и с $S_r(C_n^r) \in g_j(X_1, X_2, ..., X_n)$. Каждое пересечение можно охарактеризовать суммами коэффициентов C_{lj}, стоящих при $S_r(C_n^r)$ в функционале $f(X_1, X_2, ..., X_n)$ и ограничениях $g_j(X_1, X_2, ..., X_n)$. При этом в общем случае произвольная клика Q всегда будет характеризоваться соответствующим весом по функционалу $f(X_1, X_2, ..., X_n)$ и не более чем m весами по ограничениям $g_j(X_1, X_2, ..., X_n)$. Таким образом, произвольная задача булевого программирования может рассматриваться как задача нахождения клики Q^* максимального веса по весам функционала в графе G, у которой все m весов по весам ограничений не превышают соответственно b_j. Например, если решается задача линейного программирования вида

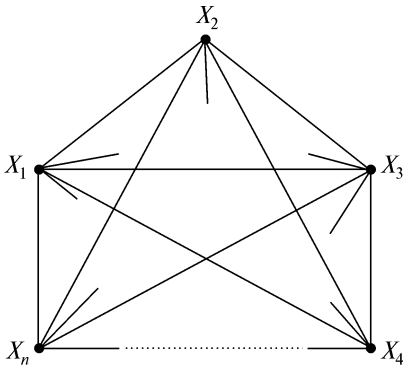

Рис. 3.1. Граф G

$$f(x) = C_1X_1 + C_2X_2 + C_3X_3 + C_4X_4 \to \max, \qquad (3.21)$$

$$B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4 \le b_1;$$

$$K_1X_1 + K_2X_2 + K_3X_3 + K_4X_4 \le b_2,$$

то граф G будет иметь вид, показанный на рис. 3.2.

Как видно из табл. 3.1, каждая вершина графа G характеризуется тремя весами — C_i, B_i, K_i. Для решения задачи в графе G на рис. 3.2 необходимо найти из взвешенных вершин такую клику Q^*, чтобы ее суммарный вес по весам $\{C_i\}$ был максимален, при условии, что суммарные веса по весам $\{B_i\}$ и $\{K_i\}$ не превышали величин b_1 и b_2 соответственно. Отметим, что в задачах линейного программирования и с нелинейностью выше второй

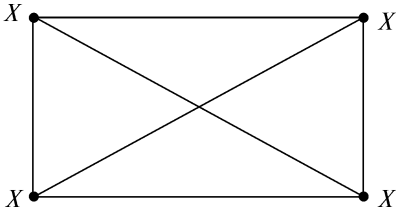

Рис. 3.2. Граф G для задачи линейного программирования

ТАБЛИЦА 3.1. **Веса вершин графа *G***

X_1	C_1	B_1	K_1
X_2	C_2	B_2	K_2
X_3	C_3	B_3	K_3
X_4	C_4	B_4	K_4

ТАБЛИЦА 3.2. **Веса вершин и ребер графа *G***

X_1	C_1	B_1	K_1
X_2	C_2	B_2	K_2
X_3	C_3	B_3	K_3
X_4	C_4	B_4	K_4

(при наличии в функции цели или ограничениях произведений, состоящих из более чем двух переменных) весовые характеристики ребер графа G полагают равными нулю. В случае решения задач квадратичного программирования удобно вводить и весовые характеристики ребер.

Рассмотрим задачу квадратичного программирования следующего вида:

$$f(x) = C_1X_1 + C_2X_2 + C_3X_3 + C_4X_4 + \\ + C_{12}X_1X_2 + C_{13}X_1X_3 + C_{14}X_1X_4 + \\ + C_{23}X_2X_3 + C_{24}X_2X_4 + C_{34}X_3X_4 \to \max, \qquad (3.22)$$

$$B_1X_1 + B_2X_2 + B_3X_3 + B_4X_4 \le b_1;$$

$$K_1X_1 + K_2X_2 + K_3X_3 + K_4X_4 \le b_2.$$

Выражению (3.22) можно поставить в соответствие граф на рис. 3.3.

Как видно из рис. 3.3 и табл. 3.2, вершинам графа G, как и в задаче линейного программирования, соответствуют веса (C_i, B_i, K_i), а ребрам — веса $\{C_{ij}\}$.

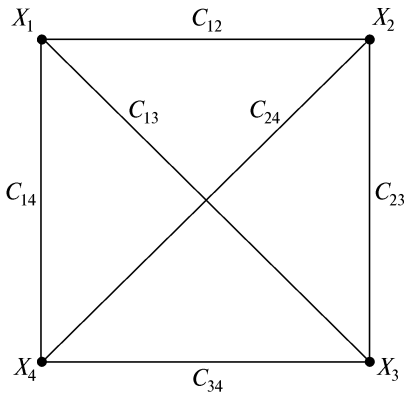

Рис. 3.3. Граф G для задачи квадратичного программирования

Для решения данной задачи требуется в графе G найти клику Q^* максимального суммарного веса по коэффициентам C_i и C_{ij} такую, чтобы суммы весов $\{B_i\}$ и $\{K_i\}$ не превышали величин b_1 и b_2 соответственно. В этом случае граф G является взвешенным и по вершинам, и по ребрам, и взвешенными являются все клики данного графа. При этом в квадратичной задаче выполняется следующее равенство: $C_{ij} = C_{ji}$.

Таким образом, для решения произвольной задачи булевого программирования необходимо построить алгоритм, позволяющий находить в заданном графе G, взвешенном по вершинам или по вершинам и по ребрам, клику Q^* максимального суммарного веса по весам функционала $f(X_1, X_2, ..., X_n)$, у которой все m весов по весам ограничений $g_j(X_1, X_2, ..., X_n)$ не превышают соответствующих b_j.

Метод решения задачи

Для решения поставленной задачи используется представление исходного графа G в виде симметричного дерева путей, предложенного в работах [10, 15, 16, 18, 19].

Пусть все возможные состояния некоторой системы определяются графом $G(V,E)$ с n вершинами, в котором вершины соответствуют возможным состояниям системы. Перейдем к пространству с $(n-1)^2$ состояниями. Для этого каждому из n состояний поставим в соответствие еще $(n-1)$ состояние, характеризующее способ достижения состояния из множества $\{1, 2,..., n\}$. При этом в качестве добавляемых состояний определим ранг пути в графе $G(V,E)$. Исходя из этого, из вершины s в произвольную вершину j графа $G(V,E)$ можно попасть путем ранга $r = 1$, используя одно ребро, путем ранга $r = 2$, используя 2 ребра, и т. д. и путем ранга $r = n - 1$, используя $(n-1)$ ребро. Такое пространство состояний можно представить в виде стянутого дерева путей D (рис. 3.4). Дерево всех путей D содержит $(n-1)$ горизонтальную линейку и $(n-1)$ ярус. Для прочтения путей на каждой горизонтальной линейке можно бывать только один раз.

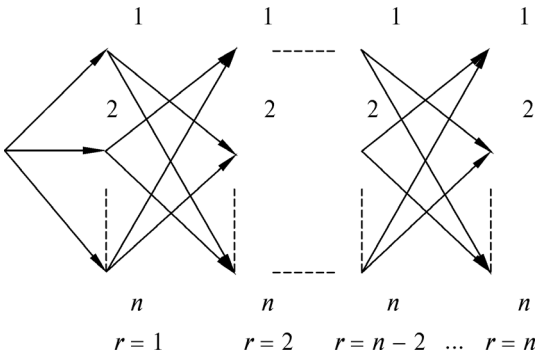

Рис. 3.4. Граф D

Исходя из стянутого дерева путей, для произвольной вершины j множество путей, ведущих в эту вершину из некоторой вершины s, представим в виде

$$m_s(j) = m_{sj}^{r=1} \cup m_{sj}^{r=2} \cup .... \cup m_{sj}^{r=n-1};\ j = \overline{(1, n-1)}, \qquad (3.23)$$

где $m_{sj}^r = \{\mu_{sj}^r\}$ — подмножества путей из произвольной вершины s в некоторую вершину j графа $G(V,E)$ ранга r.

Отметим, что дерево всех путей D может строиться и от конкретной вершины i графа, в этом случае вершина $s = i$ и i-я горизонтальная линейка исключаются в дереве D. Например, при $i = 2$ дерево D будет иметь вид, приведенный на рис. 3.5. В дальнейшем стянутое дерево путей будет использоваться для построения однопроходных алгоритмов решения задачи (3.2), а дерево D — для построения n-проходных алгоритмов, суть которых будет рассмотрена далее.

Таким образом, используя граф D и введя правила формирования путей следующего ранга, можно из произвольной вершины s поэтапно строить пути $\{\mu_{sj}^r\}$ произвольного ранга до ранга $r = n - 1$. Под состоянием системы будем понимать различные способы объединения вершин графа D в клики. Тогда каждому пути $\{\mu_{sj}^r\}$ ранга r в графе D, проходящем через вершины (v_h, v_k ... ... v_p) в исходном графе G, решаемой задачи соответствует клика из $(r - 1)$-й вершины ($X_h X_k ... X_p$), характеризующаяся соответствующим весом по функционалу $f(X_1, X_2, ..., X_n)$ и не более чем m весами по ограничениям $g_j(X_1, X_2, ..., X_n)$. Весовые характеристики $\{d_{sj}^{r-1}\}$ произвольной клики $Q^{r-1}(j)$, состоящей из $(r - 1)$ вершины и определяемой одним из путей $\mu_{sj}^r \in m_{sj}^r$ ранга r в графе D, вычисляются по весам функционала суммированием коэффициен-

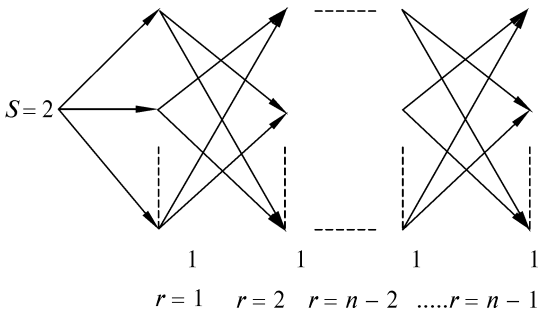

Рис. 3.5. Стянутое дерево всех путей D графа от вершины $s = 2$ $G(V,E)$

тов подмножества $L_f = \{C_{rj}\}$, стоящих при $S_{r-1}(C_n^{r-1}) \in P_f$, где P_f — все подмножества $\{S_{r-1}(C_n^{r-1})\}_f$, удовлетворяющие условию $S_{r-1}(C_n^{r-1})_f \cap Q^{r-1}(j) \neq \varnothing$, а $S_{r-1}(C_n^{r-1})_f$ определяется функционалом $f(X_1, X_2, ..., X_n)$.

Аналогично находят весовые характеристики по весам ограничений суммированием коэффициентов подмножества $L_B = \{C_{rj}\}$, стоящих при $S_{r-1}(C_n^{r-1}) \in P_B$, где P_B — все подмножества $\{S_{r-1}(C_n^{r-1})\}_B$, удовлетворяющие условию $S_{r-1}(C_n^{r-1})_B \cap Q^{r-1}(j) \neq \varnothing$, а $S_{r-1}(C_n^{r-1})_B$ определяется ограничениями $g_j(X_1, X_2, ..., X_n)$, $j = \overline{(1, m)}$.

Таким образом, весовые характеристики клик $Q^{r-1}(j)$, характеризующихся множеством путей m_{sj}^r по весам функционала и ограничений, определяются соответственно равенствами

$$d_{sj}^{f(r-1)} = \sum_{C_{rj} \in L_f} C_{rj}; \quad d_{sj}^{B(r-1)} = \sum_{C_{rj} \in L_B} C_{rj}. \tag{3.24}$$

Итак, в графе D каждый путь имеет в общем случае $(m + 1)$ длину по весам функционала и m — по весам ограничений. Таким образом, для решения поставленной задачи в графе D нужно построить путь максимальной длины по весам функционала от вершин 1, 2, ..., n ко всем остальным вершинам графа и при этом его длины по весам ограничений не должны превышать величины b_j.

Если на основе подмножеств путей $m_{sj}^{r=1}$ в графе D строить подмножества $m_{sj}^{r=2}$ и так далее до $m_{sj}^{r=n-1}$, то необходимо построить $(n - 1)!$ путей. Поэтому с целью снижения вычислительной сложности для формирования путей вводится процедура A, позволяющая отсекать неперспективные пути. Для отсечения неперспективных вариантов в процедуре A предлагается использовать принцип оптимизации по направлению к произвольной вершине p при формировании путей следующего ранга m_{sp}^{r+1} на основе путей предыдущего ранга m_{sj}^r, предложенный в работах [10, 15, 16, 18, 19], который для рассматриваемой задачи определяется следующим рекуррентным соотношением:

$$\mu_{sp}^{r+1} = \max_j \{\{\mu_{sj}^r\} \cup (j, p)\}; \; j = \overline{(1, n)}; \; p = \overline{(1, n)}; \; j \neq p, \tag{3.25}$$

где (j, p) — ребро графа D; n — число различных вершин в графе D.

Рассмотрим возможность построения n-проходных и однопроходных процедур решения задачи (3.20) на стянутых деревьях, приведенных на рис. 3.4 и 3.5 соответственно.

Перед началом работы процедуры A_1 полагаем, что значение переменной $i = 1$.

Процедура A_1 с n проходами

Шаг 1. Переменной s присваиваем значение i ($s := i$) и из вершины s строим все возможные пути ранга $r = 1$ ко всем вершинам графа D, удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$; $j = \overline{(1, m)}$, при этом длины по весам функционала и ограничениям вычисляем в соответствии с соотношениями (3.6).

Шаг 2. Используя пути текущего ранга r, строим все возможные пути ранга $r := r + 1$, удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = \overline{(1, m)}$, с использованием рекуррентного соотношения (3.25). При этом проверка ограничений и выбор пути максимального по весам функционала осуществляются на основе вычислений длин путей по весам функционала и ограничений в соответствии с соотношениями (3.24). Если в процессе применения рекуррентного соотношения (3.25) получаем несколько путей одинаковой длины, то их всех необходимо продлевать на следующем ранге.

Шаг 3. Проверяем $m_{sj}^{r+1} = \varnothing$, если да, то путь μ_{sj}^{*r} максимальной длины, полученный на ранге r, является локальным экстремумом решаемой задачи, иначе переходим к следующему шагу.

Шаг 4. Проверяем $i := n - 1$, если нет, то полагаем $i := i + 1$ и переходим к шагу 1, иначе процедура A_1 заканчивает работу. При этом из множества локальных экстремумов $\{\mu_{sj}^{*r}\}$ выбирается глобальный μ_{sj}^{**r}, соответствующий оптимальному решению задачи (3.20).

Для уменьшения временной сложности данного алгоритма можно использовать однопроходный вариант его реализации на основе стянутого дерева путей графа (см. рис. 3.4).

Процедура A_2 с 1 проходом

Шаг 1. Из вершины s строим все возможные пути ранга $r = 1$ ко всем вершинам графа D (см. рис. 3.4), удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = \overline{(1, m)}$, при этом длины по весам функционала и ограничениям вычисляем в соответствии с соотношениями (3.24).

Шаг 2. Используя пути текущего ранга r, строим все возможные пути ранга $r := r + 1$, удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = (\overline{1,m})$, с использованием рекуррентного соотношения (3.25). При этом проверка ограничений и выбор пути максимального по весам функционала осуществляются на основе вычислений длин путей по весам функционала и ограничений в соответствии с соотношениями (3.24). (Отметим, что если в процессе применения рекуррентного соотношения (3.25) получаем несколько путей одинаковой длины, то их всех необходимо продлевать на следующем ранге).

Шаг 3. Проверяем $m_{sj}^{r+1} = \varnothing$, если да, то путь μ_{sj}^{*r} максимальной длины, полученный на ранге r, является локальным экстремумом решаемой задачи, иначе переходим к следующему шагу.

Шаг 4. Проверяем ранг $r = n$, если нет, то переходим к шагу 2, иначе процедура A_2 заканчивает работу, и путь μ_{sj}^{*r} максимальной длины, полученный на ранге $r = n$, соответствует оптимальному решению задачи.

Еще одним вариантом уменьшения временной сложности алгоритмов на основе процедур A_1 и A_2 являются процедуры A' и A'', отличающиеся от A_1 и A_2 тем, что на каждом ярусе формирования путей на основе процедур A_1 и A_2 локальные экстремумы будут выделяться не в каждом множестве, а выделяются глобальные экстремумы на ярусе и на основе пути, соответствующего глобальному экстремуму на ярусе, формируются пути следующего яруса, удовлетворяющие ограничениям.

Процедура A′

Перед началом работы процедуры A' полагаем, что значение переменной $i := 1$.

Шаг 1. Переменной s присваиваем значение i ($s := i$) и из вершины s строим все возможные пути ранга $r = 1$ ко всем вершинам графа D (см. рис. 3.5), удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = (\overline{1,m})$. При этом длины по весам функционала и ограничениям вычисляются в соответствии с соотношениями (3.6). Далее определяем самый длинный путь на ярусе (ранге).

Шаг 2. Используя самый длинный путь текущего ранга r, построенный на предыдущем шаге, строим все возможные пути ранга

$r := r + 1$, удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = (\overline{1, m})$, с использованием рекуррентного соотношения (3.25). При этом проверка ограничений и выбор пути максимального по весам функционала осуществляются на основе вычислений длин путей по весам функционала и ограничений в соответствии с соотношениями (3.24).

Шаг 3. Проверяем $m_{sj}^{r+1} = \varnothing$, если да, то путь μ_{sj}^{*r} максимальной длины, полученный на ранге *r*, является локальным экстремумом решаемой задачи, иначе переходим к следующему шагу.

Шаг 4. Проверяем $i := n - 1$, если нет, то $i := i + 1$ и переходим к шагу 1, иначе процедура *A′* заканчивает работу; при этом из множества локальных экстремумов $\{\mu_{sj}^{*r}\}$, полученных за один проход, выбирается глобальный μ_{sj}^{**r}, соответствующий оптимальному решению задачи.

Процедура A″

Шаг 1. Из вершины s строим все возможные пути ранга $r = 1$ ко всем вершинам графа *D* (см. рис. 3.5), удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = (\overline{1, m})$, при этом длины по весам функционала и ограничениям вычисляются в соответствии с соотношениями (3.6). Далее выделяем самый длинный путь на ярусе.

Шаг 2. Используя самый длинный путь текущего ранга *r*, построенный на предыдущем шаге, строим все возможные пути ранга $r := r + 1$, удовлетворяющие ограничениям $g_j(X_1, X_2, ..., X_n)$, $j = (\overline{1, m})$, с использованием рекуррентного соотношения (3.25). При этом проверка ограничений и выбор пути максимального по весам функционала осуществляются на основе вычислений длин путей по весам функционала и ограничений в соответствии с соотношениями (3.24).

Шаг 3. Проверяем $m_{sj}^{r+1} = \varnothing$, если да, то путь μ_{sj}^{*r} максимальной длины, полученный на ранге *r*, является локальным экстремумом решаемой задачи, иначе переходим к следующему шагу.

Шаг 4. Проверяем ранг $r = n$, если нет, то переходим к шагу 2. Иначе процедура *A″* заканчивает работу, и путь μ_{sj}^{*r} максимальной длины, полученный на ранге $r = n$, является решением задачи.

Оценка сложности процедур {A}

Поскольку число путей, строящихся на произвольном ранге r, не может превысить величины $(n-1)\times(n-1)$, максимальный ранг r произвольного пути не превышает $(n-1)$, а число циклов, выполняемое процедурой A_1, равно n, то после n циклов число путей, которое построит процедура A_1, не превышает $(n-1)\times$ $\times(n-1)\times(n-1)n \approx n^4$, а число обработанных векторов — n^5. С учетом количества слагаемых в функционале (k) и количества ограничений (m) временная сложность алгоритма A_1 не превысит $O(n^5\,k(m+1))$. В случае, когда решение задачи осуществляется за один проход процедуры A_2 или за один проход процедуры A'', но с выделением наиболее длинного пути на ярусе, сложность процедур A_2 и A'' не превысит $O(n^4\,k(m+1))$ и $O(n^3\,k(m+1))$ соответственно. Итак, алгоритмы A_5, A_4, A_3 имеют временную сложность, не превышающую $O(n^5\,k(m+1))$, $O(n^4\,k(m+1))$ и $O(n^3k(m+1))$ соответственно.

Таким образом, предложенный подход решения произвольных задач булевого программирования позволяет на основе разработанных алгоритмов решать с единых позиций любые задачи линейного и нелинейного программирования за полиномиальное время с требуемой точностью. Если в соотношении (3.19) ввести обозначение $f_k(x) = \sum_{j=1}^{p_k} C_{kj} S_k(C_n^k)$, то оно примет вид $F(x) =$ $= \sum_k f_k(x)$. В общем случае, если функционал и ограничения представляют собой произвольную нелинейную функцию от $f_k(x)$, то такая задача нелинейного программирования также может быть эффективно решена с помощью предложенного подхода и при этом будут изменяться только правила весов вершин клик.

3.2. ПОДХОДЫ К УПРАВЛЕНИЮ И РАСПРЕДЕЛЕНИЮ РЕСУРСОВ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Современное понимание РВС широко использует понятие «ресурс», включающее в себя все, что участвует в обработке данных. К ресурсам следует отнести коммуникации, системы хранения и обработки данных, информационные системы, а также программное обеспечение. Так, если приложение анализа данных понимать как ресурс, то это означает возможность удален-

ного доступа к нему без установки на компьютер пользователя. Выход за рамки задач высокопроизводительных вычислений выявляет реальное содержание РВС как инфраструктуры для поддержки любой глобально распределенной обработки для множества типов приложений: электронного бизнеса, распределенного производства, исследования данных. При этом не требуются высокопроизводительные коммуникации; в этом случае в качестве коммуникационной среды может выступать Internet. Известные подходы к организации РВС используют двухуровневую организацию, для которых необходимо решать задачи планирования и эффективного использования ресурсов в контексте не только их географической распределенности, но и временной. Планирование распределением ресурсов осуществляется централизовано, и при этом важной задачей является определение максимального количества ресурсов, которое можно определить за некоторое допустимое время, определяющее актуальную информацию о состоянии ресурсов.

Модель управления

Постановка задачи

Пусть исходными данными для второго (управляющего) уровня РВС является множество Z сертифицированных задач (в рамках стандартных методов описания заданий) и множество сертифицированных ресурсов Y. В результате сертификации каждую задачу характеризуют объемом памяти, производительностью ресурса, количеством операций ввода/вывода, максимальным ожидаемым временем решения, стоимостью ее решения и т. д. Каждый ресурс характеризуют слотом — интервалом Δt_i времени, на котором данный ресурс может быть использован; количеством выделяемых процессорных узлов; принятой политикой планирований заданий и т. д. Будем полагать, что в подмножестве $z \in Z$ сертифицированные задачи могут решаться независимо друг от друга. Подмножества ресурсов $y \in Y$, предназначенных для решения подмножества задач $z \in Z$, входят в виртуальные организации (пользователи, владельцы ресурсов, провайдеры) (ВО). При этом планирование выполнения заданий должно учитывать не только пространственное распределение ресурсов, но и временное распределение, задаваемое множеством интервалов $\{\Delta t_i\}$. Для удобства планирования распределением задач по ресурсам внутри ВО предлагается разбить подмно-

жества ресурсов с учетом их временной занятости на такие подмножества, в которых интервалы времени Δt_i не пересекаются. Последнее условие может быть обеспечено на основе следующей формальной модели.

Пусть отрезок времени $\Delta t_i = (t_k, t_q)$ — интервал, в течение которого можно задействовать некоторый произвольный j-й ресурс для выполнения заданий: с момента времени t_k этот ресурс назначается для выполнения заданий и освобождается в момент времени t_q. Таким образом, в общем случае имеется n интервалов и требуется найти минимальное число ВО, на которое можно разбить свободные ресурсы таким образом, чтобы в каждой ВО все ресурсы независимо друг от друга выполняли задания.

Для решения задачи построим граф G, в котором каждой вершине графа i соответствует отрезок Δt_i, и соединим вершины i и j в графе G в случае, если отрезки Δt_i и Δt_j пересекаются во времени.

***Утверждение* 3.1.** Искомое число ВО равно хроматическому числу графа G.

Действительно, на всех ресурсах имеющие одинаковый цвет задания могут решаться независимо друг от друга. Однако когда есть некоторое число ресурсов для обслуживания всех заданий, то, окрасив одним цветом все отрезки, соответствующие вершинам графа G, получаем виртуальные сообщества ресурсов, в которых времена работы ресурсов не пересекаются во времени, и при этом имеем некоторую правильную раскраску графа G. Эти соображения и обосновывают данное утверждение.

Следовательно, минимальное число красок, с помощью которых можно раскрасить граф G, и определит минимальное число ВО, в которых можно независимо выполнить некоторое подмножество заданий.

Оптимальная раскраска графов. Отметим, что раскраска графов является одной из основных задач в теории графов и построении вычислительных систем и сетей, эффективное решение которой необходимо для различных приложений [20].

Как показано в [20], задача раскраски графа может рассматриваться как задача линейного булевого программирования. Решение данной задачи может быть получено существующими методами решения задач булевого программирования, но при этом из-за большой размерности эту задачу целесообразно решать в графовой постановке, используя процедуру динамического программирования, предложенную в [20] на основе следующей теоремы.

***Теорема* 3.1.** Если граф G является r-хроматическим, то он может быть раскрашен с использованием r (или меньшего числа) красок с помощью следующей процедуры: сначала в один цвет окрашивается некоторое максимальное независимое множество $S_1[G]$, затем окрашивается в следующий цвет множество $S_2\left[\langle X - S_1[G]\rangle\right]$ и т. д. до тех пор, пока не будут раскрашены все вершины.

Тот факт, что такая раскраска, использующая только r цветов, всегда существует, может быть установлен следующим образом. Пусть имеется раскраска в r цветов такая, что одно или больше множеств, окрашенных в один и тот же цвет, не являются максимальными независимыми множествами в смысле, указанном выше. Пронумеруем цвета произвольным образом. Очевидно, что мы можем всегда покрасить в цвет 1 те вершины (пусть это множество $\bar{V}_1$), которые не были окрашены в этот цвет и которые образуют независимое максимальное множество вместе с множеством V_1 всех вершин графа, уже окрашенных в цвет 1. Эта новая раскраска возможна потому, что никакая вершина из множества $\bar{V}_1$ не является смежной ни с какой вершиной из $\bar{V}_1$, и, следовательно, всякая вершина, смежная хотя бы с одной вершиной из $\bar{V}_1$, окрашена в цвет, отличный от цвета 1, и поэтому не затрагивается процедурой перемены цвета вершин из $\bar{V}_1$. Рассматривая теперь подграф $\langle X - V_1 \cup \bar{V}_2\rangle$ и проводя с ним аналогичные манипуляции, окрасим в цвет 2 какое-то (новое) независимое максимальное множество и т. д. Раскраску указанного в теореме вида называют *оптимальной независимой раскраской* [20].

Таким образом, если имеется эффективный алгоритм определения минимального вершинного покрытия произвольного графа (обозначим его A_M) и процедуры динамического программирования, определяемой теоремой 3.1, то можно получить следующий алгоритм раскраски произвольного графа.

