УДК 519.7
519.863
681.51.015.23

РГАСНТИ 50.03

Левин М. Ш. Применение оптимизационных комбинаторных моделей в автоматизированных системах. — М.; 1986. — 64 с., 10 ил. (Обзорная информация / ВНИИТЭМР. Технология, оборудование, организация и экономика машиностроительного производства. Сер. 9. Автоматизированные системы проектирования и управления. Вып. 1).

В обзоре рассмотрены детерминированные оптимизационные комбинаторные модели, имеющие содержательную интерпретацию и широкие приложения в рамках различных автоматизированных систем проектирования, технологической подготовки производства, эксплуатации инструментального обеспечения ГПС и др. Приведены описания систем поиска наилучших решений типа принятия решений, оптимизационного моделирования, экспертных. Рассмотрены процессы построения, проектирования комбинаторных моделей и эффективных алгоритмов. Приведены описания девяти групп оптимизационных комбинаторных задач (содержательная постановка, формальная постановка, основные подходы к решению, характеристики сложности задач и эффективных алгоритмов, приложения), включая задачи календарного планирования, упорядочения, формирования портфеля заказов, генерации вариантов сложных систем, унификации и др.

Обзор предназначен для специалистов станкостроительной промышленности, работающих в области разработки и организации эксплуатации автоматизированных систем различного типа.

Обзор может быть использован в системе экономического образования специалистов.

Министерство станкостроительной и инструментальной промышленности

Всесоюзный научно-исследовательский институт информации и технико-экономических исследований по машиностроению и робототехнике (ВНИИТЭМР)

ТЕХНОЛОГИЯ, ОБОРУДОВАНИЕ, ОРГАНИЗАЦИЯ И ЭКОНОМИКА МАШИНОСТРОИТЕЛЬНОГО ПРОИЗВОДСТВА

Серия 9. АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ ПРОЕКТИРОВАНИЯ И УПРАВЛЕНИЯ

Обзорная информация

Выпуск 1

М. Ш. ЛЕВИН

ПРИМЕНЕНИЕ ОПТИМИЗАЦИОННЫХ КОМБИНАТОРНЫХ МОДЕЛЕЙ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ

Издается с 1964 г. Выпускается 4 раза в год

Москва 1986

ВВЕДЕНИЕ

Одним из основных направлений повышения эффективности производства, а также внедрения достижений научно-технического прогресса в машиностроении, и в частности в станкостроении, является применение автоматизированных систем (АС). В настоящее время во многих странах разрабатываются крупные проекты ЭВМ пятого поколения, создаются компоненты их программного обеспечения [4, 63, 103]. Значительные усилия направлены на разработку интеллектуальных средств (систем управления базами знаний и непосредственно баз знаний; интеллектуальных интерфейсов; систем: автоматизации конторской деятельности, решения инженерных задач, управления производством, базовых и прикладных экспертных, обеспечения принятия решений и др.).

При этом большое значение имеет интеграция АС разного типа в рамках всего жизненного цикла продукции [26, 41, 89]. Например, система обеспечения функционирования технологического оборудования гибких производственных систем (ГПС) включает: автоматизированные системы научных исследований (АСНИ); системы автоматизированного проектирования (САПР); автоматизированные системы технологической подготовки производства (АСТПП); автоматизированную систему управления предприятием (АСУП); автоматизированные транспортно-складские системы (АТСС); автоматизированные системы инструментального обеспечения (АСТИО); системы автоматизированного контроля (САК)

и др. [34]. Интеграция обеспечивает организацию единого потока информации от проектирования новых изделий до получения готовой продукции [41]. В рамках интеграции АС целесообразно создание общих базовых интеллектуальных средств, в частности систем формирования и выбора наилучших плановых, управленческих, проектных решений. Это отвечает практическим потребностям. Так, например, отмечается, что большая доля неповторяющихся деталей при функционировании гибких автоматизированных производств обусловливает необходимость создания специального программного обеспечения для выполнения задач разработки оптимальных графиков работ с учетом колебаний в загрузке каждого станка и подаче деталей; автоматизированного проектирования технологического процесса; оптимизации маршрутов обработки и обеспечения максимальной загрузки оборудования, а также транспортировки инструмента, компоновки магазинов и др. [14].