Алгоритм раскраски графа

Шаг 1. С помощью алгоритма A_M находим максимальное независимое множество вершин и присваиваем им цвет (1).

Шаг 2. Удаляем из графа все вершины, покрашенные в цвет (B), инцидентные им ребра и запоминаем для всех оставшихся

вершин, которые инцидентны вершинам с цветом (B), что они не могут быть покрашены в цвет (B). При этом граф распадается на некоторое количество Q несвязанных компонентов и компонентам, состоящим из изолированных вершин, присваиваем цвет ($B + 1$).

Шаг 3. Для каждого компонента Q выполняем шаги 1 и 2 до тех пор, пока множество исходных вершин графа не станет пустым.

В качестве такого алгоритма можно использовать предложенный в работе [21] алгоритм с временной сложностью, не превышающей $O(n^3)$, и погрешностью, не превышающей 2—6 %.

Поскольку временная сложность алгоритма A_M не превышает $O(n^3)$ и его не придется повторять более n раз, то сложность алгоритма оптимальной раскраски не превысит $O(n^4)$. Как показано в [21], данный алгоритм может быть реализован на параллельных вычислительных системах, что позволит в n раз снизить временную сложность алгоритма. После того, как требуемое количество ресурсов, которое может обеспечить своевременное выполнение заданий, определено, необходимо решить задачу о назначении, учитывая особенности процесса распределения ресурсов, а также их временное и пространственное распределения.

В задаче о назначении необходимо распределить (назначить) ресурсы для выполнения заданий; при этом каждый ресурс может выполнить любое задание, но с различной производительностью. Если некоторому заданию назначается ресурс с производительностью, требуемой для его выполнения, то стоимость ее выполнения будет меньше, чем в случае назначения на выполнение данного задания ресурса с меньшей производительностью. Цель задачи — найти оптимальное (с точки зрения критерия или критериев) распределение ресурсов по всем заданиям, находящимся в РВС.

Исходная матрица для решения задачи назначения m ресурсов на n заданий ($m = n$) приведена в табл. 3.3, где C_{ij} — стоимость назначения ресурса i для задания j. Если по определенным ограничениям ресурс не может быть использован, то стоимости присваивают «запрещающее» значение, которое исключит попадание такой пары «ресурс—задание» в искомое решение. Для нахождения решения задачи о назначении существующими методами требуется, чтобы матрица стоимостей была квадратной, т. е. число «исполнителей» должно быть равно числу работ. В случае неравенства количества ресурсов количеству заданий в модель вводятся фиктивные ресурсы и/или фиктивные задания.

Т А Б Л И Ц А 3.3. **Исходная матрица для решения задачи**

Ресурсы	Задания				
	1	2	j	$n-1$	n
1	$C_{1,1}$	$C_{1,2}$	$C_{1,j}$	$C_{1,n-1}$	$C_{1,n}$
2	$C_{2,1}$	$C_{2,2}$	$C_{2,j}$	$C_{2,n-1}$	$C_{2,n}$
...	...	...	...	...	...
$m-1$	$C_{m-1,1}$	$C_{m-1,2}$	$C_{m-1,j}$	$C_{m-1,n-1}$	$C_{m-1,n}$
m	$C_{m,1}$	$C_{m,2}$	$C_{m,j}$	$C_{m,n-1}$	$C_{m,n}$

Задача о назначении является частным случаем транспортной задачи, в которой ресурсы эквивалентны пунктам отправления, а задания — пунктам назначения. При этом все величины спроса и предложения равны 1, а стоимость назначения ресурса i на задание j соответствует C_{ij}. Указанные условия позволяют решать задачу о назначениях теми же методами, что и транспортную. Однако та особенность, что все величины спроса и предложения в задаче о назначениях принимаются равными 1, позволяет получать более эффективные методы ее решения.

Рассматриваемый подход позволит автоматизировать процесс решения задачи распределения ресурсов как с учетом временных требований к выполнению заданий, так и с учетом затратных ограничений, возникающих в процессе управления выполнением заданий. Особенностью планирования при таком подходе является то, что время решения заданий в ВО будет равно сумме отрезков времени $\{\Delta t_i\}$ множества, определяющих слоты использования ресурсов ВО. В случаях, когда в кластерах на данный момент времени отсутствуют необходимые ресурсы, возможно перераспределение заданий внутри кластера с целью максимизации вычислительных ресурсов для обеспечения качества обслуживания пользователей ВО.

3.3. ЗАДАЧА МИНИМИЗАЦИИ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ В РВС

Рассмотрим двухуровневую РВС, в которой на первом уровне несколько независимых брокеров распределяют вычислительные задачи на кластеры РВС, а на втором уровне задачи распределяются локальными планировщиками и системами управления ресурсами (ЛСУР). Пусть в РВС имеется n кластеров, при этом каждый i-й узел кластера может решить некоторое подмножество

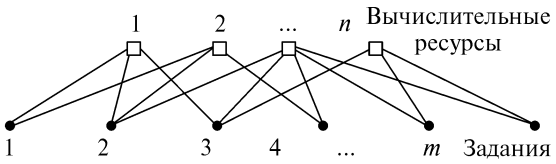

Рис. 3.6. Граф планирования заданий на вычислительные ресурсы

L_i задач с требуемой производительностью. Предположим, что в определенный момент времени в РВС на решение поступает подмножество $m \subseteq M$ заданий, которые нужно решить на заданном множестве вычислительных ресурсов n, используя при этом минимальное количество вычислительных ресурсов, обеспечивающих решение всех m задач ($m > n$) с возможными ограничениями (например, на время выполнения и бюджет).

Для представления такой постановки задачи используется двудольный граф (рис. 3.6), в котором вершины одного множества (вычислительные ресурсы) соединены ребрами с вершинами другого множества — заданиями, которые могут выполняться на ресурсах, являющихся на момент планирования доступными и свободными. Отметим, что при этом предполагается, что одно задание может решаться на одном из нескольких свободных ресурсов.

Применим данное представление для формирования некоторой булевой матрицы B, в которой строкам соответствуют задания, а столбцам — вычислительные ресурсы. Элемент матрицы (i, j) положим равным 1, если j-й узел способен с требуемой эффективностью обеспечить решение i-й задачи, и равным 0 — в противном случае. Таким образом, задача планирования сводится к задаче определения минимального количества столбцов (ресурсов), покрывающих единицами все строки (задания) булевой матрицы B, имеющей размерность $m \times n$, т. е. к задаче о наименьшем покрытии (ЗНП).

Эта задача может быть сформулирована как задача линейного булевого программирования, постановка которой в общем случае имеет вид

$$L = \sum_{j=1}^{n} c_j x_j \to \min \tag{3.26}$$

при ограничениях

$$\begin{aligned} &\sum_{j=1}^{n} \beta_{ij} x_j \geq 1, \quad \mathrm{i} = \overline{1, m}; \\ &x_j \in \{0,1\}; \quad c_j \geq 0, \end{aligned} \tag{3.27}$$

где

$$\beta_{ij} = \begin{cases} 1, \text{ если } i\text{-я переменная может быть покрыта} \\ \quad \text{переменной } x_j, \\ 0 \text{ в противном случае.} \end{cases} \qquad (3.28)$$

Задачи о наименьшем покрытии и о наименьшем вершинном покрытии (ЗНВП) имеют широкое прикладное значение в теории построения сложных систем, в системах диагностики вычислительных систем и сетей [10—12] и при разработке их программного и математического обеспечения, а также для планирования распределения ресурсов в РВС [10]. При этом основным требованием к алгоритмам решения задач этого класса является обеспечение высокой оперативности и минимальной погрешности получаемых решений.

Задачу нахождения минимальных вершинных покрытий (независимых максимальных множеств) можно решать последовательным перебором независимых множеств с одновременной проверкой каждого множества на максимальность (последнее осуществляется добавлением к исследуемому множеству дополнительной вершины, не принадлежащей ему, и определением, сохраняется ли при этом независимость) и запоминанием максимальных множеств. Однако с увеличением количества вершин этот способ становится весьма громоздким. На основе усовершенствования этой процедуры построены алгоритмы Брона и Кэрбоша [20]. Как показано в работе [21], задача вершинного покрытия является *NP*-полной, и эффективные алгоритмы ее решения для произвольных графов неизвестны. Для двудольных графов на основе алгоритмов Хопкрофта и Карпа (с поиском в глубину) разработаны методы [22], позволяющие находить минимальное вершинное покрытие и максимальное независимое множество вершин в произвольном двудольном графе $H = <X, Y, E>$ за время $O((m + n)\sqrt{n})$, где $n = |X \cup Y|$ и $m = |E|$. Полиномиальные алгоритмы определения числа устойчивости были получены для совершенных графов — графов, у которых для любого его порожденного подграфа хроматическое количество равно кликовому числу. Алгоритм вычисления числа устойчивости графа [23] основан на методе эллипсоидов и использует процедуру отделения матриц графа. Однако в вычислительном плане этот алгоритм имеет существенные недостатки, не позволяющие использовать его на практике. Как показано в работе [23], получить правильное решение при количестве вершин в графе, превышающем 10, практически не-

возможно. Применение *r*-алгоритмов позволило увеличить размерность решаемых задач до 50, а при использовании двойственных оценок в схеме ветвей и границ — до 100, с погрешностью, не превышающей 5 % [23]. Таким образом, поскольку размерности задач, решаемых в РВС, достаточно велики, представляется актуальной разработка эффективного приближенного алгоритма решения таких задач с единых позиций ЗНВП и ЗНП, имеющего малую временную сложность и погрешность.

Постановка и метод решения задачи. Для решения с единых позиций задач ЗНВП и ЗНП представим граф $G(X, E)$ и произвольную матрицу B в виде булевой функции. Рассмотрим конъюнктивное представление булевой матрицы B. Пусть задана булевая матрица B с n столбцами и m строками. Столбцы будем задавать вектором $X = \{x_1, x_2, ..., x_n\}$, строки — вектором $M = \{\mu_1, \mu_2, ..., \mu_m\}$. Покрытием Q строк M назовем такое множество столбцов B, которое покрывает единицами все строки M. Для определения всех покрытий матрицы будем использовать алгебраический метод получения по импликантной таблице приведенных систем простых импликант булевых функций. Если каждый столбец из совокупности $X = \{x_1, x_2, ..., x_n\}$ рассматривать как простую импликанту, покрывающую совокупность строк $M = \{\mu_1, \mu_2, ..., \mu_m\}$, каждую строку μ_i — как набор переменных, покрываемых простыми импликантами, то матрицу B можно представить как импликантную таблицу булевой функции. При такой интерпретации матрицы B для каждой строки μ_i можно записать дизъюнкцию столбцов b_i, покрывающих рассматриваемую строку, в следующем виде:

$$\begin{aligned} d\mu_I &= (x_1 \vee x_K \vee .....), \\ &..... \\ d\mu_I &= (x_p \vee x_t \vee .....). \end{aligned} \qquad (3.29)$$

Конъюнкция дизъюнкций (3.29) по всем строкам $\mu_1, \mu_2, ..., \mu_m$ матрицы B образует конъюнктивное представление матрицы B, содержащее в себе все покрытия совокупности строк M:

$$k(M) = d\mu_1 \cdot d\mu_2 \cdot .... \cdot d\mu_r = (x_1 \vee x_K \vee .....) \cdot ... \cdot (x_p \vee x_t \vee .....). \quad (3.30)$$

Раскрывая скобки в (3.30) в соответствии с законами дистрибутивности, получаем дизъюнктивное представление матрицы B, образующее перечень всех возможных покрытий совокупности строк $M = \{\mu_1, \mu_2, ..., \mu_m\}$. Так, в соответствии с (3.30) конъюнктивное представление матрицы B, задающей граф G (рис. 3.7), будет иметь вид

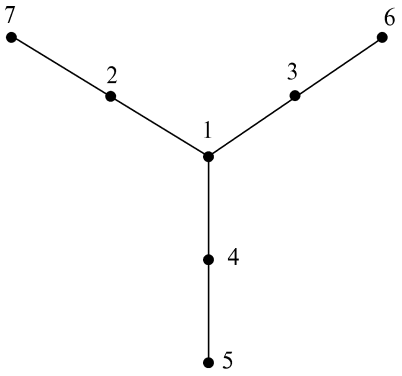

Рис. 3.7. Граф G

$$F = (X_1 \vee X_2)(X_1 \vee X_3)(X_1 \vee X_4) \times \times (X_2 \vee X_7)(X_3 \vee X_6)(X_4 \vee X_5), \quad (3.31)$$

$$B = \begin{vmatrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1-2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1-3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1-4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2-7 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 3-6 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 4-5 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{vmatrix}.$$

Как показано в работе [24], если f — булевая функция, построенная по графу $G = (V, E)$ в виде произведения дизъюнктов $(v_i \vee v_j)$, где $\{v_i\} \in \{0,1\}$, $i = \overline{(1,n)}$, $j = \overline{(1,n)}$, $i \neq j$, и при этом каждый дизъюнкт $(v_i \vee v_j)$ соответствует ребру (v_i, v_j), то все наборы переменных $\{v_i, v_j\}$, на которых она принимает значение «истинно», соответствуют вершинным покрытиям в графе $G = (V, E)$. Для перечисления всех вершинных покрытий графа $G = (V, E)$ необходимо определить те системы значений переменных $\{v_i, v_j\}$, при которых высказывание

$$F(V_1, V_2 ..., V_n) = 1 \quad (3.32)$$

«истинно». Чтобы найти эти системы значений переменных $\{v_i, v_j\}$, необходимо привести левую часть (3.31) к минимальной дизъюнктивной нормальной форме (ДНФ), раскрывая скобки и используя закон поглощения. Такая форма единственная ввиду отсутствия в (3.32) логических отрицаний.

Введем следующие понятия и определения для элементов булевой матрицы B, задающей некоторый граф $G(X, E)$. Если в матрице B выделить произвольный столбец j, то с ним можно связать некоторое подмножество столбцов $\{q\}$, с которыми данный столбец пересекается, поскольку столбец j соответствует вершине графа, а количество единиц в столбце определяется степенью вершины d_j, тогда подмножество $\{q\}$ будет всегда содержать d_j таких столбцов. Будем полагать, что подмножество столбцов $\{q\}_j$ образует связку столбцов относительно столбца j. Количество столбцов в B равно количеству вершин n в графе $G(X, E)$, а количество возможных связок столбцов для матрицы B равно n. Отметим, что все столбцы в матрице B пересекаются друг с

другом только в одной строке, поскольку в каждой строке матрицы B находится ровно по две единицы. Поэтому, с точки зрения их пересекаемости, возможность их принадлежности минимальному покрытию равновелика. Каждая связка столбцов $\{q\}_j$ покрывает определенное количество строк l_j; связку $\{q\}_j$, покрывающую максимальное количество строк в матрице B, назовем максимальной. Если в графе $G(X, E)$ есть висячие вершины, то это будет означать, что в матрице B есть столбцы P, содержащие только одну единицу, и, следовательно, связки этих столбцов будут содержать только по одному столбцу в каждой. Эти столбцы обладают интересным свойством, которое для произвольных графов $G(X, E)$ определяется следующим утверждением.

***Утверждение* 3.2.** Если граф $G(X, E)$ содержит некоторое подмножество висячих вершин $Q \in X$, то подмножество вершин $P \in X$, смежных с Q, может быть дополнено до одного из минимальных вершинных покрытий графа $G(X, E)$.

Доказательство. Пусть граф $G(X, E)$ содержит висячую вершину j, которая подсоединена к нему ребром (i, j). Тогда граф $G(X, E)$ можно представить в виде объединения подграфов G' и G'', при этом вершина i является точкой сочленения, т. е. принадлежит подграфам G' и G''. Пусть подграф G' содержит k вершин, тогда степени вершин в подграфе G' могут изменяться от 1 до $k - 1$. Предположим, что в графе $G(X, E)$ есть покрытие $\{X_i\}_{\min}$, которое не включает в себя вершину i, но тогда оно должно включать в себя вершину j (т. е. $j \in \{X_i\}_{\min}$) для того, чтобы ребро (i, j) оказалось покрытым.

Пусть степень вершины i в подграфе G' равна 1 и вершина i в подграфе G' соединена с X_1. Поскольку ребро (i, X_1) должно быть покрыто, а i в покрытие не входит, то $X_1 \in \{X_i\}_{\min}$. Следовательно, обе вершины $(j, X_1) \in \{X_i\}_{\min}$. Если вместо вершин (j, X_1) в покрытие ввести вершину i, то получим новое покрытие $\{X_i\}'$, которое содержит на одну вершину меньше, чем покрытие $\{X_i\}_{\min}$, что противоречит первоначальному предположению о том, что покрытие $\{X_i\}_{\min}$, содержащее вершину j, является минимальным вершинным покрытием в графе $G(X, E)$. Аналогичное рассуждение можно провести, полагая степень вершины i в G' равной 2, 3..., $k - 1$. Таким образом, по индукции следует, что во всех возможных случаях имеет место противоречие с первоначальным предположением, а значит, вершина i принадлежит минимальному вершинному покрытию. Поскольку аналогичное рассуждение можно провести для любой висячей вершины графа $G(X, E)$, то по индукции утверждение 3.2 можно считать доказанным.

В общем случае если в графе существует несколько минимальных вершинных покрытий, то могут существовать и минимальные вершинные покрытия без вершины, смежной с висячей вершиной графа $G(X, E)$, но если такое покрытие одно, то эта вершина обязательно в нем присутствует. Таким образом, если в матрице B есть столбец j, содержащий одну единицу, то связка столбцов относительно столбца j (в данном случае она состоит из одного столбца) принадлежит минимальному покрытию $X_{\min}$.

В случае представления графа в виде булевой функции $F(X_1, X_2, ..., X_n)$ утверждение (3.2) эквивалентно тому, что если в дизъюнктах есть переменная X_p, которая в них встречается один раз, то переменная X_n, составляющая ей пару, в дизъюнкции входит в покрытие. И, следовательно, данную дизъюнкцию можно заменить переменной X_n, при этом исключить из анализа все дизъюнкции, содержащие переменную X_n. В работе [25] при анализе графов $G(X, E)$, не имеющих висячих вершин и представленных в конъюнктивной нормальной форме в виде некоторой булевой функции F, для определения наименьшего множества переменных $\{X_i\}$, которые покрывают все дизъюнкты в конъюнктивном представлении графа $G(X, E)$, предложено использовать принцип суперпозиции в булевой алгебре, основанный на следующем равенстве:

$$f(X_1, X_2, ..., X_n) = f(X_1 = 0, X_2, ..., X_n) \vee f(X_1, X_2 = 0, ..., X_n) \vee ... \vee f(X_1, X_2, ..., X_n = 0). \quad (3.33)$$

Не нарушая принципа суперпозиции, соотношение (3.33) можно записать в виде

$$f(X_1, X_2, ..., X_n) = X_1 \cdot f(X_1, X_2, ..., X_n) \vee \vee X_2 \cdot f(X_1, X_2, ..., X_n) \vee ... \vee X_n \cdot f(X_1, X_2, ..., X_n). \quad (3.34)$$

Особенностью конъюнктивного представления графа $G(X, E)$ в виде булевой функции является то, что она содержит количество дизъюнктов, равное количеству ребер в графе, количество переменных в каждом дизъюнкте равно 2, и при этом каждая переменная соответствует некоторой вершине графа $G(X, E)$.

Введем понятие характеристического вектора $h_q = (h_{i=1}^{j_1}, h_{i=2}^{j_2}, ..., h_{i=n}^{j_n})_q$ некоторой булевой функции:

$$f_q = X_p \cdot X_h \cdot ... \cdot X_q \cdot f(X_1, X_2, ..., X_n), \quad (3.35)$$

в которой переменные X_p, X_h, ..., X_q не встречаются в функции $f(X_1, X_2, ..., X_n)$.

Вес j_i в векторе h_q указывает, как часто переменная X_i встречается в дизъюнктах функции $f(X_1, X_2, ..., X_n)$, а сам вектор бу-

дем описывать суммарной весовой характеристикой V_q, определяемой соотношением

$$V_q = \sum_{i=1}^{n} j_i \ . \qquad (3.36)$$

Рассмотрим алгоритм решения данной задачи в виде следующей процедуры A преобразования булевой функции $f(X_1,X_2...$ $...X_n)$, задающей некоторый граф $G(X, E)$.

Процедура А

Шаг 1. Проверяем, есть ли в дизъюнктах $f(X_1,X_2 ,..., X_n)$ переменные $\{X_k\}$, встречающиеся по одному разу. Если да, то умножаем $f(X_1,X_2 ,..., X_n)$ на переменные $\{X_t\}$, стоящие совместно в дизъюнктах с переменными $\{X_k\}$ и образующие сомножитель $X_p \cdot X_h \cdot ... \cdot X_q$, состоящий из r переменных, находившихся в дизъюнктах совместно с X_k. При этом все дизъюнкты в $f(X_1,X_2 ,..., X_n)$, содержащие переменные $X_p,X_h ,..., X_q$, исключаем из дальнейшего анализа.

Шаг 2. Проверяем в полученной функции $f = X_p \cdot X_h \cdot ... \cdot X_q \cdot f(X_1, X_2 ,..., X_n)$ встречаются ли оставшиеся переменные по одному разу или нет; если да, то получено минимальное вершинное покрытие, в которое входят вершины, соответствующие переменным $X_p \cdot X_h \cdot ... \cdot X_q$, выбираемым по одной из каждого оставшегося дизъюнкта, и алгоритм заканчивает работу; если нет, то переходим к следующему шагу.

Шаг 3. Полученные функции $f = X_p \cdot X_h \cdot ... \cdot X_q; f(X_1,X_2 ,..., X_n)$ умножаем поочередно на переменные X_i, оставшиеся в $f(X_1,X_2 ,..., X_n)$, и исключаем при этом дизъюнкты, содержащие X_j, из анализа. Если их q, то получаем q таких функций $\{f_1, f_2 ,..., f_q\}$ и для каждой функции вычисляем характеристический вектор $h_q = (h_{i=1}^{j_1}, h_{i=2}^{j_2},..., h_{i=n}^{j_n})_q$ и его весовую характеристику $V_q = \sum_{i=1}^{n} j_i$.

Шаг 4. Среди полученных функций выбираем функцию f_q с минимальным значением весовой характеристики и переходим к шагу 1. Если же при этом все весовые характеристики функций оказываются одинаковыми, то выбираем любую из них.

Данный алгоритм является жадным алгоритмом, который строит минимальное вершинное покрытие в графе $G(X, E)$, заданном в виде конъюнктивного представления матрицы B.

В процессе работы алгоритма, когда мы полагаем $X_i = 1$, появляется произведение X_pX_k, ..., X_c, которое поглощает все дизъ-

юнкты, содержащие эти переменные. Это эквивалентно удалению из графа вершин (i, p, k, ..., c) и инцидентных им ребер.

В результате такого преобразования исходного графа возможны два варианта: вновь полученный подграф может содержать висячие вершины или нет. Рассмотрим эти случаи.

1. Если подграф содержит висячую вершину, то, включая ее в минимальное покрытие и удаляя из графа вместе с инцидентными ей ребрами, получаем новый подграф. Если в процессе преобразования каждый раз получаем подграф с висячими вершинами, то в соответствии с утверждением 3.2 минимальное вершинное покрытие исходного графа получим в худшем случае за n шагов. Так, если исходный граф является деревом, содержащим n ярусов и имеющим ширину (максимально возможное количество вершин на ярусе дерева), равную h, то данная стратегия последовательного включения в покрытие вершин, смежных с висячими, позволит построить минимальное вершинное покрытие дерева за $O(hn)$ шагов.

2. В случае, если после удаления вершин (i, p, k,..., c) и ребер, им инцидентным, получаем подграф, не содержащий висячих вершин, можно выделить три варианта: два предельных случая — получаемый подграф является полносвязной компонентой или образует простой цикл на оставшемся множестве вершин: оптимальность работы алгоритма в этих крайних случаях очевидна, и промежуточный вариант — подграф не имеет висячих вершин и степени его вершин произвольны. Для обоснования оптимальности работы алгоритма в этом случае докажем следующую теорему.

***Теорема* 3.2.** Если, полагая $X_j = 1$, получаем связку столбцов $\{q\}_j^*$, покрывающих максимальное количество строк в матрице B, то применение к ней процедуры A приводит к построению минимального покрытия матрицы B.

Доказательство. Предположим, что множество столбцов, принадлежащее $\{q\}_j^*$, покрывает максимально возможное из всех связок столбцов количество строк l, а оставшиеся $(m - l)$ строк являются непокрытыми. Процедура A, добавляющая количество столбцов до полного покрытия строк матрицы B, является процедурой, которая на каждом шаге добавляет в покрытие столбец, покрывающий максимальное количество строк из непокрытых. Следовательно, к столбцам $\{q\}_j^*$ добавится минимальное количество столбцов $\{p\}$. Таким образом, получаем покрытие, состоящее из множества столбцов $\{q\}_j^* \cup \{p\}$. Предположим, что оно не

является минимальным, но так как количество добавленных до полного покрытия столбцов $\{p\}$ процедурой A минимально, последнее возможно, если существует связка столбцов, а связку столбцов $\{q\}_j$ покрывает количество строк $k > l$. Это противоречит первоначальному предположению о том, что $\{q\}_j^*$ является максимальной связкой столбцов. Следовательно, наше предположение неверно и множество столбцов образует минимальное покрытие матрицы B графа $G(X, E)$.

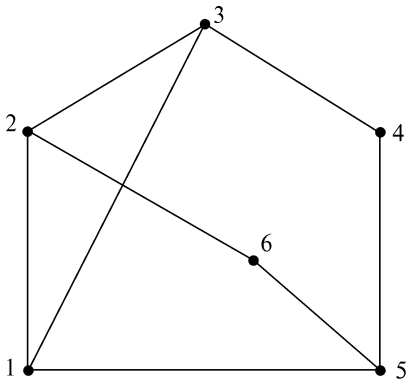

Рис. 3.8. Граф G

Потеря оптимального решения процедурой A имеет место в случае, когда возникает ситуация, при которой весовые характеристики всех булевых функций f_q в процедуре A одинаковы (выбирается любая из них). Однако возникновение такой ситуации возможно в основном на однородных графах, что в задачах планирования ресурсов в РВС встречается довольно редко.

Процедура A может быть применена не только для булевых матриц, которые задают некоторый граф $G(X, E)$, но и для произвольных булевых матриц.

***Пример* 3.1.** Определим минимальное вершинное покрытие в графе, приведенном на рис. 3.8.

Строим булевую функцию графа:

$$f = (x_1 \vee x_2)(x_1 \vee x_3)(x_1 \vee x_5)(x_2 \vee x_3)(x_2 \vee x_6)(x_3 \vee x_4)(x_4 \vee x_5)(x_5 \vee x_6).$$

Определяем, есть ли переменные, которые встречаются один раз. Если да, то умножаем функцию на эту переменную, иначе выписываем все функции, умноженные поочередно на все переменные, присутствующие в ней. После этого вычисляем векторы, характеризующие данные функции, и их весовые характеристики:

$$f_1 = x_1(x_2 \vee x_3)(x_2 \vee x_6)(x_3 \vee x_4)(x_4 \vee x_5)(x_5 \vee x_6);$$
$$h_1 = (2_2, 1_3, 2_4, 2_5, 2_6);\ V_1 = 9;$$

$$f_2 = x_2(x_1 \vee x_3)(x_1 \vee x_5)(x_3 \vee x_4)(x_4 \vee x_5)(x_5 \vee x_6);$$
$$h_2 = (2_1, 2_3, 2_4, 3_5, 1_6);\ V_2 = 10;$$

$$f_3 = x_3(x_1 \vee x_5)(x_2 \vee x_6)(x_4 \vee x_5)(x_5 \vee x_6);$$
$$h_3 = (2_1,2_2,1_4,2_5,2_6);\ V_3 = 9;$$
$$f_4 = x_4(x_1 \vee x_2)(x_1 \vee x_3)(x_1 \vee x_5)(x_2 \vee x_3)(x_2 \vee x_3)(x_5 \vee x_6);$$
$$h_4 = (3_1,3_2,2_3,2_5,2_6);\ V_2 = 12;$$
$$f_5 = x_5(x_1 \vee x_2)(x_1 \vee x_3)(x_2 \vee x_3)(x_2 \vee x_6)(x_3 \vee x_4);$$
$$h_5 = (3_1,2_2,2_3,1_4,1_6);\ V_5 = 9^*;$$
$$f_6 = x_6(x_1 \vee x_2)(x_1 \vee x_3)(x_1 \vee x_5)(x_2 \vee x_3)(x_3 \vee x_4)(x_4 \vee x_5);$$
$$h_6 = (3_1,2_2,3_3,2_4,2_5);\ V_5 = 12.$$

Из полученных таким образом функций выбираем ту, в которой суммарная весовая характеристика вектора минимальна — функцию f_5. Поскольку в выражении для f_5 переменная x_4 встречается только один раз, то, умножая f_5 на переменную, соседнюю с ней — это x_3, получаем

$$f_{53} = x_5x_3(x_1 \vee x_2)(x_2 \vee x_6).$$

Поскольку в выражении для f_{53} переменные x_6 и x_1 встречаются только один раз, то умножаем f_5 на переменную, соседнюю с ней — (x_2), и получаем

$$f_{532} = x_5x_3x_2.$$

Таким образом, минимальное вершинное покрытие графа на рис. 3.8 образуют вершины {2, 3, 5}.

3.4. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМА ОПРЕДЕЛЕНИЯ МИНИМАЛЬНОГО ЧИСЛА КЛАСТЕРОВ НА ОСНОВЕ ЗНВП

Исследование временной сложности разработанного алгоритма проводили для произвольных графов с различными плотностями ребер в графе. Плотность изменялась в диапазоне от 0,1 до 0,5, а количество вершин — в диапазоне от 4 до 100. Графики зависимости количества элементарных операций (ЭО) от количества вершин в графе приведены на рис. 3.9—3.16. Как следует из графиков, временная сложность алгоритма определения минимальных вершинных покрытий в графе в среднем не превышает $O(0,9n^3)$. Все результаты получены с доверительной вероятностью 0,95, при этом погрешность решений не превышала 2—8 %, а процент неточных решений — не более 20 %. Для сравнения использованы ранговый метод и метод исключений [10].

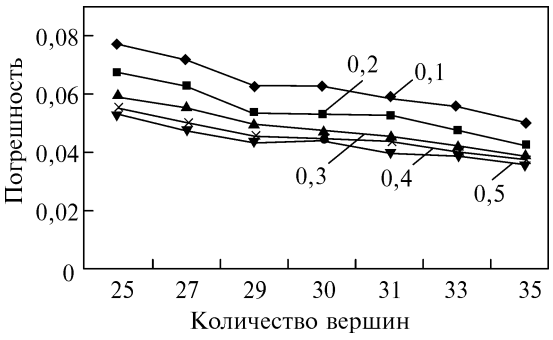

Рис. 3.9. Зависимость погрешности алгоритма от количества вершин в графе при разной плотности ребер в графе

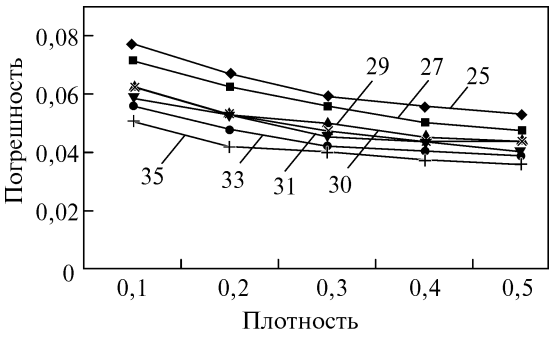

Рис. 3.10. Зависимость погрешности алгоритма от плотности ребер в графе при разном количестве вершин в графе

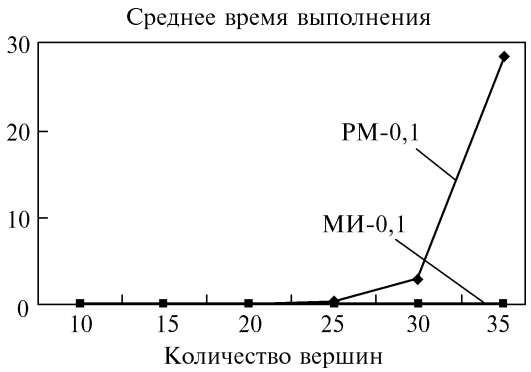

Рис. 3.11. Зависимость среднего времени выполнения алгоритма (РМ — ранговый метод, МИ — метод исключений) от количества вершин в графе

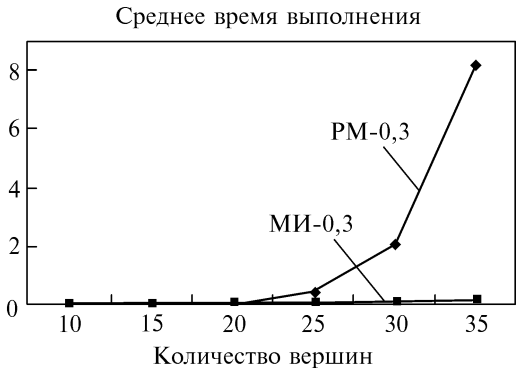

Рис. 3.12. Зависимость среднего времени выполнения алгоритма (РМ — ранговый метод, МИ — метод исключений) от количества вершин в графе

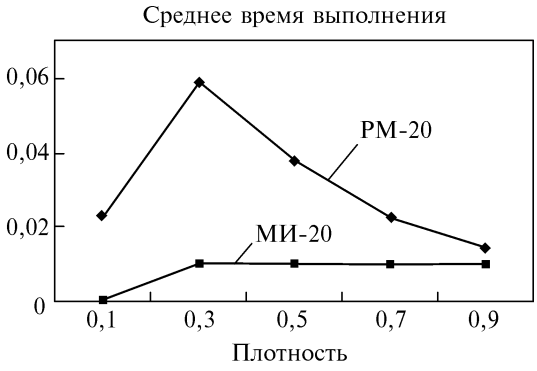

Рис. 3.13. Зависимость среднего времени выполнения алгоритма (РМ — ранговый метод, МИ — метод исключений) от плотности графа

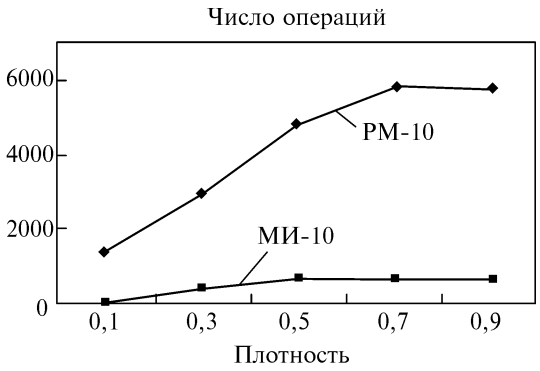

Рис. 3.14. Зависимость числа операций рангового метода (РМ) и метода исключений (МИ) от плотности графа

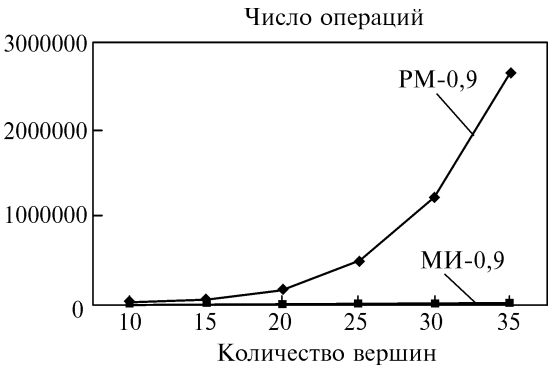

Рис. 3.15. Зависимость числа операций рангового метода (РМ) и метода исключений (МИ) от количества вершин в графе

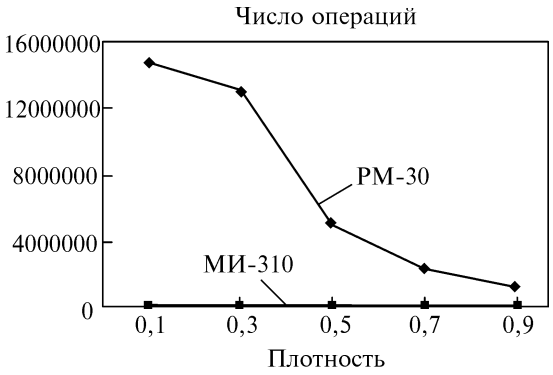

Рис. 3.16. Зависимость числа операций рангового метода (РМ) и метода исключений (МИ) от плотности графа

3.5. ПОСТРОЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ОБОБЩЕННЫХ ПРОЦЕДУР

Прежде чем обсуждать проблему, нужно определиться с тем, что будем вкладывать в понятие «интеллектуальная вычислительная система» [26]. Один из известных подходов заключается в том, чтобы в процессе общения с системой предлагается выяснить, общаетесь ли Вы с человеком или вычислительной системой, и если это не удается установить, то происходит общение с вычислительной системой. Такую систему необходимо определить как интеллектуальную вычислительную систему. Однако такой подход к построению интеллектуальных вычислительных систем не конструктивен. Поэтому поэтапно конкретизируем данное понятие.

Пусть имеется вычислительная система (ВС), в которую введена некоторая теория, — в виде программ. Предположим, что в

рамках данной теории известно некоторое подмножество M истинных высказываний в виде теорем. Если ВС сможет выдать нам новое истинное высказывание $l \in M$, то будем называть такую ВС интеллектуальной, а способность ВС перечислять истинные высказывания $\{l_i\}$, принадлежащие и не принадлежащие M, — интеллектуальной способностью ВС.

ВС, обладающую интеллектуальной способностью, отнесем к интеллектуальным системам 1-го уровня. Предположим, что в ВС вводится некоторое подмножество показателей качества и ВС перечисляет истинные высказывания, удовлетворяющие требуемым значениям введенных показателей качества. Такую ВС будем называть интеллектуальной системой 2-го уровня. Если на основе ВС 2-го уровня построить самовоспроизводящую ВС, то назовем ее интеллектуальной системой 3-го уровня, а если в процессе самовоспроизведения ВС сама выбирает показатель качества и его значение, то такую ВС будем называть интеллектуальной системой 4-го уровня. Рассмотрим возможность построения ВС различных уровней.

В математике истинность высказываний и справедливость теорем основывается на понятии «доказательство». Хотя термин «доказательство» является едва ли не самым главным в математике, оно не имеет точного определения. Понятие «доказательство» во всей своей полноте принадлежит математике не более, чем психологии. Ведь «доказательство» это просто рассуждение, убеждающее нас настолько, что с его помощью мы готовы убеждать других. Тем не менее, если теория не противоречива, а система функционально полна, то метод полной математической индукции и метод рассуждения от противного являются общепринятыми средствами доказательства теорем. Для большей определенности будем полагать, что доказательство — это определение с помощью алгоритма A того, является ли высказывание истинным или ложным. Тогда процесс перечисления истинных высказываний можно рассматривать как процесс построения некоторой теории T, состоящей из некоторого подмножества истинных высказываний, полученных в рамках некоторого алфавита L, на основе которого формируются слова, образующие высказывания. Алфавит L состоит из букв $\{l_i\}$, которые по определенным правилам R^0 объединяются в слова, а слова по правилам R^1 объединяются в высказывания. Система высказываний, объединенная правилами R^2, образует теорию T. Фактически четверка $< A, R^0, R^1, R^2>$ образует дедуктику над алфавитом L, которая эквивалентна тройке $<S, D, \delta>$ в теореме Геделя о противоречи-

вости дедуктики, где S — алфавит доказательств, D — подмножество, элементами которого являются доказательства, δ — функция выделения доказанного. Рассмотрим теоремы Геделя о противоречивости дедуктики.

***Теорема* 3.3.** Если T — перечислимое множество, то для фундаментальной пары $<L,T>$ можно построить полную и непротиворечивую дедуктику.

***Теорема* 3.4.** Если посредством фундаментальной пары $<L,T>$ выразима принадлежность хотя бы к одному не перечислимому множеству натуральных чисел, то для $<L,T>$ не может существовать полной непротиворечивой дедуктики.

Как следует из теоремы 3.4, появление бесконечных множеств является индикатором противоречивости теории. Под противоречивостью понимается наличие утверждений в рамках одной теории, которые нельзя ни доказать, ни опровергнуть. Отметим, что теория бесконечных множеств, основанная Кантором и развитая в дальнейшем в работах Бореля, Шварца, Лебега, Лузина, Урысона, Цермело, Френкеля, Брауэра, Гильберта, изобилует противоречиями и парадоксами. В бесконечных множествах понятия «истинно» и «ложно» нужно использовать очень осторожно. Брауэр более радикален и считает, что в суждениях, полученных с привлечением актуальной бесконечности, понятие «истинно» или «ложно» нельзя использовать.

Из теоремы 3.3 следует, что для конечных объектов, относящихся к конструктивным, дедуктики полны и разрешимы. В работах А.Н. Колмогорова введены комплексы на основе понятия конструктивных объектов. При этом под конструктивным объектом понимаются объекты, о которых можно мыслить, не привлекая абстракции «актуальной бесконечности», т. е. объект может быть предъявлен целиком. Вместо термина «конструктивный объект» А.Н. Колмогоров часто употреблял термин «состояние» алгоритмического процесса. Все конструктивные объекты могут быть заданы конечными графами, что собственно и представляют комплексы Колмогорова.

Формализация задачи

В общем случае конструктивный объект состоит из конечного множества базовых элементов $L = \{l_i\}$ мощности W, из которого на основе правил R^p он может строиться, состоя из r базовых элементов, где $r \le W$. В принципе сам базовый элемент может быть сложно устроен и состоять из $\{l_i\}$ элементов, объеди-

ненных между собой по правилам R^{p-1}. При этом будем полагать, что правила R^p и R^{p-1} и отношения, определенные внутри этих правил, строго определены и не вносят ни каких неопределенностей в процесс формирования конструктивных объектов. Сложность устройства базовых элементов может быть сколь угодно большой, но конечной, т. е. можно выделить первичное множество $\{l_i^0\}$ базовых элементов мощности W_0, из которых по правилам $R = (R^0, R^1, R^2, ..., R^n)$ может быть построен произвольный конструктивный объект. Если для построения конструктивного объекта привлекаются правила $R^0, R^1, R^2, ..., R^k$, то будем считать, что построенный конструктивный объект имеет сложность k. Каждому базовому элементу $\{l_i\}$ поставим в соответствие $(m + 1)$-у весовую характеристику, определяемую по правилам R^v, которые тоже должны быть конечны, строго определены и позволять определять весовые характеристики порожденных конструктивных объектов любой сложности k. Таким образом, конструктивный объект сложности k тоже характеризуется $(m + 1)$-й весовой характеристикой. Одну весовую характеристику будем считать основной, а остальные m — вспомогательными, и в общем случае на них могут налагаться ограничения.

Обозначим множество всех объектов, которое можно построить на основе базовых элементов $\{l_i\}$, через Ω. Тогда подмножество объектов $\Omega^l \in \Omega$, которое можно построить, используя правила R и R^v, удовлетворяющие ограничениям на вспомогательные m характеристик, будем называть подмножеством конструктивных объектов, удовлетворяющих свойству v. Построим граф G^1, вершины которого соответствуют базовым элементам $\{l_i^0\}$, которые соединены в соответствии с правилами R^0. Если рассматривать базовые элементы как буквы некоторого алфавита L, то сами объекты, построенные по правилам R^0, можно рассматривать как слова алфавита L. Далее построив множество всех слов $\{l_i^1\}$ на основе графа G^0 и рассматривая их как новое базовое множество элементов $\{l_i^1\}$, построим, аналогично используя правила R^1, граф G^2, в котором вершинам соответствует множество возможных высказываний $\{l_i^2\}$. Снова используя множество высказываний как базовые элементы и правила R^2, можно построить граф G^3, в котором множества $\{l_i^3\}$ образуют множества теорий. Продолжая процесс построения, можно получить последовательность графов: $G^0, G^1, G^2, G^3, G^4, ..., G^n$, которые соответствуют теориям различной сложности. В данной последовательности графов граф G^0 соответствует алфавиту, граф G^1 —

словарю некоторой предметной области, а графы G^i при $i \geq 2$ — теориям различной сложности.

Построим алгоритм B, позволяющий формировать графы G^i из исходных базовых элементов. В общем случае объект может быть произвольным, но должно быть определено конечное множество элементов $\Omega = (\omega_i)$ или подмножества $L_i \in \Omega$ и правила R, позволяющие формировать объекты из исходных элементов, или подмножеств L_i, принадлежащих множеству Ω.

Пусть задано некоторое разбиение множества Ω на семействе подмножеств $\{L_i\}$ такое, что $\cup L_i = \Omega$ и L_i описывают интересующие нас объекты, состоящие из базовых элементов $\{l_i\}$ таких, что $\cup L_i = \Omega$, и правило R, позволяющее из базовых элементов определять весовые характеристики произвольных объединений $L_k \cup L_p \in \Omega$, характеризующие свойства $\{v\}$. Требуется определить объект со свойством $v^* \in \{v\}$. Представим множество всех возможных объединений подмножеств L_i в виде графа D (см. рис. 3.4) с параллельно ярусной структурой, состоящей из n горизонтальных линеек с вершинами 1, 2, ..., n и n ярусами, каждый из которых содержит все вершины графа D, при этом каждой вершине графа D поставим в соответствие базовый элемент l_i.

Обобщенная процедура A'_0

Шаг 1. Из вершины S строим пути ранга $r = 1$, удовлетворяющие правилам R, и в соответствии с ними определяем их весовые характеристики.

Шаг 2. На основе путей текущего ранга r строим все возможные пути следующего ранга $r = r + 1$, удовлетворяющие правилам R, на основе следующего рекуррентного соотношения:

$$\mu_{SP}^{r+1} = \min_{d_j(m_{sj}^r \cup (j,p))} (\max)\{m_{sj}^r \cup (j,p)\}\ ;\ j = (\overline{1,n})\ ;\ p = (\overline{1,n});\ j \neq p\ ,$$

где весовые характеристики d_l объектов определяются в соответствии с правилами $P \in R$.

Шаг 3. Проверяем подмножества m_{sj}^r текущего ранга r, пусты или нет, если нет, то переходим к выполнению шага 2; иначе процедура заканчивает работу, а из полученных локальных экстремумов на всех рангах выбирается глобальный экстремум.

В зависимости от интересующих свойств объектов и правил R может возникнуть необходимость в работе процедуры A'_0 до

достижения некоторого конкретного значения ранга $r = k$, например, если объектом поиска является клика максимального веса, мощность которой не превышает (не должна превышать) значения k. Отметим, что в общем случае в подмножествах m_{sj}^r ; $j = \overline{(1,n)}$ возможно появление одинаковых путей и дублирующие пути в этих подмножествах можно удалять и, кроме того, удалять их на ярусах.

Представляет интерес выяснить, в каких случаях процедура позволяет получать точное или приближенное решение задачи определения объекта L_j с требуемым свойством ν, связанным с минимизацией или максимизацией весовой характеристики объекта L_j. Для этого более подробно проанализируем процесс перехода от исходного графа $G(V,E)$ с множеством вершин, соответствующих базовым элементам, из которых по правилам R в графе $G(V,E)$ могут быть построены интересующие нас объекты. Особенностью отображения графа $G(V,E)$ в D является то, что если анализируется конкретный объект L_j в $G(V,E)$, то ему в графе D соответствует некоторый путь μ_{sj}^r, построенный на том же множестве вершин, что и объект в $G(V,E)$. При этом для оценки длины пути μ_{sj}^r в D используется весовая характеристика объекта L_j в $G(V,E)$, и множества путей в графах D и $G(V,E)$ совпадают. Предположим, что объектами в $G(V,E)$ являются сами пути графа $G(V,E)$, тогда отображение графа $G(V,E)$ есть отображение на самого себя, и граф D в этом случае, как показано ранее, представляет собой стянутое дерево всех путей данного графа, для прочтения которых на каждой горизонтальной линейке разрешается бывать только один раз. Таким образом, если в этом случае будет решаться задача определения, например, кратчайших гамильтоновых путей или кратчайших гамильтоновых циклов на основе процедуры A_0', то переход к графу D только упорядочивает процесс поиска кратчайших гамильтоновых путей в графе за счет их формирования на основе приведенного рекуррентного соотношения. Нетрудно показать, что в этом случае оптимальное решение будет точным, если процесс работы процедуры непрерывен, т. е. в процессе ее работы либо не возникает ситуаций, когда множества $m_{sj}^r = \varnothing$, либо если такая ситуация возникла, то все пути, ведущие в j в графе $G(V,E)$ рангов $r + 1$, ..., $r = n$, длиннее, чем пути, полученные в j на ранге $r - 1$. Однако легко увидеть, что если множество m_{sj}^r оказалось пустым на некотором

ранге $r = q$, то может оказаться, что в вершину j существует более длинный, чем построенный процедурой A_0', путь $\mu_{sj}^{r=q}$, продление которого в дальнейшем может позволить получить более короткий гамильтонов путь. Данная ситуация фактически является камнем преткновения при решении всех *NP*-полных задач, и в этом случае гарантировано получить точное решение можно только полным перебором или неявным полным перебором на основе метода ветвей и границ, что при достаточно большой размерности искомой задачи одно и тоже. Итак, если рассматривается отображение графа $G(V,E)$ в D и объектами L_j в $G(V,E)$ являются сами пути графа, то переход к анализу на графе D не изменяет данную ситуацию, и мы имеем ту же проблему, что и при решении данной задачи на графе $G(V,E)$ без использования преобразований.

Введем процедуру B преобразования объектов $\{L_j\}$ в графе D, являющуюся частным случаем процедуры A_0', использующей выбор минимальных по длине путей.

Процедура B

Шаг 1. Используя правило R, формируем множество всех объектов $L_j^{r=2}$, состоящих из $r = 2$ базовых элементов с весами $d_j^{r=2}$, и выделяем объект L_j^{*r} с минимальным весом $d_j^{*r=2}$, если таких несколько, то выделяем их все.

Шаг 2. На основе выделенного объекта L_j^{*r} и правила R формируем все возможные объекты $L_j^{r:=r+1}$, состоящие из $r := r + 1$ базовых элементов с весами $d_j^{r=r+1}$, и выделяем объект $L_j^{*r=r+1}$ с минимальным весом $d_j^{*r=r+1}$; если таких несколько, то выделяем все.

Шаг 3. Проверяем, можно ли построить на основе L_j^{*r} и правила R объекты из $r := r + 1$ базовых элементов, если нет, то процедура заканчивает работу, иначе переходим к выполнению шага 2.

В результате работы процедуры B получим объекты $L_j^{*r=2}$, $L_j^{*r=3}$, ..., $L_j^{*r=k}$, содержащие соответственно по 2, по 3 и т. д. по k базовых элементов. Докажем следующее утверждение.

***Утверждение* 3.1.** Если объекты $\{L_j\}$ удовлетворяют свойству B^v, заключающемуся в том, что применение процедуры B к множеству $\{L_j\}$ позволяет построить объекты $L_j^{*r=2}, L_j^{*r=3},..., L_j^{*r=k}$, содержащие соответственно по 2, по 3 и т. д. по k базовых элементов с минимальными весами d_j^{*r1}, то процедура A_0' получает точное решение задачи. Утверждение справедливо, поскольку если предположить, что свойство B^v выполняется, и на основе процедуры A_0' получен глобальный экстремум, который не является точным решением, то это возможно в случае, если существуют объекты $L_j^{**r=2}, L_j^{**r=3},..., L_j^{**r=k}$, длины которых меньше, чем у объектов $L_j^{*r=2}, L_j^{*r=3},..., L_j^{*r=k}$, что противоречит первоначальному предположению о том, что свойство B^v выполняется. Таким образом, если свойство B^v выполняется, то процедура A_0' позволит получить в условиях непрерывности работы процедуры точное решение, а если не выполняется, то — приближенное, поскольку в этом случае мы фактически сталкиваемся с решением *NP*-полных задач. Отметим, что в случае выполнения свойства B^v наличие пустых множеств m_{sj}^r при работе процедуры A_0' означает, что на основе правил R построить объект $L_j^{r:=r}$, содержащий базовый элемент j, или нельзя в принципе (последнее нужно обосновывать для каждой задачи в отдельности), или же, что строить данный элемент не имеет смысла, поскольку его весовая характеристика будет существенно хуже тех, которые построены на ярусе.

Таким образом, непрерывность реализации процедуры B является необходимым и достаточным условием для получения точного решения. Для случая, когда объекты $\{L_j\}$ характеризуются одной весовой характеристикой, процедуру A_0' можно упростить за счет выделения на ярусах глобальных экстремумов и последующего формирования всех возможных путей следующего ранга в графе D.

Для этого введем процедуру A_0''.

Процедура A''_0

Шаг 1. Из вершины S строим пути ранга $r = 1$, удовлетворяющие правилам R, и в соответствии с правилами R определяем их весовые характеристики.

Шаг 2. На основе путей текущего ранга r строим все возможные пути следующего ранга $r = r + 1$, удовлетворяющие правилам R, на основе следующего рекуррентного соотношения:

$$\mu_{SP}^{r:=r+1} = \min_{d_j(m_{sj}^r \cup (j,p))} (\max)\{m_{sj}^r \cup (j,p)\}\ ;\ j = (\overline{1,n})\ ;\ p = (\overline{1,n});\ j \neq p\ ,$$

где весовые характеристики d_l объектов определяются в соответствии с правилами $P \in R$, среди которых на ярусе выделяется путь μ_{sj}^{*r} максимальной или минимальной на ярусе длины и на его основе формируются все возможные пути ранга $r := r + 1$ в графе D.

Шаг 3. Проверяем подмножества m_{sj}^r текущего ранга r — они пусты или нет; если нет, то переходим к выполнению шага 2; иначе процедура заканчивает работу, и из полученных на последнем ранге путей выбирается лучший (максимальный или минимальный) по длине путь.

Оценки временной сложности работы обобщенных процедур при использовании многопроцессорных вычислительных систем

В процедуре A_0, представляющей процедуру перечисления всех возможных путей, которые можно породить в графе D, используя правила R, и при этом, если само правило R предполагает наличие экспоненциального количества формируемых объектов, то число путей в множествах m_{sj}^r будет от ранга $r = 1$ до ранга $r = n - 1$ возрастать экспоненциально. В случае использования процедур A_0' и A_0'' возможно два варианта: первый, когда число путей во множествах возрастает экспоненциально, и процедура A_0' реализуется в виде процедуры B, в которой в каждом подмножестве может выделяться несколько одинаковых экстремальных путей. Второй — если в каждом подмножестве m_{sj}^r в процессе работы процедуры A_0' выделяется только один (произвольный) экстремальный путь, то число путей, которые строятся процедурой на каждом ярусе и общее их количество — полиномиально. Поскольку на каждом ярусе в любом подмножестве m_{sj}^r процедура A_0' в случае выделения только одного экстремального пути строит не более $(n - 1)$ пути, то на ярусе соответственно не более $(n - 1)n \approx n^2$ путей. Так как число ярусов в графе D не может превышать $(n - 1)$, то общее число путей, которое построит

процедура A_0', дойдя до последнего яруса, не превысит $(n-1)n(n-1) \approx n^3$ путей. В случае использования процедуры A_0'' количество формируемых путей еще больше уменьшается, поскольку в этом случае в каждом подмножестве m_{sj}^r формируется только по одному пути, на ярусе формируется по $(n-1)$ путь, и общее число построенных путей процедурой A_0'' по достижению яруса $(n-1)$ не превысит числа $(n-1)n \approx n^2$. Отметим, что при использовании процедур A_0' и A_0'' в конкретных алгоритмах для определения временной сложности их реализации необходимо учитывать время, затрачиваемое алгоритмом на определения весовых характеристик путей и удовлетворение правилам R, определяемым свойствами формируемых объектов.

Обозначим число операций, необходимых для вычисления длин путей и проверки правил R, через (p), тогда временная сложность алгоритмов на основе данных процедур не превысит соответственно $O(pn^3)$ и $O(pn^2)$. Следует отметить, что в соответствии с утверждением 3.1 алгоритмы, построенные на основе данных процедур A_0' и A_0'', будут давать приближенное решение. Поэтому для каждой конкретной задачи должна быть дана оценка погрешности в среднем.

Поскольку формирование путей на ярусах в множества m_{sj}^r можно осуществить, используя технологии распараллеливания трудоемких вычислений, то для n процессорных элементов в случае их реализации на многоядерной архитектуре или вычислительном кластере временная сложность алгоритмов на основе рассматриваемых процедур не превысит соответственно $O(pn^2)$ и $O(pn)$.

Введенные в параграфе 3.1 обобщенные процедуры A_0, A', A'' для формирования путей в графе D позволяют перечислять все объекты множества $\{L_j\}$ с интересующим нас свойством $v^* \in \{v\}$. В зависимости от интересующих свойств объектов и правил R может возникнуть необходимость в работе процедуры A' до достижения некоторого конкретного значения ранга $r = k$. Используя процедуры A_0, можно построить алгоритм B' порождения графов последовательности G^0, G^1, G^2, G^3, G^4 ,..., G^n.

Алгоритм B′

Шаг 1. На основе процедуры A_0 перечисляем все возможные объекты и используя правила R^0 удаляем из полученного множества объекты, не являющиеся словами в рассматриваемом языке,

и далее, используя правила $R_n \in R$, можно удалить и слова, которые не принадлежат той предметной области, для которой будет строиться теория. Таким образом, получим словарь Q^n, состоящий из множества слов $\{q_i\}$.

Шаг 2. Устанавливаем взаимно однозначное соответствие между множеством вершин графа D и множеством слов $\{q_i\}$, для чего следует перенумеровать все слова в Q^n и первой вершине графа D поставить в соответствие слово q_1, второй вершине графа D — слово q_2 и т. д.

Шаг 3. Используя процедуру A', строим все возможные высказывания удовлетворяющие интересующему нас свойству v, т. е. получаем теорию, обозначенную в последовательности как G^2.

Понятно, что используя алгоритм B', можно построить теорию любой сложности k, если существуют правила R^k, которые можно использовать для выделения базовых высказываний, необходимых для создания теории данной сложности k. Например, если существуют правила R^3 для выделения базовых высказываний в теории G^2, то установив взаимно однозначное соответствие между базовыми высказываниями теории G^2 и вершинами графа D и применив к ним процедуру, получим теорию G^4.

Таким образом, на основе алгоритма B' вычислительная система может построить непротиворечивую теорию, позволяющую в рамках рассматриваемой предметной области перечислять истинные высказывания. Мы предположили, что для определения истинности высказываний в синтезируемых теориях G^i существует алгоритм A. Множеству слов U, полученных из начального алфавита $\{l_i^0\}$, можно поставить множество векторов $\{X_1, X_2, ..., X_n\}$, в которых $X_i = 1$, если буквы из $\{l_i^0\}$ присутствуют в слове $u_p \in U$ и объединены в слово в соответствии с правилами объединения R^0, и $X_i = 0$ в противном случае. Тогда для описания всего множества слов можно использовать функцию (3.19).

В случае использования правил R^0 коэффициенты C_{rj} могут принимать значения, равные 1 и 0. Если $C_{rj} = 0$, то слово u_p в виде j-й перестановки из сочетания букв и правила R^0, не может быть построено в алфавите $\{l_i^0\}$, и $C_{rj} = 1$ — в противном случае.

Таким образом, множество всех наборов переменных, на которых $F(x)$ принимает значение «истинно», будет словом алфавита $\{l_i^0\}$, а наборы переменных, на которых $F(x)$ принимает значение «ложно», соответствуют словам, не принадлежащим алфавиту. Аналогичным образом с помощью булевой функции $F(x)$ можно описать все истинные высказывания при выполнении

правил R^1. Известно, что проблема определения истинности высказываний в теории формальных языков может быть сведена к решению задач «выполнимость» [10]. Для любого двоичного набора функция принимает одно из двух возможных значений: единицу или нуль. Задача «выполнимость» заключается в ответе на вопрос: существует ли набор значений переменных, обращающий функцию f в единицу?

Произвольная задача «выполнимость» может рассматриваться как задача k-выполнимость на наборе из k переменных, где k определяется дизъюнктом с максимальным числом переменных.

Пусть имеется алгоритм A, позволяющий определять истинность высказываний в рамках некоторого алфавита L и, следовательно, с помощью процедур B и A можно построить непротиворечивую теорию, при этом наращивать сложность теории мы можем только в рамках алфавита базовых элементов. Возникает вопрос: если имеется две непротиворечивые T^1 и T^2 теории, каждая со своим алфавитом, можно ли создать новую теорию T^3, объединяющую обе, т. е. такую, чтобы T^1 и T^2 были частными случаями T^4?

Естественно рассматривать возможность объединения только тех теорий, которые имеют общие базовые элементы алфавита. Если $L^1 \cap L^2 = \varnothing$, то такие теории существуют независимо, и в рамках этих теорий не может существовать высказываний, которые справедливы в рамках одной теории T^1 и не справедливы в рамках другой теории T^2, поскольку предметная область у них разная (например, теория кодирования и теория построения механизмов). Таким образом, именно пересечение алфавитов $L^1 \cap \cap L^2$, с одной стороны, дает принципиальную возможность объединять теории, и это же пересечение дает возможность для возникновения противоречивых высказываний. Например, в геометрии Евклида справедливо высказывание: «параллельные прямые не пересекаются», а в геометрии Лобачевского справедливо противоположное высказывание: «параллельные прямые пересекаются». Введение определенных правил R позволяет объединить эти две теории и показать, что они непротиворечивы и что геометрия Евклида является частным случаем геометрии Лобачевского. Таким образом, $|L^1 \cap L^2|$ можно использовать как оценку сверху возможной противоречивости теории T^3, т. е. числа высказываний, для которых необходимо ввести правила R, согласующие эти две теории. Правила R будем называть согласующими для теорий T^1 и T^2, если их введение позволяет ис-

ключить все противоречивые высказывания в рамках объединенной теории T^4. Относительно алфавита L^1 теория T^1 не противоречива и, соответственно, относительно алфавита L^2 теория T^2 тоже не противоречива. Выделим высказывание $l \in L^1 \cap L^2$ о возможности построения некоторого объекта P, которое истинно в T^1 и ложно в T^2. Тогда если мы попытаемся проанализировать возможность существования объекта P в рамках алфавита $L^1 \cap \cap L^2$, используя метод рассуждения от противного в рамках обеих теорий, то можно получить высказывание вида: объект P существует (т. е. может быть построен), если P не существует (т. е. не может быть построен) и, вследствие противоречивости этого высказывания будет сделан вывод о том, что объект P не существует. Таким образом, для множества частных случаев, которые никак не определяются, а рассматриваются как опыт построения программ, следует обобщение о том, что проблемы «останова» для них не существует, т. е. мы пришли к противоречию. Возникает вопрос: высказывание «проблема “останова” неразрешима» истинно или ложно? Для того чтобы однозначно ответить на этот вопрос, следует признать, что понятие истинности или ложности относится к исходным данным, т. е. к исходному базовому алфавиту. По всей видимости, нельзя говорить об абсолютности истинности или ложности высказываний в отрыве от входных данных базового алфавита и способа их кодировки, в рамках которых данное высказывание определено. Так, высказывание l о возможности построения объекта P относительно алфавита L^1 истинно, а относительно алфавита $L^1 \cap L^2$ оно не определено. Если с помощью правил R^3 (по умолчанию предполагая, что они не выводят за рамки понятий конструктивных объектов) доопределим все противоречивые высказывания в виде аксиом, принимаемых в теориях T^1 и T^2 как истинные, то тогда получим непротиворечивую теорию T^3 в конечном алфавите $L^1 \cap L^2$. Вопрос о существовании таких правил зависит от свойств базовых объектов, определяющих исходный алфавит, и, следовательно, выходит за рамки формальной разработки самих теорий.

Таким образом, рассмотрены два подхода к созданию непротиворечивых теорий: один, основанный на постепенном усложнении теории в соответствии с последовательностью G^0, G^1, G^2, G^3, G^4, ..., G^n, построенной в рамках алфавита заданных базовых элементов на основе предложенного алгоритма B, и второй, связанный с возможностью объединения непротиворечивых тео-

рий, имеющих общее подмножество базовых элементов алфавита и требующий введения правил согласования, которые устраняют противоречия. А сами правила выходят за рамки процесса формализации построения самой теории, поскольку определяются свойствами базовых элементов, являющихся исходными данными для формирования самой теории. При этом возникает довольно сложный вопрос алгоритмической разрешимости проблемы, которая, как показано, является относительной по отношению к исходным данным. Оба подхода имеют право на существование, но второй приемлем для создания метатеорий и не может быть автоматизирован полностью, поскольку на этапе создания правил, устраняющих противоречия, необходимо вмешательство человека.

Первый подход фактически полностью формализован, и вычислительная система будет только перечислять истинные высказывания, а человек будет являться их пользователем. На основе первого подхода для каждой предметной области можно легко построить язык и словарь терминов и понятий, удобный для пользователя. При этом создаваемые языки будут такими, какие мы используем в повседневной жизни, но несколько ограниченные в возможностях формирования высказываний в рамках предметной области. Сам процесс создания таких языков может быть автоматизирован за счет разработки соответствующего программного обеспечения на основе имеющегося опыта построения языков программирования. С его помощью легко корректировать и расширять большой объем информации в зависимости от потребностей пользователя и, в общем случае, нет необходимости в разработке новых технологий построения вычислительных систем.

ВЫВОДЫ К ГЛАВЕ 3

1. Рассмотрены математические модели общих и частных задач планирования распределенных вычислений. Показано, что они сводятся к задачам дискретной оптимизации в виде задач линейного и нелинейного программирования.

2. Предложен и обоснован общий подход к решению задач линейного и нелинейного программирования с булевыми переменными.

3. Для решения задачи минимизации количества кластеров в РВС предложена модель ее решения в виде задачи о наимень-

шем вершинном покрытии. Предложен метод ее решения, позволяющий получать решения с малой трудоемкостью и низкой погрешностью. Временная сложность данного алгоритма может быть уменьшена с помощью использования n-процессорной вычислительной системы, позволяющей в разработанных алгоритмах параллельно вычислять вычисления на ярусах.

4. Приведены результаты вычислительных экспериментов, подтверждающие эффективность предложенных методов для планирования заданий в РВС.

5. Предложен подход к построению интеллектуальных вычислительных систем, использующий идеи рангового метода для решения задач дискретной оптимизации.

6. Разработаны обобщенные процедуры, позволяющие на основе единого подхода решать задачи оптимизации с различными критериями качества. При этом можно определять экстремум некоторых совокупностей свойств объектов, в частности, решать задачи минимизации и максимизации выбранной функции цели.

7. Важнейшим направлением в повышении производительности решения трудоемких задач (*NP*-полных задач) является использование информационно-коммуникационных технологий. К современным трендам следует отнести использование высокопроизводительных кластерных фреймворков, которые могут быть развернуты в локальном режиме и с использованием облачных платформ. В настоящее время наиболее распространенными задачами, решаемыми на этих платформах, являются задачи искусственного интеллекта и машинного обучения. К используемым при этом системам следует отнести такие системы как ApacheSpark, ApacheFlink, ApacheHadoop, позволяющие повысить производительность решаемых задач в десятки раз. Кроме этого, применение указанных платформ позволяет реализовывать технологии параллельных вычислений и распределенных баз данных, что обусловливает повышение производительности вычислений.

МЕТОД МОНИТОРИНГА СОСТОЯНИЯ РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ НА ОСНОВЕ ОПРЕДЕЛЕНИЯ КРАТЧАЙШИХ ПУТЕЙ И КРАТЧАЙШИХ ГАМИЛЬТОНОВЫХ ЦИКЛОВ В ГРАФЕ

Решение задач мониторинга распределенных вычислительных систем является актуальной с точки зрения обеспечения требуемой при обработке потоков заданий производительности и пропускной способности.

Важнейшие задачи управления распределенными вычислениями — это задачи планирования заданий и выбора (назначения) требуемых вычислительных ресурсов. Формальные методы управления планированием заданий на основе задач линейного программирования с булевыми переменными рассмотрены в работах [1—10]. В процессе решения задач планирования, как правило, все локальные планировщики и системы управления локальными ресурсами используют компоненты систем мониторинга, для применения которых в РВС разработан метод [10, 11], в основу которого положена модель графика (расписания) мониторинга узлов распределенной вычислительной системы, использующего кратчайшие гамильтоновы пути и гамильтоновы циклы для оптимизации процедур запуска программных расширений, устанавливаемых на узлах РВС.

Наиболее распространенные системы мониторинга Nagios [12], Icinga [13] используют программные расширения (агенты), устанавливаемые на объектах мониторинга для их удаленного запуска и реализующие различные сервисы для решения задач мониторинга. Для реализации сервисов на узлах РВС требуется установить плагин NPRE (Nagios Remote Plugin Executor), инициализирующий работу программных агентов. Процедура работы вызываемого удаленного сервиса включает: инициализацию командой запуска, осуществляемой агентами NRPE на серверах и узлах РВС; запуск и выполнение сервиса; получение результатов

работы сервиса; передачу полученных данных на управляющий узел (базу данных). Для комплексной оценки состояния программно-аппаратных средств, коммуникационных компонентов и выполняемых заданий РВС следует использовать сервисы удаленного доступа, непосредственно связанные с решением задач планирования распределенных вычислений. К ним относятся [14, 15]: доступность и уровень загрузки процессорных узлов кластеров; доступность и пропускная способность коммуникационных каналов (включая коммуникационные каналы кластеров и сетей передачи данных); количество доступных и свободных узлов кластеров РВС; состояние выполняемых заданий на узлах кластеров; доступность, текущая производительность и загрузка узлов, используемых для хранения данных системы мониторинга. При наличии большого количества необходимых для проведения мониторинга объектов РВС резко увеличивается нагрузка на коммуникационную сеть, ограниченную ее пропускной способностью. Информация о состоянии объектов мониторинга является фоновой относительно основной — пользовательской и служебной (управляющей) информации, объемы которой непосредственно влияют на качество обслуживания пользователей — время реализации запросов и приложений, суммарное запаздывание, стоимость вычислений и т. д.

Для минимизации времени задержки на инициализацию запуска и работу удаленных агентов предлагаем применять метод оптимизации последовательности опроса узлов РВС, использующий представление РВС в виде неполносвязного графа, в котором требуется определить минимальную временную задержку опроса необходимого количества мониторинговых устройств путем решения задач нахождения кратчайшего пути и кратчайшего гамильтонова цикла.

4.1. ПОСТАНОВКА ПРОБЛЕМЫ

Модель реализации множеств приведенных сервисов Rem_Serv можно представить в следующем виде:

$$Rem\_Serv : AN \times AU \times ANet \times AJ \times ADB \rightarrow \{State_1, State_2, State_3, State_4\},$$

где AN — множество сервисов определения доступности узлов; AU — множество сервисов определения загрузки (использования) узлов; $ANet$ — множество сервисов определения состояния

коммуникационных каналов; AJ — множество сервисов определения состояния выполняемых заданий; ADB — множество сервисов определения состояния БД; $State_i$, $i = \overline{1,4}$ — множество состояний объекта мониторинга, определяемых переменными $\{OK, WARNING, CRITICAL, UNKNOWN\}$.

Учитывая, что мощности множеств рассмотренных сервисов для мониторинга составляют [12]: $|AN| = 2$, $|AU| = 3$, $|ANet| = 3$, $|AJ| = 4$, $|ADB| = 6$, а контролировать системному администратору и пользователям виртуальных организаций РВС необходимо, используя, например, только два состояния объектов мониторинга — $WARNING$ и $CRITICAL$, минимальное количество сообщений, получаемых от участвующих только в одном опросе удаленных агентов, составит 2^{18}. Учитывая, что размер одного сообщения равен порядка 4 Кб, такой объем служебной информации приводит к значительной загрузке коммуникационных каналов и необходимости минимизации временных задержек работы удаленных сервисов для мониторинга вычислительных кластеров и узлов РВС.

Для получения математической модели с целью оптимизации последовательности опроса объектов мониторинга РВС используем схему на рис. 4.1, на котором показана технология работы сервисов удаленного доступа. На рис. 4.1 использованы следующие обозначения: R_i, $i = \overline{1,m}$ — ресурсы (вычислительные кластеры) РВС; S_i, $i = \overline{1,m}$ — серверы ресурсов; R_{ij}, $i = \overline{1,m}$; $j = \overline{1,N_m}$ — узлы кластеров; M_i — маршрутизаторы; УУ — управляющий узел, с которого инициируется запуск удаленных агентов.

Обозначим время передачи запроса на запуск удаленного агента как t_{ij}, определяемое по формуле:

$$t_{ij} = \frac{l_{ij}}{p_{ij}} x_{ij},$$

где l_{ij} — размер инициализирующего пакета, передаваемого между ресурсами R_i и R_j, байт; p_{ij} — пропускная способность коммуникационного канала между ресурсами R_i и R_j (байт/с); $x_{ij} = 1$, если ресурсы R_i и R_j соединены коммуникационным каналом, 0 — в противном случае.

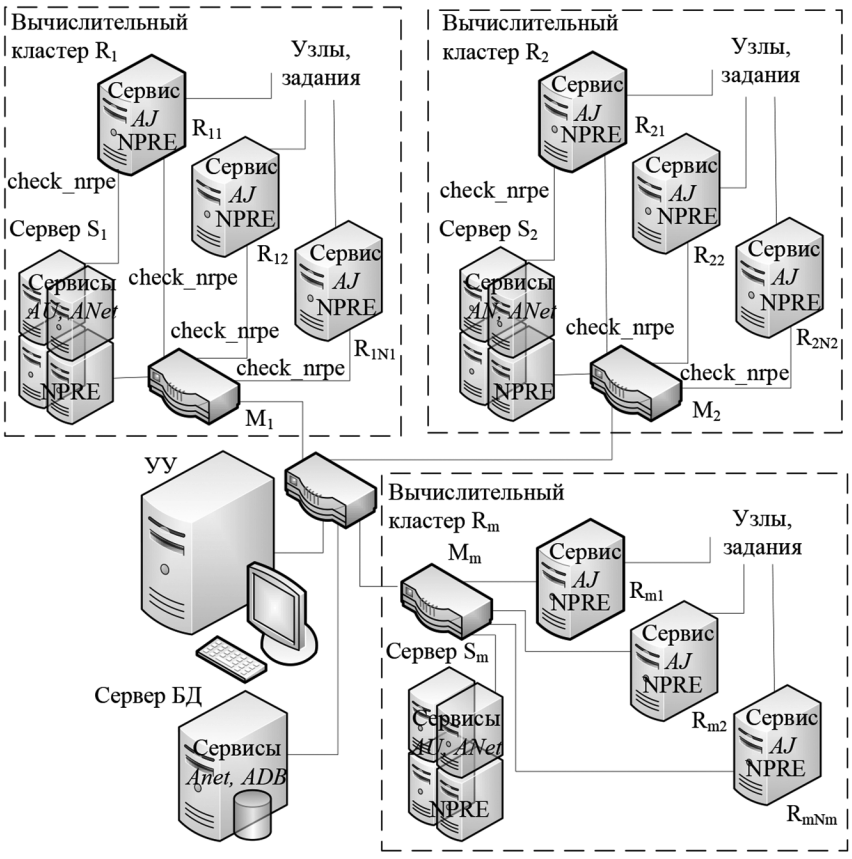

Рис. 4.1. Схема работы сервисов удаленного доступа в системе мониторинга РВС

Тогда временная задержка пакетов для запуска агентов на всех кластерах РВС составит

$$T_{delay}^{1} = \sum_{i=1}^{m} \sum_{j=1}^{m} t_{ij} .$$

После запуска команды check_nrpe на серверах S_i происходят инициализация и запуск удаленных сервисов на узлах вычислительных кластеров РВС. Временная задержка, определяющая запуск сервисов на узлах кластера, составит

$$T_{delay\_Node} = \sum_{k=1}^{Rem\_Serv} \sum_{l=1}^{Nm} t_{kl} .$$

Временная задержка, определяющая запуск агентов на всех кластерах РВС, определяется по формуле:

$$T^2_{delay} = \sum_{i=1}^{m} \sum_{k=1}^{Rem\_Serv} \sum_{l=1}^{Nm} t_{ikl}.$$

Таким образом, общая временная задержка запуска удаленных агентов для мониторинга вычислительных кластеров РВС составляет

$$T_{delay} = \sum_{i=1}^{m} \sum_{j=1}^{m} t_{ij} + \sum_{i=1}^{m} \sum_{k=1}^{Rem\_Serv} \sum_{l=1}^{Nm} t_{ikl}.$$

Учитывая, что топология коммуникационных каналов РВС не является полносвязной, т. е. не все ресурсы РВС соединены между собой, требуется минимизировать суммарную временную задержку: $T_{delay} \to \min$. При этом предполагается, что на интервалах планирования пропускная способность коммуникационных сетей между вычислительными кластерами РВС является постоянной.

4.2. МЕТОД РЕШЕНИЯ

Задачу минимизации суммарной временной задержки запуска удаленных агентов на узлах РВС предлагается решить в два этапа.

Этап 1. Построить для всех опрашиваемых узлов РВС дерево кратчайших путей к остальным узлам сети, которую будем представлять в виде графа G с вершинами, соответствующими узлам РВС, и ребрами — коммуникационными каналами между узлами. При этом длины этих ребер будут определять временную задержку передачи данных по каналам связи, объединяющим выделенные узлы в единую сеть, а длины путей в дереве кратчайших путей — минимальное время доставки запросов из некоторого узла i в узел j данной сети.

Этап 2. Построить полный граф из опрашиваемых узлов РВС, в котором ребра имеют веса t_{ij}, определяющие минимальную задержку передачи запроса от некоторого узла i к узлу j рассматриваемой сети, равную длине кратчайшего пути в дереве кратчайших путей, построенном на этапе 1. Далее для данного графа определить порядок обхода всех этих узлов за минимальное время, т. е. определить кратчайший гамильтонов цикл.

Поскольку эти задачи должны решаться в реальном времени, для реализации указанных алгоритмов на этапе 1 предлагается разработать параллельную реализацию алгоритма Дейкстры на основе рангового подхода [16], а на этапе 2 решить задачу определения кратчайшего обхода всех узлов сети на основе решения системы квадратичных уравнений.

При решении задачи определения путей возможны случаи, когда граф, к которому сведена исходная задача, имеет положительные и отрицательные веса ребер. В литературе выделяют следующие случаи: в графе все ребра имеют положительные веса; в графе имеются ребра с отрицательными весами, но нет циклов отрицательной длины; в графе имеются циклы отрицательной длины. При решении этих задач используются следующие методы: *индексные* (пометок), *матричные* и *аппаратные*. Все эти методы применяют для решения двух типов задач: нахождения кратчайшего пути между заданной парой вершин и кратчайших путей между всеми парами вершин в графе. Для решения задачи поиска кратчайшего пути между парой вершин лучшими являются индексные и аппаратные методы, для решения задач определения кратчайших путей между всеми парами вершин в графе — матричные методы. Одним из наиболее эффективных алгоритмов определения кратчайших путей в графе с положительными весами ребер является алгоритм Дейкстры, использование которого в графах с отрицательными весами ребер приводит к экспоненциальному росту временной сложности [16]. В случае, когда в графе есть циклы отрицательной длины, задача относится к классу *NP*-полных задач. Количество публикаций по методам определения кратчайших путей сейчас достаточно велико [17].

Задача поиска гамильтонова цикла минимального веса в полном графе с целыми неотрицательными весами ребер известна как *задача коммивояжера* (ЗК). К методам ее решения следует отнести точные алгоритмы, полученные в работах [18, 19]. В них для решения ЗК предложен алгоритм динамического программирования, имеющий временную сложность $O(2^n)$ и использующий память M, где M — максимальный вес ребра. Лучший алгоритм с полиномиальной памятью с временной сложностью $O(4^n n^{\log n})$ приведен в работах [20, 21]. В работе [22] показано, что ЗК может быть решена за время $O(2^{2n-t} n^{\log(n-t)})$ с памятью $O(2^t)$ для любого $t = n, n/2, n/4, \ldots$ В работах [23—26] предложен алгоритм для решения ЗК с временной сложностью $O(2^n)$ и памятью $O(M)$. Остается открытым вопрос о существовании алгоритма с временной сложностью $O(2^n) = 2^n \,\mathrm{poly}(n; \log M)$ и полиномиальной

памятью [27]. В работе [28] предложен алгоритм для решения ЗК в кубических графах со временем работы 1.260^n и для графов степени 4 со временем работы 1.890^n. Авторы работы [29] улучшили оценку для кубических графов до 1.51^n, в [30] разработан алгоритм для графов степени 4 со временем работы 1.733^n и экспоненциальной памятью. Актуальность использования ЗК в различных приложениях подтверждается разработанными в последнее время *эвристическими алгоритмами* — муравьиным алгоритмом [31, 32] и его модификацией [33, 34] с временной сложностью в худшем случае $O(n^3)$. В работе [35] предложен точный алгоритм решения ЗК с временной сложностью в интервале ($O(n^4)$, $O(n^3 2n)$). В работе [36] для оптимизации стоимости перевозок предложен метод решения транспортной задачи в вероятностной постановке, интерпретируемый как задача минимизация стоимости доставки сообщений в сетях с различной пропускной способностью.

Таким образом, проведенный анализ показывает, что получение точных решений ЗК связано с большими временными затратами. В связи с этим целью этапа 2 является разработка метода решения ЗК с малой временной сложностью и малой погрешностью для использования его в системе мониторинга РВС в режиме реального времени.

4.3. МЕТОД РЕШЕНИЯ ЗАДАЧИ ОПРЕДЕЛЕНИЯ КРАТЧАЙШЕГО ПУТИ НА ОСНОВЕ РАНГОВОГО ПОДХОДА

Для решения поставленной задачи используем представление исходного графа G в виде симметричного дерева путей [37, 38]. Пусть все возможные состояния некоторой системы определяются графом $G(V, E)$ с n вершинами, где вершины соответствуют возможным состояниям системы. Перейдем к пространству с $(n-1)^2$ состояниями. Для этого каждому из n состояний поставим в соответствие еще $(n-1)$ состояние, характеризующее способ достижения состояния из множества $\{1, 2,..., n\}$. При этом в качестве добавляемых состояний определим ранг пути в графе $G(V, E)$ — из вершины s графа $G(V, E)$ в произвольную вершину j можно попасть:

- путем ранга $r = 1$, используя одно ребро,
- путем ранга $r = 2$, используя два ребра и т. д.,
- путем ранга $r = n - 1$, используя $n - 1$ ребро.

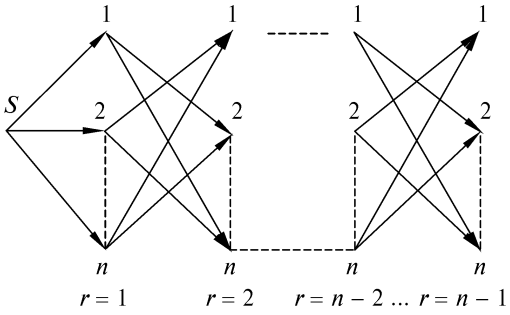

Рис. 4.2. Стянутое дерево всех путей D графа $G(V, E)$

Такое пространство состояний можно представить в виде стянутого дерева путей D (рис. 4.2).

Исходя из стянутого дерева путей, для произвольной вершины j множество путей, ведущих в эту вершину из некоторой вершины s, можно представить в следующем виде:

$$m_s(j) = m_{sj}^{r=1} \cup m_{sj}^{r=2} \cup .... \cup m_{sj}^{r=n-1};\ j = (\overline{1,n-1}),$$

где $m_{sj}^r = \{\mu_{sj}^r\}$ — подмножества путей из произвольной вершины s в некоторую вершину j графа $G(V, E)$ ранга r. Отметим, что дерево всех путей D может строиться и от конкретной вершины i графа: в этом случае вершина $s = i$ и i-я горизонтальная линейка исключаются в D. Таким образом, используя граф D и введя правила формирования путей следующего ранга, можно из произвольной вершины s поэтапно строить пути $\{\mu_{sj}^r\}$ произвольного ранга вплоть до ранга $r = n - 1$. Здесь рассматривается взвешенный граф, ребрам которого присвоены веса l_{ij}, где i, j — смежные вершины, которые задаются матрицей длин $L = \{l_{ij}\}$. Если $\mu_{st} = (s, v_1, v_2, ..., v_k, t)$ — некоторый путь из вершины s в t, то его длина определяется как сумма длин ребер, образующих данный путь. Величину длины пути между вершинами i и j обозначим $d(i, j)$, величину кратчайшего пути между вершинами i и j — $P(i, j)$.

Как показано в работе [16], данная задача может интерпретироваться как задача линейного программирования на основе единичного потока минимальной стоимости. В терминах математического программирования она формулируется следующим образом:

необходимо минимизировать

$$\sum_{i,j} l_{ij} X_{ij} \qquad (4.1)$$

при условии

$$\sum_i X_{ij} - \sum_i X_{ji} = \begin{cases} 1, & \text{если } i = s; \\ -1, & \text{если } i = t; \\ 0, & \text{если } i \neq s, t; \end{cases} \tag{4.2}$$

$$X_{ij} \in \{0,1\}. \tag{4.3}$$

Требуется определить поток $\{X_{ij}\}$, минимизирующий функционал (4.1). Если ребро (i, j) принадлежит кратчайшему пути, то $X_{ij} = 1$, в противном случае $X_{ij} = 0$. Значения функционала определяет $P(s, t)$. Ограничения (4.2), (4.3) являются условиями непрерывности пути. Задачу определения кратчайшего пути предлагается решить с помощью рангового метода, использующего переход представления в виде графа G к стянутому дереву всех путей D.

Рассмотрим некоторый произвольный граф $G(V, E)$ с множеством вершин V и множеством ребер E, мощность которого не превышает значения $n(n - 1)/ 2$. Если в графе выделить свободные вершины s и t, то множество всех путей $\{\mu_{st}\}$ между этой парой вершин можно представить в виде объединения подмножеств:

$$M_{st} = M_{st}^{r=1} \cup M_{st}^{r=2} \ldots \cup M_{st}^{r=n-1}, \tag{4.4}$$

где M_{st}^{r} — множества путей между вершинами s и t ранга r.

Пути ранга $r = n - 1$ — пути с максимально возможным рангом — являются гамильтоновыми путями, проходящими через все вершины графа G. Таким образом, задача определения кратчайшего пути между заданной парой вершин s и t сформулирована как задача определения пути минимальной длины некоторого ранга r в множестве M_{st}:

$$\min M_{st} = \min \left\{ M_{st}^{r=1}, M_{st}^{r=2}, \ldots, M_{st}^{r=n-1} \right\}. \tag{4.5}$$

Из (4.5) следует, что если в каждом из подмножеств M_{st}^{r}, $r = \overline{(1, n-1)}$ определить кратчайший путь ранга r и обозначить полученное множество как M_{st}^{*r}, $r = \overline{(1, n-1)}$, то задача определения кратчайшего пути сводится к задаче определения кратчайшего пути в множестве M_{st}^{*r}:

$$\min \left\{ M_{st}^{*r=1}, M_{st}^{*r=2}, \ldots, M_{st}^{*r=n-1} \right\}. \tag{4.6}$$

Пусть некоторая процедура A позволяет строить кратчайшие пути между вершинами s и t произвольного ранга. Тогда в соот-

ветствии с (4.5), (4.6) требуется определить кратчайший путь между вершинами s и t.

В случае определения кратчайших путей между вершиной s и всеми остальными вершинами графа G задача формулируется следующим образом: необходимо определить $\min \mu_{si}^{r} = \min\left\{\mu_{si}^{r=1}, \mu_{si}^{r=2}, \ldots, \mu_{si}^{r=n-1}\right\}$, где $i = (1,n)$, $r = (1,n-1)$.

Переход от дерева всех путей графа к стянутому дереву всех путей D соответствует разбиению множества путей графа G на подмножества M_{si}^{*r}, при этом вершины графа D рассматриваются как множества путей M_{si}^{*r}.

4.4. АЛГОРИТМЫ ОПРЕДЕЛЕНИЯ КРАТЧАЙШИХ ПУТЕЙ НА ОСНОВЕ РАНГОВОГО ПОДХОДА

Построение кратчайших путей основывается на следующей процедуре ***A***.

Процедура A

Шаг 1. Из вершины s строятся все возможные пути второго ранга ко всем вершинам графа G.

Шаг 2. Полученное множество путей M^2 разбивается на подмножества M_i^2 таким образом, что каждый из них заканчивался на i.

Шаг 3. В каждом подмножестве выделяется путь минимальной длины.

В результате этого получаем $M_{\min}^2$ — множество путей второго ранга, в котором каждый путь является кратчайшим путем второго ранга от вершины s к соответствующей вершине t. Используя множество $M_{\min}^2$, строятся все пути ранга 3 ко всем вершинам графа. Далее полученное множество M^3 разбивается на подмножества M_i^3, в каждом из которых определяется путь минимальной длины.

Выполнение процедуры A повторяется до построения путей максимально возможного ранга $r = n - 1$.

Очевидно, что после $(n-1)$-го шага повторения процедуры A будут получены все возможные кандидаты на кратчайшие пути между вершинами s и рангов $r = 1, 2, \ldots, n-1$. Выбирая среди

них для каждой вершины пути минимальной длины, определяем кратчайшие пути между вершиной *s* и остальными вершинами графа.

Таким образом, алгоритм определения кратчайших путей между заданной вершиной *s* и остальными вершинами графа представляется следующей последовательностью шагов.

Алгоритм А1

Шаг 1. Процедура ***A*** повторяется (*n* – 1) раз; в результате формируется множество минимальных путей всех возможных рангов ко всем вершинам.

Шаг 2. В каждом из полученных на шаге 1 множеств путей выбираются минимальные пути к вершине *i*, которые и являются искомыми кратчайшими путями от вершины *s* к остальным вершинам графа *G*.

***Утверждение* 4.1.** Алгоритм ***A*1** позволяет определить кратчайшие пути между заданной вершиной *s* и остальными вершинами графа *G*.

Доказательство. Пусть процедура построила все пути ранга 1 от вершины *s* к остальным вершинам графа и на их базе — все возможные пути ранга $r = 2$, т. е. сформированы все подмножества $M_i^{r=2}$. Выберем в каждом подмножестве кратчайший путь $\mu_i^{*r=2}$. Понятно, что пути $\mu_i^{*r=2}$ являются кратчайшими путями ранга $r = 2$ к вершинам *i*, поскольку на этом шаге каждое из подмножеств содержит все возможные пути ранга $r = 2$. Сформируем на базе путей $\{\mu_{si}^{*r=2}\}$ в соответствии с алгоритмом *A*1 все возможные пути ранга $r = 3$, т. е. подмножества $M_i^{r=3}$, и выделим в каждом пути кратчайший путь $\mu_{si}^{*r=3}$.

Докажем, что полученные таким образом пути $\mu_{si}^{*r=3}$ являются кратчайшими путями ранга $r = 3$ из вершины *s* в вершины $i = (\overline{1, n})$, $i \neq s$. Для этого предположим, что к некоторой вершине $i = j$ есть путь $\mu_{si}^{**r=3}$, длина которого меньше, чем $\mu_{si}^{*r=3}$. Поскольку длины путей ранга $r = 3$ в соответствии с предложенной процедурой определяются как $d_{si}^{r=3} = \min\left\{d_{si}^{r=2} + l_{ij}\right\}$, $i = (\overline{1,n})$, $j = (\overline{1,n})$, $i \neq j$, это предположение может быть выполнено, если в графе есть более короткий путь $\mu_{si}^{**r=2}$ ранга $r = 2$, чем пути $\mu_i^{*r=2}$. Это невозможно, так как в графе *G* они являются кратчайшими путями ранга $r = 2$.

Предположим, что алгоритм $A1$ построил кратчайшие пути $\mu_{si}^{*r=k}$ ранга $r = k$ ко всем вершинам графа. Далее в соответствии с процедурой $A1$ построим на их основе все возможные пути ранга $r = k + 1$, т. е. сформируем множества $M_i^{r=k+1}$ и в каждом множестве $M_i^{r=k+1}$, $i = \overline{(1,n)}$ выделим кратчайшие пути $\mu_{si}^{*r=k+1}$.

Докажем, что они являются кратчайшими путями ранга $r = k + 1$. Для этого повторим предыдущее рассуждение: предположим, что есть путь $\mu_{si}^{**r=k+1}$ короче, чем путь $\mu_{si}^{*r=k+1}$, что противоречит первоначальному предположению о том, что построенные алгоритмом $A1$ пути $\mu_{si}^{*r=k}$ являются кратчайшими путями ранга $r = k$ и, следовательно, пути $\mu_{si}^{*r=k+1}$ действительно являются кратчайшими путями ранга $r = k + 1$.

Таким образом, доказано, что алгоритм $A1$ при $r = 3$ определяет кратчайшие пути ранга $r = 3$. На их основе алгоритм $A1$ построил кратчайшие пути ранга $r = k + 1$. Следовательно, в соответствии с методом полной математической индукции алгоритм $A1$ позволяет определять кратчайшие пути произвольного ранга.

После $(n - 1)$-го шага во всех множествах M_i^r построены пути μ_{si}^{*r}. Тогда в соответствии с соотношением (4.6) определены кратчайшие пути между вершиной s и всеми остальными вершинами графа, что и требовалось доказать.

В процессе работы алгоритма $A1$ для произвольных графов возможна ситуация, когда множества $\{M_i^r\}$ могут оказаться пустыми. Это возможно в двух случаях: когда пути текущего ранга r к вершине i в графе G отсутствуют, и когда путь существует, но алгоритм $A1$ не смог его построить (тупиковая ситуация).

Покажем, что тупиковые ситуации не нарушают оптимальность алгоритма $A1$. Тупиковая ситуация может иметь место в случае, если $M_t^{r=q+1} = \varnothing$. Согласно алгоритму $A1$ подмножество формируемых путей M_i^r может быть пустым, если вершина t вошла во все кратчайшие пути ранга $r = q$ к вершинам $i = \overline{(1,n)}$.

***Утверждение* 4.2.** Если в процессе работы алгоритма $A1$ построены кратчайшие пути следующих рангов $\mu_{st}^{*r=1}$, $\mu_{st}^{*r=2}$, ..., $\mu_{st}^{*r=q}$ и подмножество путей $M_{st}^{*r=q+1} = \varnothing$, то кратчайшим путем между вершинами s и t является путь $\mu_{st}^{*} = \min\{\mu_{st}^{*r=1}, \mu_{st}^{*r=2}, ..., \mu_{st}^{*r=q}\}$.

Доказательство. Так как подмножество $M_{st}^{r=q+1} = \varnothing$, то вершина t вошла в пути $\mu_{si}^{*r=q}$, представляющие собой кратчайшие пути ранга $r = q$ из вершины s ко всем остальным вершинам графа $i = (\overline{1,n})$. Предположим, что есть путь $\mu_{st}^{**k=q+1}$, который короче пути $\mu_{st}^{*r=q+1}$. Это возможно в случае, если существуют пути $\mu_{st}^{**r=q}$, более короткие, чем пути $\mu_{st}^{*r=q}$ ранга $r = q$, не проходящие через вершину t.

Однако это противоречит предположению о том, что пути $\mu_{si}^{*r=q}$, проходящие через вершину t, являются кратчайшими путями ранга $r = q$ к вершинам i, следовательно, предположение о наличии более короткого пути $\mu_{st}^{**k=q+1}$, чем путь μ_{st}^{*}, неверно, что и требовалось доказать.

Справедливость утверждения 4.2 вытекает из свойства, присущего всем кратчайшим путям, — любой отрезок кратчайшего пути также есть кратчайший путь между соединенными им вершинами. Из утверждения 4.2 следует важный вывод: если веса в матрице длин $L = \|l_{ij}\|$ — положительные величины, то возникновение тупиковых ситуаций не влияет на оптимальность работы алгоритма $A1$, поскольку кратчайшие пути сами являются источником возникновения тупиков. Значительно сложнее решить задачу, если в матрице весов (длин) содержатся веса отрицательной величины и есть циклы отрицательной длины. В анализируемом графе в этом случае оптимальность работы алгоритма $A1$ может нарушиться. В этом случае следует перейти к другому классу задач дискретной оптимизации — *NP*-полным задачам, возможность решения которых на графе G требует дополнительного исследования.

Фактически предложенный алгоритм $A1$ отображает основное функциональное уравнение динамического программирования для данной задачи:

$$d_{\min}^{r}[i] = \min_{r} \{d_{\min}^{r-1}[i] + l_{ij}\}, \qquad (4.7)$$

где $d_{\min}^{r}[i]$ — минимальная длина $(r - 1)$-го ранга от вершины s к вершине i.

Так как предложенный алгоритм разворачивает процесс динамического программирования от первого шага к последнему, то нахождение оптимального решения возможно после нахождения

всех условных локальных решений и соответствующих значений локальных минимумов. При этом величины кратчайших путей от корневой вершины s ко всем остальным вершинам i определяются следующим соотношением:

$$d_{\min}[i] = \min_{r} \{d_{\min}^{r-1}[i]\}. \quad (4.8)$$

Использование рассмотренного алгоритма в соответствии с принципом оптимальности Беллмана [19] позволяет определить кратчайшие пути из некоторой, произвольной вершины s графа G ко всем остальным вершинам этого графа.

Отличительной особенностью алгоритма $A1$ является то, что на каждом шаге работы сравниваются величины путей одного и того же ранга. Это, в отличие от алгоритмов, основанных на методе расстановок меток, позволяет реализовать эффективное распараллеливание алгоритма определения кратчайших путей в графе.

4.5. АЛГОРИТМ А2 (ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА А1)

Поскольку формирование множеств путей M на каждом ранге можно проводить одновременно, в соответствии с соотношениями (4.7) и (4.8) для завершения работы алгоритма $A1$ при его параллельной реализации необходимо $(n-1)$ раз выполнить операции сложения и сравнения и одну операцию выбора минимального элемента в массиве, определяемую соотношением (4.7). После построения множества путей M^{r+1} в множествах предыдущего ранга M^r необходимо сохранять только экстремальные значения $d_{\min}^r$, и, следовательно, объем требуемой памяти для реализации алгоритма $A2$ определяется множеством путей, формируемых на данном ранге. Количество таких путей равно $n^2 - 3n + 2$, так как в каждом подмножестве M_i^r содержится $(n-2)$ пути ранга r, а число таких подмножеств равно $(n-1)$. Таким образом, временная сложность алгоритма $A2$ не превышает $O(n)$, требуемый объем памяти — $O(n^2)$. Временная сложность алгоритма $A2$ в последовательном варианте его реализации не превысит $O(n^3)$.

Быстродействие алгоритма $A2$ может быть повышено, если для уравнения (4.7) в каждом из формирующих множеств M^r ввести дополнительную проверку:

$$d_{\min}^r[i] \le d_{\min}^{r+1}[i]. \quad (4.9)$$

Если на ранге r для всех подмножеств выполняются соотношения (4.9), то дальше использовать его не следует, так как все полученные на последующих шагах величины путей более высокого ранга будут заведомо длиннее, чем кратчайшие пути, полученные до ранга $r + 1$. В худшем случае, когда кратчайшим путем является гамильтонов путь, временная сложность равна $O(n)$. Последняя ситуация встречается довольно редко, и введение проверки неравенства (4.9) позволяет значительно уменьшить количество шагов в процедуре $A2$. Если в графе ранга r определяется кратчайший путь, то он будет найден на $(r + 1)$-м шаге. Необходимая память для реализации предложенного алгоритма пропорциональна $O(n^2)$.

Практическая реализация $A2$ осложняется тем, что для построения на их основе параллельных архитектур вычислительных систем в процессорных элементах необходимо иметь $(n - 1)$-ну группу регистров для промежуточного хранения путей и при этом, если сеть задается неполным графом, большая часть регистров не используется, что существенно ограничивает размерность решаемых задач.

Предположим, что известны самые короткие пути μ_{sj}^{*r} в каждом подмножестве m_{sj}^{r}. Тогда самый короткий путь от вершины s к вершине j равен $\mu_{sj}^{**} = \min\limits_{r} \{\mu_{sj}^{*r=1}, \mu_{sj}^{*r=2}, ..., \mu_{sj}^{*r=n-1}\}$. Обозначим длину пути μ_{sj}^{*r} как $d(\mu_{sj}^{*r})$ и назовем ее локальным экстремумом ранга r к вершине j, $d(\mu_{sj}^{**r})$ — глобальным экстремумом ранга r к вершине j и длину $d(\mu_{sj}^{*r}) = \min\limits_{j} \{\mu_{sj}^{*r}\}$ — глобальным экстремумом на ярусе. Тогда задача определения кратчайшего пути (КП) может быть сформулирована как задача определения глобальных экстремумов в пространстве, определяемом стянутым деревом путей. Используя свойство, что любой отрезок КП является также кратчайшим путем между вершинами, определяющими данный отрезок, можно построить процедуру определения КП между заданной вершиной s и всеми остальными вершинами произвольного графа $G(V, E)$.

4.6. АЛГОРИТМ А3

Шаг 1. Из вершины s строим пути $m_{sj}^{r=1}$, $j = (\overline{1, n-1})$, определяем локальные экстремумы и находим глобальный экстремум $d(\mu_{sj}^{*r=1})$ на ярусе. Вершину j исключаем из дальнейшего анализа.

Шаг 2. На основе пути, соответствующего текущему глобальному экстремуму на ярусе, строим все возможные пути множества $m_{sj}^{r:=r+1}$ и подтягиваем в них пути, соответствующие локальным экстремумам предыдущего ранга. После этого определяем локальные экстремумы во множествах $m_{sj}^{r:=r+1}$ и глобальный экстремум на ярусе (r + + 1), соответствующий некоторой вершине j. Вершину j исключаем из дальнейшего анализа.

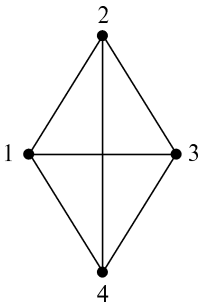

Рис. 4.3. Граф G

Шаг 3. Проверяем, имеет ли место равенство $r = n - 1$; если нет, то переходим к шагу 2; иначе алгоритм заканчивает работу.

Поясним работу данной процедуры на примере. Пусть требуется определить КП в графе G, приведенном на рис. 4.3, от вершины 1 ко всем остальным вершинам графа. Процесс работы процедуры на всех ее шагах работы приведен в табл. 4.1, в которой каждый квадрат соответствует вершине в стянутом дереве путей. В табл. 4.1 показаны пути, построенные процедурой на всех ярусах, символом * отмечены пути, соответствующие локальным экстремумам, а символом ** — пути, соответствующие глобальным экстремумам на ярусах (в скобках указана длина пути).

Отметим, что коэффициент ускорения для алгоритма $A2$ составляет $n^2/\log_2 n$, а для алгоритма $A3$ — $n/\log_2 n$. При этом в алгоритме $A2$ максимальное количество путей, формируемых во множествах m_{sj}^r на произвольных рангах, не превышает величины (n – 1), а для алгоритма $A3$ — двух путей. Кроме того, алгоритм $A2$ всегда должен строить пути ранга $r = n - 1$, а алгоритм $A3$ — до некоторого значения r_k, определяемого неравенством (4.9).

ТАБЛИЦА 4.1. **Процесс построения дерева кратчайших путей**

12(1)** 2	---------- **2**	---------- **2**	---------- **2**	12(1)
13(5)* 3	123(2)** 13(5) 3	---------- **3**	---------- **3**	123(2)
------- 4	124(4)* 4	1234(4) 124(4)** 4	---------- **4**	124(4)
-------- 5	---------- 5	1235(6)* 5	1245(5)** 1235(6) 5	1245(5)
$r = 1$	$r = 2$	$r = 3$	$r = 4$	

ТАБЛИЦА 4.2. **Значения среднего ранга кратчайших путей**

N	25	30	35	40	45	50	60	70
r_k	7	8	8	8	9	9	9	10
max r	11	11	11	12	12	12	14	12

На основании результатов проведенных экспериментальных исследований средних значений рангов кратчайших путей в произвольных графах с весовыми характеристиками, изменяющимися по равномерному закону, для графов с количеством вершин в диапазоне от 5 до 70 и с плотностями ребер в диапазоне от 0,125 до 1 получены значения r_k, приведенные в табл. 4.2.

Таким образом, выигрыш алгоритма $A2$ относительно алгоритма $A3$ определяется величиной n/r_k.

Для реализации этапа 1 рассматриваемого метода решения поставленной задачи целесообразно использовать предложенную параллельную реализацию алгоритма Форда (алгоритм $A2$) и алгоритм Дейкстры (алгоритм $A3$), использующих ранговый подход.

4.7. МЕТОД РЕШЕНИЯ ЗАДАЧИ НАХОЖДЕНИЯ КРАТЧАЙШЕГО ГАМИЛЬТОНОВА ЦИКЛА

Для реализации этапа 2 решения поставленной задачи предложенного подхода необходимо использовать эффективный алгоритм определения кратчайшего гамильтонова цикла (КГЦ).

4.7.1. Формализация задачи КГЦ

Рассмотрим произвольный полный неориентированный граф $G(V, E)$ с n вершинами и m ребрами, пронумерованными от 1 до m. Каждой вершине i графа инцидентно $(n-1)$-но ребро. Поставим в соответствие каждому r-у ребру переменную X_r, принимающую значение, равное 1, если ребро входит в гамильтонов цикл, и равное 0 — в противном случае. Множество всех таких переменных обозначим W, мощность этого множества равна $\frac{n(n-1)}{2}$. Тогда сумма переменных $\sum_{r=1}^{n-1} X_r$, входящих в гамильтонов цикл для каждой вершины i, должна удовлетворять условию

$\sum_{r=1}^{n-1} X_r = 2$. Следовательно, система булевых уравнений

$$\sum_{r=1}^{n-1} X_r = 2\ (i=1), \sum_{r=1}^{n-1} X_r = 2\ (i=2),$$
$$\ldots, \sum_{r=1}^{n-1} X_r = 2\ (i=n) \qquad (4.10)$$

описывает все возможные циклы в графе $G(V, E)$, т. е. для описания произвольного цикла в графе $G(V, E)$ необходимо выделить такое подмножество переменных $\{X_r\}$, чтобы в каждом уравнении в (4.10) присутствовало только по две переменные.

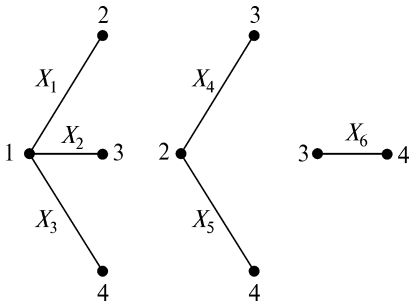

Рис. 4.4. Нумерация ребер графа G

Если каждое ребро r и, соответственно, каждая переменная X_r имеют весовую характеристику P_r, и если в каждом i-м уравнении системы выделить подмножество таких переменных $\{X_r\}$, чтобы в каждом уравнении (4.10) присутствовало по две переменных и сумма весов переменных была минимальной, то данное подмножество образует цикл из ребер графа, и он будет иметь минимальную длину. Например, граф G, приведенный на рис. 4.3, с матрицей весов $\boldsymbol{P}$ и нумерацией ребер, показанной на рис. 4.4, можно представить в виде следующей системы булевых уравнений:

$$\begin{aligned} X_1 + X_2 + X_3 &= 2;\ i=1;\\ X_1 + X_4 + X_5 &= 2;\ i=2;\\ X_2 + X_4 + X_6 &= 2;\ i=3;\\ X_3 + X_5 + X_6 &= 2;\ i=4, \end{aligned} \qquad (4.11)$$

и при этом каждое ребро в (4.11) характеризуется весовой характеристикой P_r (табл. 4.3).

ТАБЛИЦА 4.3. **Описание графа (см. рис. 4.4)**

X_r	X_1	X_2	X_3	X_4	X_5	X_6
Концевые вершины ребра (i, j)	(1,2)	(1,3)	(1,4)	(2,3)	(2,4)	(3,4)
Вес ребра P_r	0	2	1	1	2	0

Для того чтобы произвольное уравнение в (4.10) выполнялось, в нем должны присутствовать только две переменные, равные 1, а остальные должны быть равны нулю, поскольку в гамильтонов цикл входит только по два ребра, инцидентных каждой вершине *i*.

Обозначим множество всех возможных сочетаний пар переменных $X_l X_k$ в каждом *i*-м уравнении через Ω_i. Тогда мощность этого множества будет равна $\alpha_i = \alpha = \frac{(n-1)(n-2)}{2}$, так как для полного графа значения α_i во всех уравнениях будут одинаковы. Тогда уравнения (4.10) можно представить в следующем виде:

$$\begin{gathered}\sum_{x_l x_k \in \Omega_{i=1}} X_l X_k = 1, \quad i = 1; \\ \sum_{x_l x_k \in \Omega_{i=2}} X_l X_k = 1, \quad i = 2; \\ \ldots, \\ \sum_{x_l x_k \in \Omega_{i=n}} X_l X_k = 1, \quad i = n.\end{gathered} \tag{4.12}$$

В соответствии с (4.12) уравнения (4.11) можно записать в виде

$$\begin{aligned} X_1X_2 + \; X_1X_3 + \; X_2X_3 = \; 1 ; \; i = 1; \\ X_1X_4 + \; X_1X_5 + \; X_4X_5 = \; 1 ; \; i = 2; \\ X_2X_4 + \; X_2X_6 + \; X_4X_6 = \; 1; \; i = 3; \\ X_3X_5 + \; X_3X_6 + \; X_5X_6 = \; 1; \; i = 4. \end{aligned} \tag{4.13}$$

Пары $X_l X_k$ в выражениях (4.12) и (4.13) будем характеризовать суммарной весовой характеристикой P_{lk}, равной сумме весовых характеристик P_l и P_k переменных X_l и X_k соответственно. Особенностью решения системы (4.12) является то, что подмножества переменных $\{X_r\}$, удовлетворяющих системе квадратичных булевых уравнений (4.12), должны иметь в каждом из уравнений этой системы только по две переменных, равных 1, чтобы этот набор определял гамильтонов цикл в графе $G(V, E)$. Таким образом, задачу определения кратчайшего гамильтонова цикла в графе $G(V, E)$ можно рассматривать как задачу определения подмножества переменных $\{X_r\}$, удовлетворяющих системе квадра-

тичных булевых уравнений (4.12) и имеющего минимальную суммарную весовую характеристику.

Каждое *i*-е уравнение в (4.12) будем характеризовать величиной $\Delta_i = P_i^{\max} - P_i^{\min}$, определяемой как разность между максимальной весовой характеристикой слагаемого и минимальной весовой характеристикой слагаемого в *i*-м уравнении и весовой характеристикой $\gamma_i = \sum_{i=1}^{\alpha_i} P_i - P_i^{\min}$. Для уравнений (4.13) в соответствии с табл. 4.4 эти характеристики будут иметь вид

$$\begin{aligned}
&i = 1)\ X_1^{2}X_2 + X_1^{1}X_3 + X_2^{3}X_3 = 1; \quad \Delta_1 = 2;\ \gamma_1 = 5;\\
&i = 2)\ X_1^{1}X_4 + X_1^{2}X_5 + X_4^{3}X_5 = 1; \quad \Delta_2 = 2;\ \gamma_2 = 5;\\
&i = 3)\ X_2^{3}X_4 + X_2^{2}X_6 + X_4^{1}X_6 = 1; \quad \Delta_3 = 2;\ \gamma_3 = 5;\\
&i = 4)\ X_3^{3}X_5 + X_3^{1}X_6 + X_5^{2}X_6 = 1; \quad \Delta_4 = 2;\ \gamma_4 = 5.
\end{aligned} \tag{4.14}$$

В уравнениях (4.14) над каждой парой переменных $X_l X_k$ указана сумма весовых характеристик P_l и P_k переменных X_l и X_k соответственно. Выделим в системе (4.14) базовое уравнение, определяемое на основе максимального значения параметра Δ_i. В случае, если максимальные значения окажутся равными, в качестве базового уравнения выбирается то, которое имеет большее значение параметра γ_i; в случае, если Δ_i и γ_i окажутся равными, в качестве базового уравнения выбирается любое из них. Так как в (4.14) в уравнениях параметры Δ_i и γ_i равны, то любое из них может быть выбрано в качестве базового уравнения. Поскольку каждая пара переменных соответствует объединению пары ребер *i—p—j*, то далее вместо уравнений (4.12) рассматриваются последовательности пар ребер, соответствующих парам переменных $X_l X_k$ в (4.12).

Таким образом, вместо уравнений (4.14) имеем последовательности ребер следующего вида:

$$\begin{aligned}
&i = 1)\ 2 -^{0}_{1}- ^{2}3; \quad 2 -^{0}_{1}- ^{1}4; \quad 3 -^{2}_{1}- ^{1}4;\\
&i = 2)\ 1 -^{0}_{2}- ^{1}3; \quad 1 -^{0}_{2}- ^{2}4; \quad 3 -^{1}_{2}- ^{2}4;
\end{aligned} \tag{4.15}$$

$$i = 3)\ 1 \overset{2}{-} 3 \overset{1}{-} 2; \quad 1 \overset{2}{-} 3 \overset{0}{-} 4; \quad 2 \overset{1}{-} 3 \overset{0}{-} 4;$$
$$i = 4)\ 1 \overset{1}{-} 4 \overset{2}{-} 2; \quad 1 \overset{1}{-} 4 \overset{0}{-} 3; \quad 2 \overset{2}{-} 4 \overset{0}{-} 3.$$

В (4.15) над ребрами приведены их весовые характеристики, при этом каждая последовательность имеет те же характеристики Δ_i и γ_i, что и соответствующие уравнения в (4.14). Для решения системы уравнений (4.12) в (4.15) необходимо указать в каждой последовательности одну пару ребер таким образом, чтобы они образовывали цикл минимальной длины.

4.7.2. Процедура формирования одного цикла В

Построение цикла начинается с последовательности, соответствующей базовому уравнению, выбирая в ней пару ребер с минимальным суммарным весом. Например, пусть это будет пара ребер $i—s—j$ с суммарным весом $(P_{is} + P_{sj})$. Далее формирование цикла может осуществляться на основе объединения с другой парой ребер: либо по ребру $i—s$, либо по ребру $s—j$. Если осуществляется формирование цикла по ребру $s—j$, то осуществляется переход на последовательность $i = j$ и в ней просматриваются все пары ребер, содержащие ребро $s—j$. Среди них выбирается пара ребер с минимальным суммарным весом. Пусть это будет пара ребер $s—j—t$. Далее осуществляется объединение с другой парой по ребру $j—t$, для чего выполняется переход на последовательность $i = t$ и в этой последовательности просматриваются все пары ребер, содержащие ребро $j—t$. Среди них выбирается пара ребер с минимальным суммарным весом — пара $j—t—q$. На данный момент сформирована цепочка $i—s—j—t—q$, и далее аналогично, переходя на последовательность $i = q$, снова в этой последовательности просматриваются все пары ребер, содержащие ребро $t—q$, среди них выбирается пара ребер с минимальным суммарным весом. Процедура продолжается до тех пор, пока не просмотрится $(n - 1)$-я последовательность, соответствующая системе уравнений (4.12). Последнюю последовательность можно не посещать, поскольку в ней гарантировано содержится ребро, замыкающее данную цепочку, так как в нем находятся все возможные сочетания ребер по два, инцидентных

вершине, соответствующей номеру данной последовательности. При этом в процессе формирования цепочки на этапах выбора пары ребер, с помощью которой можно продлить цепочку, из анализа будут исключаться пары ребер вершины, которые уже присутствуют в цепочке или образуют ранний цикл. Таким образом, если сформирована цепочка $i—s—j—t—q$ и на основе ребра $t—q$ в q-й последовательности необходимо найти подмножества ребер, содержащих ребро $t—q$, то из анализа исключаются пары вида $t—q—s$, $t—q—j$, так как вершины s, j уже содержатся в сформированной цепочке, а также исключается пара вида $t—q—i$, если число ребер в цепочке меньше n, поскольку она образует ранний цикл.

На базе процедуры *B* предлагается следующая процедура *A*4 определения кратчайшего гамильтонова цикла на основе решения системы уравнений (4.12), представленной последовательностями наборов ребер вида (4.14), соответствующих уравнениям системы (4.12).

4.7.3. Процедура A4

Шаг. 1. Для последовательностей пар ребер, определяемых уравнениями (4.12), рассчитываются характеристики Δ_i и γ_i; выбирается последовательность, определяемая как базовое уравнение.

Шаг 2. В базовой последовательности выделяются пары ребер с минимальным весом (их, в общем случае, может быть λ), которые включаются в множество U_1. Таким образом, сформированы ребра первого яруса.

Шаг 3. С применением процедуры ***B*** выбирается пара ребер $i—s—j$ из U_1, $\lambda := \lambda - 1$, далее подсоединяется следующее ребро, или, если при выборе минимальной пары число минимальных одинаковых пар равно h, h ребер. Таким образом, сформированы ребра следующего яруса.

Шаг 4. На основе всех ребер, сформированных на предыдущем ярусе, используя процедуру ***B***, формируются ребра следующего яруса.

Шаг 5. Проверяется, равно ли число построенных ярусов $n - 1$; если нет, то осуществляется переход на шаг 4, иначе выполняется следующий шаг.

Шаг 6. Из всех построенных циклов выбирается цикл минимальной длины и заносится в множество U_2.

Т А Б Л И Ц А 4.4. **Описание 6-вершинного полного графа**

X_r	Концевые вершины ребер (i, j)	Вес ребра P_r	X_r	Концевые вершины ребер (i, j)	Вес ребра P_r
X_1	(1,2)	0	X_9	(2,6)	1
X_2	(1,3)	0	X_{10}	(3,4)	1
X_3	(1,4)	1	X_{11}	(3,5)	1
X_4	(1,5)	1	X_{12}	(3,6)	1
X_5	(1,6)	0	X_{13}	(4,5)	0
X_6	(2,3)	0	X_{14}	(4,6)	1
X_7	(2,4)	1	X_{15}	(5,6)	0
X_8	(2,5)	0,5			

Шаг 7. Проверяется равенство $\lambda = 0$, если оно не выполняется, то осуществляется переход на шаг 3, иначе выполняется следующий шаг.

Шаг 8. Из множества U_2 выбирается самый короткий цикл.

На этом процедура заканчивает работу, так как кратчайший гамильтонов цикл найден.

Рассмотрим пример работы процедуры **A4** для 6-вершинного полного графа, заданного набором ребер $\{X_r\}$ с весовыми характеристиками $\{P_r\}$ (табл. 4.4).

Для данного графа уравнения (4.12) и их характеристики Δ_i и γ_i будут иметь следующий вид:

$$1)\ X_1^0X_2 + X_1^1X_3 + X_1^1X_4 + X_1^0X_5 + X_2^1X_3 + X_2^1X_4 + X_2^0X_5 + X_3^2X_4 + X_3^1X_5 + X_4^1X_5 = 1;\ \Delta_1 = 2;\ \gamma_1 = 8,$$

$$2)\ X_1^0X_6 + X_1^1X_7 + X_1^{0,5}X_8 + X_1^1X_9 + X_6^1X_7 + X_6^{0,5}X_8 + X_6^1X_9 + X_7^{1,5}X_8 + X_7^2X_9 + X_8^{1,5}X_9 = 1;\ \Delta_2 = 2;\ \gamma_2 = 10,$$

$$3)\ X_2^0X_6 + X_2^1X_{10} + X_2^1X_{11} + X_2^1X_{12} + X_6^1X_{10} + X_6^1X_{11} + X_6^1X_{12} + X_{10}^2X_{11} + X_{10}^2X_{12} + X_{11}^2X_{12} = 1;\ \Delta_3 = 2;\ \gamma_3 = 12,$$

$$4)\ X_3^2X_7 + X_3^2X_{10} + X_3^1X_{13} + X_3^2X_{14} + X_7^2X_{10} + X_7^1X_{13} + X_7^2X_{14} + X_{10}^1X_{13} + X_{10}^2X_{14} + X_{13}^1X_{14} = 1;\ \Delta_4 = 1;\ \gamma_4 = 15,$$

$$5)\ \overset{1,5}{X_4}\overset{1,5}{X_8}+\overset{2}{X_4}\overset{2}{X_{11}}+\overset{1}{X_4}\overset{1}{X_{13}}+\overset{1}{X_4}\overset{1}{X_{15}}+\overset{1,5}{X_8}\overset{1,5}{X_{11}}+\overset{0,5}{X_8}\overset{0,5}{X_{13}}+\overset{0,5}{X_8}\overset{0,5}{X_{15}}+\overset{1}{X_{11}}\overset{1}{X_{13}}+$$

$$+\overset{1}{X_{11}}\overset{1}{X_{15}}+\overset{0}{X_{13}}\overset{0}{X_{15}}=1;\ \Delta_5=2;\ \gamma_5=12,$$

$$6)\ \overset{1}{X_5}\overset{1}{X_9}+\overset{1}{X_5}\overset{1}{X_{12}}+\overset{1}{X_5}\overset{1}{X_{14}}+\overset{0}{X_5}\overset{0}{X_{15}}+\overset{2}{X_9}\overset{2}{X_{12}}+\overset{2}{X_9}\overset{2}{X_{14}}+\overset{1}{X_9}\overset{1}{X_{15}}+\overset{2}{X_{12}}\overset{2}{X_{14}}+$$

$$+\overset{1}{X_{12}}\overset{1}{X_{15}}+\overset{1}{X_{14}}\overset{1}{X_{15}}=1;\ \Delta_2=2;\ \gamma_6=13*. \qquad (4.16)$$

Составляются последовательности ребер, соответствующие уравнениям (4.16) (в последовательностях над каждым ребром приведена его весовая характеристика):

$$1)\ 2\overset{0}{-}1\overset{0}{-}3;\quad 2\overset{0}{-}1\overset{1}{-}4;\quad 2\overset{0}{-}1\overset{0}{-}6;\quad 2\overset{0}{-}1\overset{0}{-}5;\quad 3\overset{0}{-}1\overset{1}{-}4;\quad 5\overset{1}{-}1\overset{0}{-}3;$$
$$6\overset{0}{-}1\overset{0}{-}3;\quad 4\overset{1}{-}1\overset{1}{-}5;\quad 6\overset{0}{-}1\overset{1}{-}4;\quad 5\overset{1}{-}1\overset{0}{-}6;$$

$$2)\ 1\overset{0}{-}2\overset{0}{-}3;\quad 1\overset{0}{-}2\overset{1}{-}4;\quad 1\overset{0}{-}2\overset{0,5}{-}5;\quad 1\overset{0}{-}2\overset{1}{-}6;\quad 4\overset{1}{-}2\overset{0}{-}3;\quad 5\overset{0,5}{-}2\overset{0}{-}3;$$
$$3\overset{0}{-}2\overset{1}{-}6;\quad 4\overset{1}{-}2\overset{0,5}{-}5;\quad 4\overset{1}{-}2\overset{1}{-}6;\quad 6\overset{1}{-}2\overset{0,5}{-}5;$$

$$3)\ 1\overset{0}{-}3\overset{0}{-}2;\quad 1\overset{0}{-}3\overset{1}{-}4;\quad 1\overset{0}{-}3\overset{1}{-}5;\quad 1\overset{0}{-}3\overset{1}{-}6;\quad 2\overset{0}{-}3\overset{1}{-}4;\quad 2\overset{0}{-}3\overset{1}{-}5;$$
$$2\overset{0}{-}3\overset{1}{-}6;\quad 4\overset{1}{-}3\overset{1}{-}5;\quad 4\overset{1}{-}3\overset{1}{-}6;\quad 5\overset{1}{-}3\overset{1}{-}6;$$

$$4)\ 1\overset{1}{-}4\overset{1}{-}2;\quad 1\overset{1}{-}4\overset{1}{-}3;\quad 1\overset{1}{-}4\overset{0}{-}5;\quad 1\overset{1}{-}4\overset{1}{-}6;\quad 2\overset{1}{-}4\overset{1}{-}3;\quad 2\overset{1}{-}4\overset{0}{-}5;$$
$$2\overset{1}{-}4\overset{1}{-}6;\quad 3\overset{1}{-}4\overset{0}{-}5;\quad 3\overset{1}{-}4\overset{1}{-}6;\quad 5\overset{0}{-}4\overset{1}{-}6;$$

$$5)\ 1\overset{1}{-}5\overset{0,5}{-}2;\quad 1\overset{1}{-}5\overset{1}{-}3;\quad 1\overset{1}{-}5\overset{0}{-}4;\quad 1\overset{1}{-}5\overset{0}{-}6;\quad 2\overset{0,5}{-}5\overset{1}{-}3;\quad 2\overset{0,5}{-}5\overset{0}{-}4;$$
$$2\overset{0,5}{-}5\overset{0}{-}6;\quad 3\overset{1}{-}5\overset{0}{-}4;\quad 3\overset{1}{-}5\overset{0}{-}6;\quad 4\overset{0}{-}5\overset{0}{-}6;$$

$$6)\ 1\overset{0}{-}6\overset{1}{-}2;\quad 1\overset{0}{-}6\overset{1}{-}3;\quad 1\overset{0}{-}6\overset{1}{-}4;\quad 1\overset{0}{-}6\overset{0}{-}5*;\quad 2\overset{1}{-}6\overset{1}{-}3;\quad 2\overset{1}{-}6\overset{1}{-}4;$$
$$2\overset{1}{-}6\overset{0}{-}5;\quad 3\overset{1}{-}6\overset{1}{-}4;\quad 3\overset{1}{-}6\overset{0}{-}5;\quad 4\overset{1}{-}6\overset{0}{-}5*. \qquad (4.17)$$

В последовательности, соответствующей базовому уравнению (в (4.17) отмечена звездочкой), выбирается пара ребер с минимальным суммарным весом — это ребра $1\overset{0}{-}6\overset{0}{-}5$ — с суммарной весовой характеристикой, равной 0. Далее на основе ребра 6—5

(это ребро первого яруса) формируются ребра следующего яруса. Для этого в последовательности 5) ищем пару ребер, содержащую ребро 6—5 с минимальным весом, которую можно подсоединить к предыдущей. Такой парой является последовательность ребер $4\overset{0}{-}5\overset{0}{-}6$. Таким образом, сформированы ребра второго яруса. Объединяя пары $1\overset{0}{-}6\overset{0}{-}5$ и $4\overset{0}{-}5\overset{0}{-}6$, получаем цепочку $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4$.

Далее в последовательности 4) ищем пару (пары), содержащую ребро 5—4 с минимальным весом. Такими парами являются ребра $1\overset{1}{-}4\overset{0}{-}5$; $3\overset{1}{-}4\overset{0}{-}5$; $5\overset{0}{-}4\overset{1}{-}6$. При этом подсоединение пары ребер $1\overset{1}{-}4\overset{0}{-}5$ приводит к образованию раннего цикла $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}1$, а подсоединение пары $5\overset{0}{-}4\overset{1}{-}6$ невозможно, поскольку вершина 6 уже присутствует в решении. Поэтому к предыдущему решению можно подсоединить либо пару $3\overset{1}{-}4\overset{0}{-}5$, либо $2\overset{1}{-}4\overset{0}{-}5$, имеющие одинаковый вес (сформированы ребра третьего яруса). Объединяя их с предыдущим решением, получаем две ветки: $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}3$ и $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}2$. Сначала продолжается построение первой ветки: для этого на основе ребра 4—3 формируются ребра четвертого яруса, для чего в последовательности 3) ищем пару, содержащую ребро 4—3 с минимальным весом. Такой парой являются ребра $2\overset{0}{-}3\overset{1}{-}4$ и $1\overset{0}{-}3\overset{1}{-}4$, но объединение с $1\overset{0}{-}3\overset{1}{-}4$ приводит к получению раннего цикла $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}3\overset{0}{-}1$, поэтому объединяется $2\overset{0}{-}3\overset{1}{-}4$ с предыдущим решением, что приводит к решению $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}3\overset{0}{-}2$. Далее на основе ребра 3—2 в последовательности 2) формируются ребра пятого яруса, ищем пару, содержащую ребро 3—2 с минимальным весом. Такой парой являются ребра $1\overset{0}{-}2\overset{0}{-}3$, объединение которых с предыдущим решением позволяет получить $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}3\overset{0}{-}2\overset{0}{-}1$. Получен цикл, поскольку номер яруса стал равным 6 – 1 = 5, и при этом во всех последовательностях присутствует пара ребер; в последовательности 1), которую не анализировали, — это пара $2\overset{0}{-}1\overset{0}{-}6$ из данного цикла. Аналогичным способом строится вто-

ТАБЛИЦА 4.5. **Описание 6-вершинного полного графа с нулевым весом в цикле**

X_r	Концевые вершины ребер (i, j)	Вес ребра P_r	X_r	Концевые вершины ребер (i, j)	Вес ребра P_r
X_1	(1,2)	0	X_1	(2,5)	0
X_1	(1,3)	0	X_1	(2,6)	0
X_1	(1,4)	1	X_1	(3,4)	0
X_1	(1,5)	1	X_1	(3,6)	1
X_1	(1,6)	0	X_1	(4,5)	0
X_1	(2,3)	0	X_1	(4,6)	0
X_1	(2,4)	1	X_1	(5,6)	0

рая ветка, которая имеет $1\overset{0}{-}6\overset{0}{-}5\overset{0}{-}4\overset{1}{-}2\overset{0}{-}3\overset{0}{-}1$ и является циклом с длиной, равной 1. Поскольку оба цикла имеют одинаковую длину, то в исходном графе они являются кратчайшими. Отметим, что наращивание цепочек ребер можно вести в любую сторону и при этом результат не изменится.

Рассмотрим еще один пример с полным 6-вершинным графом, в котором всем ребрам, образующим цикл 1–2–3–4–5–6–1, присвоен вес, равный нулю, а остальным ребрам — произвольным образом присвоены веса, равные 1 и 0. Особенностью данного графа является то, что длины всех кратчайших гамильтоновых циклов в нем равны 0. Граф задан набором ребер $\{X_r\}$ с весовыми характеристиками $\{P_r\}$ (табл. 4.5).

Составим последовательности ребер, соответствующие уравнениям (4.12):

1) $2\overset{0}{-}1\overset{0}{-}3$; $2\overset{0}{-}1\overset{1}{-}4$; $2\overset{0}{-}1\overset{0}{-}6$; $2\overset{0}{-}1\overset{0}{-}5$; $3\overset{0}{-}1\overset{1}{-}4$; $5\overset{1}{-}1\overset{0}{-}3$; $6\overset{0}{-}1\overset{0}{-}3$; $4\overset{1}{-}1\overset{1}{-}5$; $6\overset{0}{-}1\overset{1}{-}4$; $5\overset{1}{-}1\overset{0}{-}6$;

2) $1\overset{0}{-}2\overset{0}{-}3$; $1\overset{0}{-}2\overset{1}{-}4$; $1\overset{0}{-}2\overset{0}{-}5$; $1\overset{0}{-}2\overset{0}{-}6$; $4\overset{1}{-}2\overset{0}{-}3$; $5\overset{0}{-}2\overset{0}{-}3$; $3\overset{0}{-}2\overset{0}{-}6$; $4\overset{1}{-}2\overset{0}{-}5$; $4\overset{1}{-}2\overset{0}{-}6$; $6\overset{0}{-}2\overset{0}{-}5$;

3) $1\overset{0}{-}3\overset{0}{-}2$; $1\overset{0}{-}3\overset{0}{-}4$; $1\overset{0}{-}3\overset{0}{-}5$; $1\overset{0}{-}3\overset{1}{-}6$; $2\overset{0}{-}3\overset{0}{-}4$; $2\overset{0}{-}3\overset{0}{-}5$; $2\overset{0}{-}3\overset{0}{-}6$; $4\overset{0}{-}3\overset{0}{-}5$; $4\overset{0}{-}3\overset{1}{-}6$; $5\overset{0}{-}3\overset{0}{-}6$;

4) $1\overset{1}{-}4\overset{1}{-}2$; $1\overset{1}{-}4\overset{0}{-}3$; $1\overset{1}{-}4\overset{0}{-}5$; $1\overset{1}{-}4\overset{0}{-}6$; $2\overset{1}{-}4\overset{0}{-}3$; $2\overset{1}{-}4\overset{0}{-}5$; $2\overset{1}{-}4\overset{0}{-}6$; $3\overset{0}{-}4\overset{0}{-}5$; $3\overset{0}{-}4\overset{0}{-}6$; $5\overset{0}{-}4\overset{0}{-}6*$;

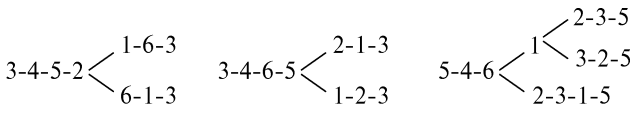

Рис. 4.5. Деревья, построенные процедурой *A*4

5) $1\overset{1}{-}5\overset{0}{-}2$; $1\overset{1}{-}5\overset{0}{-}3$; $1\overset{1}{-}5\overset{0}{-}4$; $1\overset{1}{-}5\overset{0}{-}6$; $2\overset{0}{-}5\overset{0}{-}3$; $2\overset{0}{-}5\overset{0}{-}4$; $2\overset{0}{-}5\overset{0}{-}6$; $3\overset{0}{-}5\overset{0}{-}4$; $3\overset{0}{-}5\overset{0}{-}6$; $4\overset{0}{-}5\overset{0}{-}6$;

6) $1\overset{0}{-}6\overset{1}{-}2$; $1\overset{0}{-}6\overset{1}{-}3$; $1\overset{0}{-}6\overset{1}{-}4$; $1\overset{0}{-}6\overset{0}{-}5$; $2\overset{1}{-}6\overset{1}{-}3$; $2\overset{1}{-}6\overset{1}{-}4$; $2\overset{1}{-}6\overset{0}{-}5$; $3\overset{1}{-}6\overset{1}{-}4$; $3\overset{1}{-}6\overset{0}{-}5$; $4\overset{1}{-}6\overset{0}{-}5$. (4.18)

Базовым уравнением является уравнение, составленное для 4-й вершины, с максимальными характеристиками $\Delta_{i=4} = 2$ и $\gamma_{i=4} = 8$. Соответствующая ему последовательность ребер отмечена звездочкой в (4.18). В последовательности 4) из (4.18) три пары ребер имеют минимальный вес, равный 0: 3—4—5; 3—4—6; 5—4—6. На основе этих пар процедура *A*4 построит три дерева (рис. 4.5).

На рис. 4.5 все ребра деревьев имеют вес, равный 0, т. е. процедура в исходном графе построила все кратчайшие циклы, которые можно построить на основе пар ребер 3—4—5; 3—4—6; 5—4—6. В общем случае, кратчайшие циклы, построенные от разных пар ребер, соответствующих базовому уравнению, могут иметь разную длину. Поэтому следует выбрать из них самый короткий, что реализуется процедурой *A*4 на восьмом шаге ее работы.

Рассмотрим вопрос оптимальности работы процедуры *A*4. Последовательности ребер, соответствующих уравнениям (4.12), можно рассматривать как множества $\{U_i\}$ объектов $\{Q_r\}$, имеющих определенную стоимость $\{C_r\}$. Требуется выбрать по одному объекту из каждого множества таким образом, чтобы суммарная стоимость выбранных объектов была минимальной. При этом после выбора очередного объекта следующий можно выбрать только в случае, если он удовлетворяет некоторым правилам *R*. Совокупность объектов из каждого множества U_i, удовлетворяющих правилу *R*, будем называть полной. Роль правила *R* реализуется процедурой *B*. Работа процедуры *A*4 заключается в том, что в некотором множестве $U_{i=s}$ выбирается объ-

ект $Q_{r=d}(C_{r=d}^{\min})$ минимальной стоимости, и при этом все остальные объекты, находящиеся в этом множестве, имеют большую стоимость. Далее среди всех объектов, удовлетворяющих правилу R в множестве U_i, находим следующий объект $Q_{r+1}(C_{r+1}^{\min})$ с минимальной стоимостью, при этом стоимость $C_{r+1}^{\min} < C_r \in U_i$, где C_r — стоимости объектов, принадлежащих U_i, но не удовлетворяющих правилам R. Процедура продолжает работать до тех пор, пока в соответствии с правилами R из всех множеств U_i не будет выбрано по одному объекту с суммарной стоимостью S_d. Правило R отождествляется с процедурой B, а это значит, что если процедура $A4$ построила один цикл, то он является кратчайшим при условии, что каждый раз при выборе пар ребер в последовательностях нет ситуации, когда в последовательностях встречаются несколько пар ребер с минимальным весом. В дальнейшем о такой ситуации будем говорить как о некотором свойстве ν.

При наличии свойства ν при решении уравнения (4.12) процедура $A4$ строит λ деревьев всех цепочек из ребер, которые на последнем шаге замыкаются в циклы с μ ветками, где λ — количество пар ребер с минимальным весом в базовой последовательности с минимальным весом. Каждое дерево, построенное на основе одной из пар ребер $X_l X_k$ с минимальным весом, принадлежащих базовой последовательности, является деревом всех кратчайших цепочек, которое можно построить на основе ребер, соответствующих паре переменных $X_l X_k$. После того, как все деревья построены, процедура $A4$ из всех возможных кратчайших циклов, построенных с учетом возникновения свойства ν, выбирает самый короткий цикл, который и является решением рассматриваемой задачи. Отметим, что в принципе начало работы процедуры $A4$ можно начинать с любой последовательности, но в процедуре $A4$ выбирается базовая последовательность с наибольшим значением параметров Δ_i и γ_i, что позволяет существенно сократить число веток при построении деревьев кратчайших цепей. Это можно объяснить на примере для графа, заданного табл. 4.5. Если построение начинается не с базовой последовательности $i = 6$ с $\Delta_{i=6} = 2$ и $\gamma_{i=6} = 13$, а с последовательности $i = 2$ с $\Delta_{i=2} = 2$ и $\gamma_{i=2} = 10$, то в этом случае дерево цепей строится на основе пары ребер $1 \overset{0}{-} 2 \overset{0}{-} 3$ с мини-

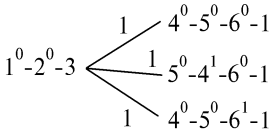

Рис. 4.6. Дерево, построенное процедурой *A*4 на основе пары ребер 1—2—3

мальным суммарным весом, равным 0, и при этом в построенном дереве цепей будет не две ветки, а три (рис. 4.6).

В худшем случае общее число веток, построенных в деревьях кратчайших цепей процедурой *A*4, равно $\varphi = \lambda \cdot \mu$ и зависит от числа пар ребер минимального веса, используемых на каждом шаге работы процедуры *A*4: сколько таких пар, столько и ветвлений в дереве, построенных процедурой *A*4. Оценим временную сложность работы процедуры *A*4.

Число пар ребер во всех последовательностях, соответствующих уравнениям (4.12), равно $\alpha = \dfrac{(n-1)(n-2)}{2}$, поэтому для анализа одной последовательности требуется не более $(n-1)(n-2) < n^2$ операций сравнения. Так как необходимо проанализировать n таких последовательностей, то для построения одного цикла с помощью процедуры *A*4 понадобится не более n^3 операций сравнения. Для расчета параметров Δ_i и γ_i — не более n^2 и $2\log_2 n$ операций выбора минимального элемента из n^2 чисел. Поскольку общее число веток, построенных процедурой *A*4, равно φ, то для определения длин всех веток, содержащих по n ребер, понадобится $\varphi \cdot n$ операций сложения. Для построения всех веток для выбора ветки минимальной длины требуется еще $\log_2 \varphi$ операций (для выбора минимального элемента из массива из φ чисел). Таким образом, общая временная сложность работы процедуры не превысит

$$O(n^3 + n^2 + 2\log_2 n + n\varphi)\varphi + \log_2 \varphi \approx O(n^3\left(1 + \frac{\varphi}{n^2}\right)\varphi + \log_2 \varphi). \quad (4.20)$$

Понятно, что если построить деревья от всех пар ребер в произвольной последовательности без выбора минимальных элементов, то дерево будет содержать число веток, равное $\dfrac{(n-1)!}{2}$.

Процедура *A*4 строит не все деревья, а равное количеству пар ребер с минимальными весами, находящихся в базовой последо-

вательности. Число веток в самом дереве зависит только от числа пар ребер с минимальными весами на каждом шаге процедуры $A4$. Так, в примере анализа 6-вершинного графа (см. табл. 4.4) в базовом множестве оказались две пары ребер минимального веса, и процедура $A4$ построила дерево, состоящее из одной ветки, т. е. из множества 60 различных циклов, выполнив около $2n^3$ элементарных операций, процедура построила кратчайший цикл. Для 6-вершинного графа, заданного табл. 4.5, 10 ребер имеют вес, равный 0, и пять ребер — равный 1. Таким образом, в анализируемых последовательностях почти 60 % ребер участвуют в образовании минимальных пар ребер и при этом процедура строит три дерева (см. рис. 4.6), содержащие всего 7 веток, и для определения кратчайшего гамильтонова цикла процедура $A4$ выполнила не более $7n^3$ элементарных операций. Из приведенных примеров следует, что при увеличении диапазона изменения весов ребер можно ожидать, что среднее число веток, которые будет строить предложенная процедура $A4$, будет соизмерима с размерностью задачи n из-за того, что в процессе работы процедуры $A4$ число пар ребер с минимальным весом на каждом шаге ее работы будет уменьшаться. При этом алгоритм ее решения будет иметь в среднем полиномиальную сложность. Для проверки данной гипотезы был проведен эксперимент. При равномерном законе распределения весов ребер определялось среднее значение числа веток, построенных процедурой $A4$ при изменении весовых характеристик ребер в диапазонах от 0 до 5; от 0 до 30 и от 0 до 50 и при разных значениях n. Все результаты получены с доверительной вероятностью 0,95. Погрешность предложенной процедуры при равномерном законе распределения весовых характеристик ребер графа при изменении числа вершин в диапазоне от 10 до 100 не превысила в среднем 20 %. Преимуществом данной процедуры является то, что ее можно адаптировать для решения задачи определения гамильтоновости произвольных графов, и с увеличением диапазона изменения весовых коэффициентов ребер графа число выполняемых шагов процедурой $A4$ уменьшается с уменьшением погрешности.

ВЫВОДЫ К ГЛАВЕ 4

1. Предложена модель мониторинга состояния объектов мониторинга РВС на основе минимизации суммарного времени запуска и выполнения удаленных сервисов мониторинга объектов на ресурсах РВС.

2. Разработан двухэтапный метод решения задачи минимизации суммарного времени задержки при запуске удаленных агентов на ресурсах РВС, использующий на этапе 1 стянутое дерево всех путей и распараллеливание процесса решения на узлах РВС, что частично решает проблему разработки специальных архитектур для решения поставленной задачи.

3. Для этапа 2 предложен метод определения кратчайших гамильтоновых циклов с полиномиальной временной сложностью, который позволяет эффективно решить задачу определения порядка запуска (опроса) удаленных сервисов для мониторинга объектов на ресурсах РВС в реальном времени. Особенностью предложенного метода является сведение решения задачи определения кратчайших гамильтоновых циклов к решению системы квадратичных булевых уравнений с задаваемыми характеристиками.

Предложенные для выполнения обоих этапов методы и алгоритмы позволяют использовать технологии параллельного программирования их реализации с применением CUDA-технологий, что значительно повышает производительность системы мониторинга РВС.

ВЫВОДЫ

В монографии изложены общие подходы к анализу фундаментальных и прикладных проблем, связанных с математическим моделированием и решением широких классов оптимизационных задач, имеющих комбинаторную структуру. Среди них задачи конструкторского проектирования микроэлектронных устройств, размещения геометрических объектов, управления распределенными вычислениями с различными функциями цели и ограничениями, построения оптимальных расписаний мониторинга объектов в распределенных средах.

Основные полученные результаты заключаются в следующем.

Сформулирована и обоснована общая постановка задачи конструкторского проектирования микроэлектронных устройств. Построены математические модели основных классов задач конструкторского проектирования микроэлектронных устройств в рамках общей постановки задачи. Разработана общая схема итерационно-последовательного алгоритма решения задач для построенных математических моделей. Проанализированы и продемонстрированы на примерах способы улучшения глобальных свойств алгоритмов.

С единых позиций представлены формализация описаний, генерация и перечисление комбинаторных конфигураций с заданными свойствами с целью математического моделирования и решения задач со сложной комбинаторной структурой. Введено понятие композиционного k-образа комбинаторных множеств (k-множества), как класса комбинаторных множеств, обладающих специальными комбинаторными свойствами. Проанализированы задачи описания, генерации и перечисления k-множеств, при этом применены результаты теории комбинаторных видов. Результаты использованы в математическом моделировании комбинаторных оптимизационных задач геометрического проектирования.

Разработаны математические модели общих и частных задач планирования в распределенных вычислительных средах. Проведенный анализ показал, что они могут быть сведены к задачам с булевыми переменными с линейными и нелинейными функциями цели. Для решения этих задач разработан и обоснован общий подход, позволяющий на основе единой теоретической платформы получать эффективные решения. Для оптимизации системно-ориентированных показателей функционирования распределенных вычислительных систем (РВС), а именно, минимизации количества используемых вычислительных кластеров, предложена модель в виде задачи о наименьшем вершинном покрытии (ЗНВП). Разработан метод решения ЗНВП с малой вычислительной сложностью и низкой погрешностью.

Разработаны подход и процедуры построения интеллектуальных вычислительных систем на основе стратегии рангового метода.

Для контроля состояния объектов РВС предложена модель мониторинга, которая построена на основе минимизации суммарного времени запуска и выполнения удаленных сервисов на объектах РВС и создания на их основе централизованной базы данных. Она позволяет оптимизировать время обработки оперативной информации для принятия управленческих решений по выбору стратегий планирования. Для минимизации времени обхода объектов мониторинга разработан метод определения кратчайшего гамильтонова цикла с полиномиальной сложностью, позволяющий построить оптимальный график запуска удаленных сервисов на объектах РВС в реальном времени. Для разработанных методов представляется эффективным использование технологий параллельных вычислений и гибридных систем на основе CPU- и GPU-архитектур.

В.В. СЕМЕНЕЦЬ,
І.В. ГРЕБЕННІК, С.В. ЛІСТРОВИЙ,
С.В. МІНУХІН, А.О. ОВЕЗГЕЛЬДИЄВ

МОДЕЛІ І МЕТОДИ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ В ПРОЕКТУВАННІ ТА КЕРУВАННІ

Київ: Наукова думка, 2019. — 176 с.

Монографія присвячена розробці та дослідженню математичних моделей і методів комбінаторної оптимізації, орієнтованих на розв'язання задач проектування та керування, серед яких конструкторське проектування мікроелектронних пристроїв, геометричне проектування та керування розподіленими обчисленнями. Наведено загальну формальну постановку і постановки основних класів задач конструкторського проектування мікроелектронних пристроїв, а також методи і алгоритми їх розв'язання. Викладено єдиний підхід до формалізації описів, генерації та перерахування класів комбінаторних конфігурацій, що мають задані властивості, для математичного моделювання і розв'язання задач зі складною комбінаторною структурою. В межах єдиного підходу побудовано та проаналізовано математичну модель багатокритеріальної задачі розміщення паралелепіпедів. Розглянуто математичні постановки та методи розв'язання задач планування в розподілених обчислювальних системах (РОС), що зводяться до задач дискретної оптимізації з лінійними і нелінійними функціями цілей, для розв'язання яких запропоновано загальний підхід. У рамках розв'язання задач керування розподіленими обчисленнями запропоновано метод моніторингу стану компонентів РОС, що використовує зображення РОС у вигляді неповнозв'язного графу, на основі розв'язання задач знаходження найкоротшого шляху і найкоротшого гамільтонова циклу.

Для науковців і фахівців, які займаються оптимізацією широкого кола задач проектування та керування системами, а також для аспірантів відповідних спеціальностей.

V.V. SEMENETS,
I.V. GREBENNIK, S.V. LISTROVOY,
S.V. MINUKHIN, A.O. OVEZGELDYYEV

MODELS AND METHODS OF COMBINATORIAL OPTIMIZATIONIN DESIGNAND CONTROL

Kiyv: Naukova dumka, 2019. — 176 p.

The monograph is devoted to the development and investigation of mathematical models and methods of combinatorial optimization, focused on solving design and control problems, including design engineering of microelectronic devices, geometric design and control of distributed computing. The general formal description and statements of the main classes of problems for the design engineering of microelectronic devices are given; methods and algorithms for solving them are proposed. A general approach to the formalization of description, generation and enumeration for classes of combinatorial configurations with given properties for mathematical modeling and solving problems with a complex combinatorial structure is represented. Based on the general approach, a mathematical model of the multi-criteria parallelepiped placement problem is constructed and analyzed. Mathematical models and methods for solving planning problems in distributed computing systems (DCS), which are reduced to discrete optimization problems with linear and nonlinear objective functions, are considered; for solving of the problems a general approach has been proposed. In the framework of solving of control of distributed computing problems, a method for monitoring the state of the DCS components is proposed. The method is based on representing the DCS as an incompletely connected graph and solving the problems of finding the shortest path and the shortest Hamiltonian cycle.

The monograph is oriented to researchers and professionals involved in the optimization of a wide range of design and control systems, as well as graduate students of relevant specialties.

СПИСОК ЛИТЕРАТУРЫ

К главе 1

1. Review on VLSI design using optimization and self-adaptive particle swarm optimization / S.B. Vinay Kumar, P.V. Rao, H.A. Sharath, B.M. Sachin, U.S. Ravi, B.V. Monica // Journal of King Saud University — Computer and Information Sciences. 2018.

2. *Funke J., Hougardy S., Schneider J.* An exact algorithm for wirelength optimal placements in VLSI design // Integr. VLSI J. 2016. Vol. 52. P. 355—366.

3. Mansor Mohd. Asyraf, Shareduwan Mohd., Kasihmuddin Mohd., Sathasivam Saratha. VLSI circuit configuration using satisfiability logic in Hopfield network // I.J. Intell. Syst. 2016. Appl. 9. P. 22—29.

4. Algorithms for VLSI Physical Design Automation, 3e Paperback. 2005.

5. *Vai M. Michael.* VLSI Design. CRC Press, 2017. 424 p.

6. *Gerez Sabih H.* Algorithms for VLSI Design Automation. John Wiley & Sons, Inc. New York, 1999.

7. *Abboud Nadine, Gro Martin, Koch Thorsten Tschel.* Mathematical Methods for Physical Layout of Printed Circuit Boards: An Overview // Operations Research-Spektrum. 2008. Vol. 30 (3). P. 453—468.

8. *Семенец В.В.* Оптимизационные методы и алгоритмы автоматизированного конструирования микроэлектронных устройств: учеб. пособие. Киев: Вища школа, 1992. 45 с. ISBN 5-11-004140-7.

9. *Семенець В.В.* Проектування одноблокових радіоелектронних приладів із заданим тепловим режимом: Монографія / В.В. Семенець, А.М. Сінотін, Т.А. Колесникова. Харків: ХНУРЕ, 2006. 172 с. ISBN 966-659-133-2.

10. Технология межсоединений электронной аппаратуры: учебник / В.В. Семенец, Кратц Джон, И.Ш. Невлюдов, В.А. Палагин. Харьков: СМИТ, 2005. 432 с. ISBN 966-8530-28-4.

11. *Невлюдов І.Ш.* Введення в мікросистемну техніку та нанотехнології / І.Ш. Невлюдов, В.А. Палагін, В.В. Семенець. Харків: Компанія «СМІТ», 2011. 416 с. ISBN 978-966-2028-73-7.

12. *Сакало С.М.* Надвисокі частоти в медицині (терапія і диагностика): навч. пос. / С.М. Сакало, В.В. Семенець, О.Ю. Азархов. Харків: ХНУРЕ, 2005. 264 с. ISBN 966-8604-07-5.

13. Микропроцессоры в информационно-измерительных системах: учеб. пособие / О.Г. Аврунин, О.В. Запорожец, В.В. Семенец и др. Харьков: ХНУРЭ, 2015. 180 с.

14. *Semenets V.V., Hahanova I.V., Hahanov V.I.* Design of digital systems by using VHDL language. Kharkov: KHNURE, 2003. 492 p.

15. *Прасол И.В., Семенец В.В.* Проблемы агрегирования моделей при решении задачи электрического анализа сложных электронных схем / Тех-

нічна електродинаміка. Тем. вип. «Силова електроніка та енергоефективність». Київ: Інститут електродинаміки НАН України, 2009. С. 98—101.

16. *Семенец В.В.* Принципы создания методологических основ проектирования САПР / В.В. Семенец, В.Г. Иванов // Прикладная радиоэлектроника: науч.-техн. журн. Харьков: ХНУРЭ, 2008. Т. 7, № 2. С. 139—144.

17. *Semenets V.V.* Technical aspects for development laboratory base for learning FPGA and microcontroller systems, Proceedings of the X-th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics», CADSM-2009. P. 145.

18. *Безкоровайний В.В.* Синтез моделей багатокритеріального оцінювання методом компараторної ідентифікації / В.В. Безкоровайний, О.М. Драз, В.В. Семенець // Інформаційні технології та комп'ютерне моделювання: матер. Міжн. наук.-практ. конф., Івано-Франківськ, 2018. С. 266—269.

19. *Аврунін О.Г.* Основи мови VDHL для проектування цифрових пристроїв на ПЛІС: навч. пос. / О.Г. Аврунін, Т.В. Носова, В.В. Семенець. Харків: ХНУРЕ, 2018. 196 с. ISBN 978-966-659-247-0.

20. *Stallmann M.F.M., Brglez F., Ghosh D.* Heuristics, experimental subjects, and treatment evaluation in bigraph crossing minimization. ACM Journal of Experimental Algorithms, 6:8, 2001.

21. *Garey M.R., Johnson D.S.* Computers and Intractability: A Guide to the theory of NP-Completeness. Freeman, 1979.

22. *Стоян Ю.Г., Яковлев С.В.* Математические модели и оптимизационные методы геометрического проектирования. Киев: Наукова думка, 1986. 268 с.

23. *Ng A.P.-C., Raghavan P., Thompson C.D.* Experimental results for a linear program global router // Computers and Artificial Intelligence. 1987. Vol. 6, N 3. P. 229—242.

24. *Лошаков В.Н.* Метод исключений и его применение в задачах машинной трассировки соединений // Электронная техника. Сер. 3. Микроэлектроника. 1974. Вып. 1 (49). С. 54—61.

25. Алгоритм трассировки конкурирующих проводников / Ю.Х. Вермишев, М.И. Макаров, Л.П. Рябов, Ю.Н. Темницкий // Обмен опытом в радиопромышленности. 1975. Вып. 6. С. 127—131.

26. Селютин В.А. Машинное конструирование электронных устройств. М.: Сов. радио, 1977. 384 с.

27. *Гуляницкий Л.Ф., Малышко С.А.* Комбинаторный подход к решению одного класса задач размещения. Киев, 1988. 19 с. (Препр. АН УССР / Ин-т кибернетики им. В.М. Глушкова: 88-32).

28. *Реклейтис Г., Рейвиндран А., Рэгсдел К.* Оптимизация в технике: В 2-х кн. Кн. 1. Пер. с англ. М.: Мир, 1986. 349 с., ил.

29. *Абрайтис Л.Б.* Автоматизация проектирования топологии цифровых интегральных микросхем. М.: Радио и связь, 1985. 200 с.

30. *Базилевич Р.П.* Декомпозиционные и топологические методы автоматизированного конструирования электронных устройств. Львов: Вища школа, 1981. 168 с.

31. *Preas B.T., Karger P.G.* Automatic placement: a review of current techniques // Proc. of the 23rd DAC. 1986. P. 622—629.

32. Методы разбиения схем РЭА на конструктивно законченные части / К.К. Морозов, А.Н. Мелихов, Л.С. Бернштейн, В.Г. Одиноков, В.М. Курейчик; Под ред. К.К. Морозова. М.: Сов. радио, 1978. 136 с.

33. *Петренко А.И., Тетельбаум А.Я.* Формальное конструирование электронно-вычислительной аппаратуры. М.: Сов. радио, 1979. 256 с.

34. *Gamblin R.L., Jacobs M.Q., Tunis C.J.* Automatic packaging of miniaturized circuits // Advanced Electronic Circuit Packaging Symposium. 1962. Vol. 2. P. 219—232.

35. *Hightower D.W.* The interconnection problem: a tutorial // Computer. 1974. Vol. 7, N 4. P. 18—32.

36. *Вентцель Е.С.* Исследование операций. М.: Сов. радио, 1972. 552 с.

37. *Гуревич Д.З., Селютин В.А.* Реализация на ЭВМ системы алгоритмов топологического проектирования // Электронная техника. Сер. 6. Микроэлектроника. 1971. Вып. 6. С. 145—152.

38. *Kirpatrick S., Yelatt C.D., Vecchi M.P.* Optimization by simulated annealing // Science. 1983. Vol. 220, N 4598. P. 571—680.

39. *Растригин Л.А.* Статистические методы поиска. М.: Наука, 1968. 376 с.

К главе 2

1. *Сергиенко И.В.* Математические модели и методы решения задач дискретной оптимизации. Киев: Наукова думка, 1988. 472 с.

2. *Стоян Ю.Г., Яковлев С.В.* Математические модели и оптимизационные методы геометрического проектирования. Киев: Наукова думка, 1986. 268 с.

3. *Згуровский М.З., Павлов А.А.* Труднорешаемые задачи комбинаторной оптимизации в планировании и принятии решений. Киев: Наукова думка, 2016.

4. *Toth P., Vigo D.* The Vehicle Routing Problem. Society for Industrial and Applied Mathematics. Philadelphia, 2002.

5. *Сергиенко И.В., Шило В.П.* Задачи дискретной оптимизации: проблемы, методы, решения, исследования. Нац. акад. наук Украины, Ин-т кибернетики им. В.М. Глушкова. Киев: Наукова думка, 2003. 261 с.

6. *Aigner M.* Combinatorial theory. Berlin; Heidelberg; New York: Springer-Verlag, 1979.

7. *Sachkov V.N.* Combinatorial methods in discrete mathematics // Encyclopedia of Mathematics and its Applications. Vol. 55. Cambridge: Cambridge University Press, 1996.

8. *Баранов Б.И., Стечкин Б.С.* Экстремальные комбинаторные задачи и их приложения. М.: Наука, 1989. 160 с.

9. *Berge C.* Principes de combinatoire. Paris: Dunod, 1968.

10. *Пойа Дж.* Комбинаторные вычисления для групп, графов и химических соединений // Перечислительные задачи комбинаторного анализа. М.: Мир, 1979. С. 36—138.

11. *Goulden I.P., Jackson D.M.* Combinatorial enumeration. Ontario, Canada: John Wiley&Sons, 1983.

12. *Stanley R.P.* Enumerative combinatorics. Vol. 1. Wadsworth, Inc. California, 1986.

13. *Wilf Herbert S.* Generatingfunctionology. Wellesley, Massachusetts: A K Peters Ltd., 2006.

14. *Knuth Donald.* The Art of Computer Programming. Vol. 4. Fascicle 2: Generating All Tuples and Permutations. Addison-Wesley, 2005.

15. *Knuth Donald.* The Art of Computer Programming. Vol. 4. Fascicle 3: Generating All Combinations and Partitions. Addison-Wesley, 2005.

16. *Kreher Donald L., Stinson Douglas R.* Combinatorial Algorithms: Generation Enumeration and Search. CRC Press, 1999.

17. *Bona M.* Combinatorics of Permutations. Chapman Hall-CRC, 2004.

18. *Ruskey F.* Combinatorial generation. Dept. of Computer Science Univ. of Victoria, Canada, 2003. 1j-CSC 425/520.

19. *Stanton Dennis, White Dennis.* Constructive Combinatorics. Springer, 1986

20. *Bergeron F., Labelle G., Leroux P.* Combinatorial species and tree-like structures. Cambridge: University Press, 1998.

21. *Bergeron F., Labelle G., Leroux P.* Introduction to the Theory of Species of Structures, University of Quübec, Montréal. 2008.

22. *Stoyan. Yu.G., Grebennik I.V.* Description and Generation of Combinatorial Sets Having Special Characteristics // International Journal of Biomedical Soft Computing and Human Sciences. 2013. Special Volume «Bilevel Programming, Optimization Methods, and Applications to Economics». Vol. 18, N 1. P. 83—88.

23. *Flajolet P., Sedgewick R.* Analytic Combinatorics. Cambridge: University Press, 2009.

24. *Korte B.* Combinatorial Optimization: Theory and Algorithms / B. Korte, J. Vygen. Heidelberg; New York: Springer, 2012. 660 p.

25. *Papadimitriou C.H.* Combinatorial Optimization: Algorithms and Complexity / C.H. Papadimitriou, K. Steiglitz. Dover Publications Inc., 2013. 528 p.

26. *Grande F.* On k-level matroids: geometry and combinatorics / F. Grande Doctor of Natural Sciences Dissertation. Berlin: Institut für Mathematik und Informatik, Freie Universität Berlin, 2015. 122 p.

27. *Семенова Н.В., Колєчкіна Л.М.* Векторні задачі дискретної оптимізації на комбінаторних множинах: методи дослідження та розв'язання. Київ: Наукова думка, 2009. 266 с.

28. *Сергиенко И., Гуляницкий Л.* Классификация прикладных методов комбинаторной оптимизации // Кибернетика и системный анализ. 2009. Т. 5. С. 71—83.

29. *Сергиенко И.В., Шило В.П.* Современные подходы к решению сложных задач дискретной оптимизации // Проблемы управления и информатики. 2016. № 1. Р. 32—40.

30. *Стоян Ю.Г.* Некоторые свойства специальных комбинаторных множеств. Харьков. 1980. 22 с. (Препр. АН УССР / Ин-т пробл. машиностроения; 85).

31. *Стоян Ю.Г.* Об одном отображении комбинаторных множеств в евклидово пространство. Харьков, 1982. 33 с. (Препр. АН УССР / Ин-т пробл. машиностроения; 173).

32. *Stoyan Yu., Yemets O.* Theory and methods of Euclidean combinatorial optimization. Kyiv: ISDO, 1993.

33. *Стоян Ю.Г., Гребенник И.В., Емец О.А.* Комбинаторные множества размещений и их свойства. Харьков, 1990. 38 с. (Препр. АН УССР / Ин-т пробл. машиностроения; 342).

34. *Емец О.А.* Множество сочетаний с повторениями, отображенное в R^k и свойства задач оптимизации на нем // ДАН УССР. Сер. А. 1991. № 4. С. 69—72.

35. *Ємець О.О., Колєчкіна Л.М., Недобачій С.І.* Дослідження областей визначення задач евклідової комбінаторної оптимізації на переставних множинах. Полтава: Полтав. держ. техн. ун-т ім. Юрія Кондратюка, ЧПКП «Легат», 1999. Ч. 1. 64 с.; Ч. 2. 32 с.

36. *Ємець О.О., Колєчкіна Л.М.* Задача оптимізації на переставленнях з дробово-лінійною цільовою функцією: властивості множини допустимих розв'язків // Український математичний журнал. 2000. Т. 52, № 12. С. 1630—1640.

37. *Стоян Ю.Г.* Евклидовы комбинаторные конфигурации: монография / Ю.Г. Стоян, С.В. Яковлев, О.С. Пичугина. Харьков: Константа, 2017. 404 с.

38. *Стоян Ю.Г., Яковлев С.В.* Построение выпуклых и вогнутых функций на перестановочном многограннике // ДАН УССР. Сер. А. 1988. № 5. С. 68—70.

39. *Стоян Ю.Г., Яковлев С.В.* Свойства выпуклых функций на перестановочном многограннике// ДАН УССР. Сер. А. 1988. № 3. С. 238—240.

40. *Стоян Ю.Г., Яковлев С.В., Гребенник И.В.* Экстремальные задачи на множестве размещений. Харьков, 1991. 37 с. (Препр. АН УССР / Ин-т пробл. машиностроения; 347).

41. *Яковлев С.В., Гребенник И.В.* О некоторых классах задач оптимизации на множествах размещений и их свойствах // Изв. вузов. Математика. 1991. № 11. С. 74—86.

42. *Яковлев С.В.* Оценки минимума выпуклых функций на евклидовых комбинаторных множествах // Кибернетика. 1989. № 3. С. 80—97.

43. *Ємець О.О., Роскладка А.А.* Про оцінки мінімумів цільових функцій при оптимізації на сполученнях // Український математичний журнал. 1999. Т. 51, № 8. С. 1118—1121.

44. *Ємець О.О., Ємець Є.М.* Безумовна оптимізація на полірозміщеннях: достатні умови та оцінки мінімумів сильно опуклих цільових функцій // Вісник Запорізького державного університету. Запоріжжя: ЗДУ, 2000. № 1. С. 44—48.

45. *Ємець О.О., Ємець Є.М.* Оцінки та достатні умови мінімуму сильно опуклої функції при її мінімізації на розміщеннях // Волинський математичний вісник. Рівне: РДГУ, 2000. № 7. С. 67—69.

46. *Ємець О., Романова Н., Роскладка О.* Про властивості деяких задач евклідової комбінаторної оптимізації на переставленнях та методи їх розв'язування // Вісник Львів. ун-ту. Сер. приклад. математика та інформатика. 2002. Вип. 5. С. 89—94.

47. *Ємець О.О., Колєчкіна Л.М.* Моделювання деяких прикладних задач оптимізаційними задачами з дробово-лінійною функцією цілі на переставленнях // Волинський математичний вісник. Рівне: РДГУ, 2000. № 7. С. 70—77.

48. *Емец О.А., Роскладка Е.В.* Многоуровневая задача обслуживания как задача евклидовой комбинаторной оптимизации и ее решение // Динамические системы (межвед. науч. сб.). 2001. Вып. 17. Симферополь: Таврич. нац. ун-т. С. 205—209.

49. *Стоян Ю.Г., Ємець О.О., Ємець Є.М.* Оптимізація на полірозміщеннях: теорія та методи. Полтава: РВЦ ПУСКУ, 2005. 103 с.

50. *Яковлев С.В.* Теория выпуклых продолжений функции на вершинах выпуклых многогранников // Журн. вычислит. математ. и матем. физики. 1994. Т. 34, № 7. С. 1112—1119.

51. *Валуйская О.А., Пичугина О.С., Яковлев С.В.* Выпуклые продолжения полиномов на комбинаторных множествах и их приложения // Радиоэлектроника и информатика. 2001. № 2. С. 121—129.

52. *Стоян Ю.Г., Яковлев С.В., Емец О.А.. Валуйская О.А.* Построение выпуклых продолжений для функций, заданных на гиперсфере // Кибернетика и системный анализ. 1998, № 2. С. 27—36.

53. *Валуйская О.А., Емец О.А., Романова Н.Г.* Выпуклое продолжение многочленов, заданных на полиперестановках, модифицированным методом Стояна-Яковлева // Журнал вычислит. математ. и матем. физики. 2002. Т. 42, № 4. С. 591—596.

54. *Стоян Ю.Г., Яковлев С.В., Ємець О.О., Валуйська О.О.* Про існування опуклого продовження функцій, які задані на гіперсфері // Доповіді НАН України. 1998, № 2. С. 128—133.

55. *Valuiskaya O.A., Emotes O.A., Romanova N.G.* Stoyan-Yakovlev's modified method applied to convex continuation of polynomials defined on polypermutations // Computational Mathematics and Mathematical Physics. 2002. Vol. 42, N 42. P. 566—570.

56. *Yakovlev S.* Convex extensions in combinatorial optimization and their applications. Optimization Methods and applications // Butenko S. et al. (Eds.). New York: Springer, 2017. P. 576—584.

57. *Виленкин Н.Я.* Комбинаторика. М.: Наука, 1969. 328 с.

58. *Холл М.* Комбинаторика. М.: Мир, 1970. 424 с.

59. *Стоян Ю.Г., Яковлев С.В., Паршин О.В.* Оптимизация квадратичных функций на множестве перестановок, отображенном в R^n // Доклады АН УССР. Сер. А. 1989. № 5. С. 73—77.

60. *Stoyan Yu., Yaskov G., Scheithauer G.* Packing of various radii solid spheres into a parallelepiped: Central European Journal of Operations Research. 2003. Vol. 11, N 4. P. 389—407.

61. *Стоян Ю.Г., Придатко Д.И.* Упаковка различных круговых цилиндров в параллелепипеде // Доклады НАН Украины. 2004. № 4. С. 27—32.

62. *Стоян Ю.Г., Чугай А.М.* Оптимизация упаковки одинаковых кругов в многосвязную область // Доповіді НАН України. 2004. № 12. С. 64—68.

63. *Чугай А.М.* Метод поиска локальных экстремумов в задаче упаковки одинаковых цилиндров в многоугольной призме с зонами запрета // Радиоэлектроника. Информатика. Управление. 2004. № 2. С. 94—101.

64. *Stoyan Yu., Grebennik I., Kalashnikov V., Lytvynenko O.* Enumeration and Generation of Permutations with a Partially Fixed Order of Elements // International Journal of Combinatorial Optimization Problems and Informatics. 2017. Vol. 8, N 1. P. 19—30.

65. *Гребенник И.В., Литвиненко А.С.* Генерация комбинаторных множеств с заданными свойствами // Кибернетика и системный анализ. 2012. № 6. С. 106—113.

66. *Dyckhoff H., Finke U.* Cutting and Packing in Production and Distribution. Berlin: Physica. Verlag, 1992. 205 p.

67. *Christensen S.G., Rousue D.M.* Container loading with multi-drop constraints // International Transactions on Operational research. 2009. N 16 (6). P. 727—743.

68. *Гребенник И.В., Панкратов А.В., Чугай А.М., Баранов А.В.* Упаковка n-мерных параллелепипедов с возможностью изменения их ортогональной ориентации в n-мерном параллелепипеде // Кибернетика и системный анализ. 2010. № 5. С. 122—131.

69. *Гиль Н.И., Софронова М.С.* Решение задач упаковки n-мерных параллелепипедов для оптимизации выполнения работ на машиностроительных предприятиях // Проблемы машиностроения. 2005. Т. 8, № 4. С. 55—66.

70. *Подиновский В.В., Ногин В.Д.* Парето-оптимальные решения многокритериальных задач. 2-е изд., испр. и доп. М.: ФИЗМАТЛИТ, 2007. 256 с.

71. *Подиновский В.В., Гаврилов В.М.* Оптимизация по последовательно применяемым критериям. М.: Сов. радио, 1975. 192 с.

72. *Стоян Ю.Г., Соколовский В.З.* Решение некоторых многоэкстремальных задач методом сужающихся окрестностей. Киев: Наукова думка, 1980. 208 с.

73. *Стоян Ю.Г.* Об одном обобщении функции плотного размещения // Доклады АН УССР. 1980. № 8. С. 70—74.

74. *Стоян Ю.Г., Гиль Н.И., Муравьева М.С.* Ф-функция n-мерных параллелепипедов // Доповіді НАН України. 2005. № 3. С. 22—27.

75. *Гребенник И.В., Баранов А.В.* Оценки минимума выпуклых функций на классах комбинаторных множеств перестановок // Радіоелектроніка. Інформатика. Управління. 2009. № 1. С. 81—86.

76. *Гребенник И.В., Баранов А.В.* Оптимизация линейных функций с линейными ограничениями на комбинаторных множествах на основе случайного поиска // Искусственный интеллект. 2007. № 1. С. 132—137.

77. *Гребенник И.В., Баранов А.В.* Математическое моделирование и решение некоторых многокритериальных задач размещения параллелепипедов // Сборник научных трудов ХУВС. 2010. № 2 (24). С. 49—55.

78. *Овезгельдыев А.О., Петров Э.Г., Петров К.Э.* Синтез и идентификация моделей многофакторного оценивания и оптимизации. Киев: Наукова думка, 2002. 164 с.

79. *Ovezgeldyev A.O., Petrov K.E.* Fuzzy-Interval Choice of Alternatives in Collective Expert Evaluation // Cybernetics and Systems Analysis. 2016. Vol. 52, N 2. P. 269—276.

К главе 3

1. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.

2. *Танаев В.С.* Декомпозиция и агрегирование в задачах математического программирования. Минск: Наука и техника, 1987. 183 с.

3. *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний одностадийные системы. М.: Наука, 1984. 381 с.

4. *Вагнер Г.* Основы исследования операций. Т. 2. М.: Мир, 1973. 180 с.

5. *Кофман А., Анри-Лабродер А.* Методы и модели исследования операций. Целочисленное программирование. М.: Мир, 1977. 235 с.

6. *Гуляницкий Л.Ф., Сергиенко И.В., Ходзинский А.Н.* О методах дискретной оптимизации для многопроцессорных вычислительных комплексов // Кибернетика. 1988. № 4. С. 26—33.

7. *Васильев В.В., Баранов В.Л.* Моделирование задач оптимизации и дифференциальных игр. Киев: Наука, 1989. 296 с.

8. *Ермольев Ю.М.* Методы стохастического программирования. М.: Наука, 1976. 239 с.

9. *Моисеев Н.Н.* Математические задачи системного анализа. М.: Наука, 1981. 488 с.

10. *Понаморенко В.С., Листровой С.В., Минухин С.В., Знахур С.В.* Методы и модели планирования ресурсов в GRID системах. Харьков: ИД «ИНЖЕК», 2008. 408 с.

11. *Баранов В.Л., Засядьмо А.А., Лістровий С.В., Панченко С.В.* Теоретичні основи проектування комп'ютерно-інтегрованих систем транспортних засобів. Книга 1. Диференціальні перетворення для комп'ютерного моделювання керуючих систем. Харків, 2007. 175 с.

12. *Лістровий С.В., Луханін М.І., Мартинова О.П., Семчук Р.В.* Теорія графів у задачах розподілу ресурсів. Книга 2. Диференціально-ігровий підхід до моделювання систем. Харків, 2007. 143 с.

13. *Баранов В.Л.* Рефлексивные модели конфликтного взаимодействия автоматизированных комплексов радиоэлектронной борьбы и системы военной радиосвязи // Электронное моделирование. 1995. Т. 17, № 4. С. 57—64.

14. *Баранов В.Л.* Моделирование многокритериальных задач управления методом дифференциальных преобразований // Электронное моделирование. 1995. Т. 17, № 1. С. 16—23.

15. *Листровой С.В., Яблочков С.В.* Метод решения задачи определения минимальных вершинных покрытий и независимых максимальных множеств // Электронное моделирование. 2003. Т. 25, № 2. С. 31—40.

16. *Listrovoy S.V., Golubnichiy D.Yu., Listrovaya E.S.* Solution method on the basis of rank approach for integer linear problems with boolean variables // Engineering Simulation. 1999. Vol. 16. P. 707—725.

17. *Listrovoy S.V., Tretjak V.F., Listrovaya A.S.* Parallel algorithms of calculation process optimization for the boolean programming problems // Engineering Simulation. 1999. Vol. 16. P. 569—579.

18. *Listrovoy S.V., Gul A.Yu.* Method of Minimum Covering Problem Solution on the Basis of Rank Approach // Engineering Simulation. 1999. Vol. 17. P. 73—89.

19. Методы моделирования и дискретной оптимизации вычислительных систем реального времени / В.Я. Жихарев, В.М. Илюшко, Л.Г. Кравец, С.В. Листровой, В.С. Харченко; Под ред. В.Я. Жихарева. Харьков—Житомир: ЖГУ, 2004. 494 с.

20. *Кристофидес Н.* Теория графов. Алгоритмический подход. М.: Мир, 1978. 309 с.

21. *Жихарев В.Я., Илюшко В.М., Кравец Л.Г., Листровой С.В., Харченко В.С.*; Под ред. В.Я. Жихарева. Харьков—Житомир: ЖГУ, 2004. 494 с.

22. *Липский В.* Комбинаторика для программистов. М.: Мир, 1988. 203 с.

23. *Шор Н.З.* Квадратичные экстремальные задачи и недифференцируемая оптимизация / Н.З. Шор, С.И. Стеценко. Киев: Наукова думка, 1989. 196 с.

24. *Listrovoy S.V., Gul A.Yu.* Method of Minimum Covering Problem Solution on the Basis of Rank Approach // Engineering Simulation. 1999. Vol. 17. P. 73—89.

25. *Gomory R.* An algorithm for integer solutions to linear programs, Recent Advances in mathematical Programming. N-Y.: McGraw-Hill, 1963.

26. *Листровой С.В., Минухин С.В.* Общий подход к решению задач оптимизации в распределенных вычислительных системах и теории построения интеллектуальных систем // Проблемы управления и информатики. 2010. № 2. С. 47—64.

К главе 4

1. *Минухин С.В.* Модели и методы решения задач планирования в распределенных вычислительных системах. Харьков: Изд-во ООО «Щедрая усадьба плюс», 2014. 324 с.

2. *Minukhin S.V., Zadachin V.M.* Methods of task processing in the computing systems based on the solution of combinatorial optimization problems / Информационные технологии в управлении, образовании, науке и промышленности; Под ред. В.С. Пономаренко. Харьков: Издатель Рожко С.Г., 2016. С. 56—70.

3. *Минухин С.В.* Подходы к организации планирования распределения ресурсов в GRID-системах // Системи обробки інформації. 2009. № 3 (77). С. 49—54.

4. *Листровой С.В., Минухин С.В.* Общий подход к решению задач оптимизации в распределенных вычислительных системах и теории построения интеллектуальных систем // Проблемы управления и информатики. 2010. № 2. С. 47—64.

5. *Листровой С.В., Минухин С.В.* Метод решения задач о минимальном вершинном покрытии в произвольном графе и задачи о наименьшем покрытии // Электронное моделирование. 2012. Т. 34, № 1. С. 29—43.

6. *Листровой С.В., Минухин С.В.* Модель и подход к планированию распределения ресурсов в гетерогенных Грид-системах // Международный на-

учно-технический журнал «Проблемы управления и информатики». 2012. № 5. С. 120—133.

7. *Listrovoy S.V., Minukhin S.V, Znakhur S.V.* Investigation of the Scheduler for Heterogeneous Distributed Computing Systems based on Minimal Cover Method // International Journal of Computer Applications. 2012. Vol. 51, N 19. P. 35—44.

8. *Минухин С.В.* Моделирование и анализ влияния основанных на методе покрытия алгоритмов планирования заданий и распределения ресурсов на производительность гетерогенных распределенных систем // Системи обробки інформації. 2012. Вип. 8 (106). С. 27—32.

9. *Минухин С.В.* Метод планирования пакетов заданий с высокой интенсивностью и выбора ресурсов в распределенных вычислительных системах // Системи обробки інформації. 2015. Вип. 4 (129). С. 38—44.

10. *Листровой С.В., Минухин С.В., Листровая Е.С.* Разработка метода мониторинга распределенной вычислительной системы на основе определения кратчайших путей и кратчайших гамильтоновых циклов в графе // Восточно-Европейский журнал передовых технологий. 2015. Вып. 6/4 (78). С. 32—45.

11. *Минухин С.В.* Модель планирования пакетов заданий в распределенной вычислительной системе // Інформаційні технології: проблеми та перспективи; Под ред. В.С. Пономаренко. Харьків: Вид. Рожко С.Г., 2017. С. 210—220.

12. Nagios — The Industry Standard in IT Infrastructure Monitoring [Electronic resource]. Available at: http://www.nagios.org.

13. Icinga. Open Source Monitoring [Electronic resource]. Available at: https://www.icinga.org.

14. *Минухин С.В.* Информационные технологии реализации двухуровневой модели планирования пакетов заданий в распределенной вычислительной системе на основе решения задачи о наименьшем покрытии // Системи управління, навігації та зв'язку. 2015. Вип. 1 (33). С. 111—115.

15. *Пономаренко В.С.* Методы и модели планирования ресурсов в GRID-системах // В.С. Пономаренко, С.В. Листровой, С.В. Минухин и др. Харьков: ИНЖЭК, 2008. 408 с.

16. *Кофман А.* Методы и модели исследования операций. Целочисленное программирование / А. Кофман, А. Анри-Лабродер. М.: Мир, 1977. 236 с.

17. *Вагнер Г.* Основы исследования операций / Г. Вагнер. М.: Мир. 1973. Т. 2. 489 с.

18. *Bellman R.* Dynamic Programming Treatment of the Travelling Salesman Problem / R. Bellman // Journal of the ACM. 1962. Vol. 9, Iss. 1. P. 61—63. doi: 10.1145/321105.321111.

19. *Held M.* The Traveling-Salesman Problem and Minimum Spanning Trees / M. Held, R.M. Karp // Mathematical Programming. 1971. Vol. 1, Iss. 1. P. 6—25. doi: 10.1007/bf01584070.

20. *Gurevich Y.* Expected computation time for Hamiltonian path problem / Y. Gurevich, S. Shelah // SIAM Journal on Computing. 1987. Vol. 16, Iss. 3. P. 486—502. doi: 10.1137/0216034.

21. *Bjorklund A.* Exact Algorithms for Exact Satisfiability and Number of Perfect Matchings / A. Bjorklund, T. Husfeldt // Algorithmica. 2008. Vol. 52. P. 226—249. doi: 10.1007/s00453-007-9149-8.

22. *Koivisto M.* A space-time tradeoff for permutation problems / M. Koivisto, P. Parviainen // ACM-SIAM Symposium on Discrete Algorithms: Proceedings of the Twenty-First Annual. Austin, 2010. P. 484—492. doi: 10.1137/1.9781611973075.41.

23. *Kohn S.* A generating function approach to the traveling salesman problem / S. Kohn, A. Gottlieb, M. Kohn // ACM'77: Proceedings of the 1977 annual conference. New York, USA, 1977. P. 294—300. doi: 10.1145/800179.810218.

24. *Karp R.M.* Dynamic Programming Meets the Principle of Inclusion and Exclusion / R.M. Karp // Operations Research Letters. 1982. Vol. 1, Iss. 2. P. 49—51. doi: 10.1016/0167-6377(82)90044-x.

25. *Bax E.* A Finite-Difference Sieve to Count Paths and Cycles by Length / E. Bax, J. Franklin // Information Processing Letters. 1996. Vol. 60, Iss. 4. P. 171—176. doi: 10.1016/s0020-0190(96)00159-7.

26. *BjËorklund A.* Determinant Sums for Undirected Hamiltonicity / A. BjËorklund // Foundations of Computer Science: Proceedings of the 51st Annual Symposium. Washington, DC, 2010. P. 173—182. doi: 10.1109/FOCS.2010.24.

27. *Woeginger G.J.* Open Problems Around Exact Algorithms / G.J. Woeginger // Discrete Applied Mathematics. 2008. Vol. 156, Iss. 3. P. 397—405. doi: 10.1016/j.dam.2007.03.023.

28. *Eppstein D.* The traveling salesman problem for cubic graphs / D. Eppstein // Algorithms and Data Structures. Lecture Notes in Computer Science. 2003. Vol. 2748. P. 307—318. doi: 10.1007/978-3-540-45078-8_27.

29. *Iwama K.* An improved exact algorithm for cubic graph TSP / K. Iwama, T. Nakashima // Computing and Combinatorics. Lecture Notes in Computer Science. 2007. Vol. 4598. P. 108—117. doi: 10.1007/978-3-540-73545-8_13.

30. *Gebauer H.* Finding and enumerating hamilton cycles in 4-regular graphs / H. Gebauer // Theoretical Computer Science. 2011. Vol. 412, Iss. 35. P. 4579—4591. doi: 10.1016/j.tcs.2011.04.038.

31. *Могила І.А.* Вплив параметрів мурашиного алгоритму на розв'язок задачі комівояжера / І.А. Могила, І.І. Лобач, О.А. Якимець // Східно-Європейський журнал передових технологій. 2014. Т. 4, № 4 (70). С. 18—23. doi: 10.15587/1729-4061.2014.26290.

32. *Боронихина Е.А.* Сравнение методов решения задачи коммивояжера. Ч. 3: материалы XIII Междунар. науч.-практ. конф. им. А.Ф. Терпугова / Е.А. Боронихина, В.А. Сибирякова // Информационные технологии и математическое моделирование И74 (ИТММ—2014). Томск: Изд-во Том. ун-та, 2014. С. 18—21.

33. *Курейчик В.М.* Об алгоритмах решения задачи коммивояжера с временными ограничениями / В.М. Курейчик, А.В. Мартынов // Информатика, вычислительная техника и инженерное образование. 2014. № 1 (16). С. 1—13.

34. *Частикова В.А.* Разработка и сравнительный анализ эвристических алгоритмов для поиска наименьшего гамильтонова цикла в полном графе / В.А. Частикова, К.А. Власов // Фундаментальные исследования. 2013. № 10. С. 63—67.

35. *Li Y.* A New Exact Algorithm for Traveling Salesman Problem with Time Complexity Interval ($O(n^4)$, $O(n^3*2^n)$) [Electronic resource]. Available at: http://arxiv.org/abs/1412.2437.

36. *Серая О.В.* Анализ методов решения транспортных задач со случайными стоимостями перевозок // Информационно-управляющие системы на железнодорожном транспорте. 2013. № 4. С. 42—45.

37. *Listrovoy S.V.* Method of Minimum Covering Problem Solution on the Basis of Rank Approach / S.V. Listrovoy, Yu. Gul // Engineering Simulation. 1999. Vol. 17. P. 73—89.

38. *Listrovoy S.V.* Solution method on the basis of rank approach for integer linear problems with boolean variables / S.V. Listrovoy, D.Yu. Golubnichiy, E.S. Listrovaya // Engineering Simulation. 1999. Vol. 16. P. 707—725.

ОБ АВТОРАХ

СЕМЕНЕЦ
ВАЛЕРИЙ ВАСИЛЬЕВИЧ
доктор технических наук,
профессор
Сферы научных интересов:
- ✓ автоматизация процессов управления,
- ✓ конструкторское проектирование микроэлектронных устройств,
- ✓ биомедицина, микропроцессорные медицинские системы.

E-mail: valery.semenets@nure.ua

ГРЕБЕННИК
ИГОРЬ ВАЛЕРИЕВИЧ
доктор технических наук,
профессор
Сферы научных интересов:
- ✓ комбинаторика,
- ✓ комбинаторная оптимизация,
- ✓ комбинаторная генерация,
- ✓ математическое моделирование и решение комбинаторных оптимизационных задач геометрического проектирования,
- ✓ комбинаторные задачи маршрутизации.

E-mail: igor.grebennik@nure.ua

ЛИСТРОВОЙ
СЕРГЕЙ ВЛАДИМИРОВИЧ
доктор технических наук,
профессор
Научные интересы были в таких областях:
- ✓ теория построения вычислительных систем и управления телекоммуникационными сетями,
- ✓ методы дискретной оптимизации,
- ✓ методы комбинаторной оптимизации на графовых структурах.

МИНУХИН
СЕРГЕЙ ВЛАДИМИРОВИЧ
доктор технических наук,
профессор
Сферы научных интересов:
- ✓ методы и модели управления распределенными вычислительными системами,
- ✓ методы комбинаторной оптимизации на графах,
- ✓ теория расписаний,
- ✓ методы построения гибридных высокопроизводительных систем на основе современных информационных технологий.

E-mail: minukhin.sv@gmail.com

ОВЕЗГЕЛЬДЫЕВ
АТА ОРАЗГЕЛЬДЫЕВИЧ
доктор технических наук,
профессор
Сферы научных интересов:
- ✓ системный анализ и теория оптимальных решений,
- ✓ автоматизированные системы управления,
- ✓ информационная безопасность.

E-mail: ataovezgeldyyev@gmail.com

ОГЛАВЛЕНИЕ

Наукове видання

СЕМЕНЕЦЬ Валерій Васильович
ГРЕБЕННІК Ігор Валерійович
ЛІСТРОВИЙ Сергій Володимирович
МІНУХІН Сергій Володимирович
ОВЕЗГЕЛЬДИЄВ Ата Оразгельдийович

МОДЕЛІ І МЕТОДИ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ В ПРОЕКТУВАННІ ТА КЕРУВАННІ

(Російською мовою)

Київ, Науково-виробниче підприємство
«Видавництво "Наукова думка" НАН України», 2019

Художнє оформлення *М.А. Панасюк*
Художній редактор *І.П. Савицька*
Технічний редактор *Т.С. Березяк*
Коректор *В.М. Ткаченко*
Оператор *В.Г. Каменькович*
Комп'ютерна верстка *О.І. Фуженко*

Підп. до друку 26.02.2019. Формат 60×90/16. Папір офс. № 1.
Гарн. Таймс. Друк. офс. Ум. друк. арк. 11,0.
Ум. фарбо-відб. 11,5. Обл.-вид. арк. 9,46.
Тираж 200 прим. Зам. № 19-022

Оригінал-макет виготовлено
у НВП «Видавництво "Наукова думка" НАН України»
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготівників
і розповсюджувачів видавничої продукції
ДК № 2440 від 15.03.2006 р.
01601 Київ 1, вул. Терещенківська, 3

ПП «Видавництво "Фенікс"»
03680 Київ 680, вул. Шутова, 13б
Свідоцтво про внесення до Державного реєстру
серія ДК № 271 від 07.12.2000 р.