Современные сложные системы (организационно-технические, организационно-экономические) часто имеют модульную, дискретную структуру. Это обусловливает применение комбинаторного моделирования процессов управления, планирования, проектирования. Комбинаторные модели, т. е. разделение, распределение, упорядочение некоторого набора объектов, естественным образом возникают в процессе подготовки и принятия решений, при выборе, упорядочении альтернативных вариантов решений, при проведении экспертного оценивания, организации человеко-машинных процедур. При реализации АС возникают проблемы рационального использования различных видов ресурсов, в частности вычислительных (производительности, памяти), в виде задач планирования работы вычислительных систем, организации памяти, что также связано с решением оптимизационных комбинаторных задач.

При использовании комбинаторных моделей вследствие их сложности особенно важны тщательный анализ и разумная декомпозиция этапов построения (выбора) моделей, разработки стратегий решения, алгоритмов. Неудачи на этих этапах часто невозможно компенсировать на других этапах разработки автоматизированных процедур, например при непосредственном программировании.

В обзоре рассмотрены различные типы АС поиска наилучших решений: поддержки принятия решений (СППР) или обеспечения принятия решений (СОПР); оптимизационного моделирования (СОМ); экспертных (ЭС). Приведены уровни архитектуры, структурно-функциональные схемы систем подобного типа, рассмотрена роль оптимизационных комбинаторных моделей (или комбинаторных задач, их постановок).

Особое внимание уделяется этапам построения комбинаторных моделей и алгоритмов, подходам к построению (выбору) эффективных алгоритмов решения на основе имеющихся ресурсов (вычислительных, трудовых, требований к точности и др.). Эти этапы

часто являются определяющими в рамках всего процесса проектирования технологии решения задач на базе АС.

Число комбинаторных моделей чрезвычайно велико. Некоторые классы задач хорошо исследованы (например, о потоках в сетях [1]). В обзоре рассматривались детерминированные комбинаторные модели, имеющие содержательную интерпретацию и широкие приложения, в частности календарного планирования, упорядочения, формирования портфеля заказов, генерации вариантов сложных систем, упаковки, коммивояжера, унификации и др.

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ ПОИСКА НАИЛУЧШИХ РЕШЕНИЙ

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ И ОПТИМИЗАЦИОННЫЕ КОМБИНАТОРНЫЕ МОДЕЛИ

При обработке данных в рамках информационной технологии (с использованием различных средств вычислительной техники) выделяются следующие процессы [19, 64, 80]:

создание и модификация данных (сбор, ввод, входной контроль, хранение и др.);

использование данных (включая поиск, генерацию документов, сводок, таблиц, проведение анализа).

Большое значение в настоящее время приобретают интеллектуальные средства обработки данных типа формирования и поиска наилучших решений (плановых, управленческих, проектных). При их разработке рассматриваются различные системы: оптимизационного, а также имитационного моделирования; поддержки (обеспечения) принятия решений; экспертные; обеспечения управленческих решений и др. [3, 4, 10, 11, 15, 19—21, 24, 39, 40, 44, 49, 50, 75, 76, 87, 101, 108, 111—114]. Существенную роль при разработке и использовании указанных систем играют комбинаторные оптимизационные модели.

Результаты исследований в области оценки эффективности применения АС показывают, что эффект (совокупные результаты) в основном связан с решением задач поиска наилучших решений, с внедрением оптимизационных моделей [71]. Указанные задачи можно условно разделить на внешние (глобальные) и внутренние (локальные). Внешние соответствуют проблемам, поставленным потребителями АС, и направлены на повышение эффекта, например формирование наилучших плановых и проектных решений. Внутренние задачи направлены на рациональную организацию работы АС (организация данных, планирование работы вычислительных комплексов, т. е. рациональное использование ресурсов). Решение внутренних задач позволяет, как правило, снизить затраты на создание и эксплуатацию АС. При этом в основном используются комбинаторные модели, что связано с дискрет-

ным характером вычислительного процесса, взаимодействия человека с ЭВМ. Решение внешних оптимизационных задач вследствие дискретного характера большого числа хозяйственных процессов, например календарного планирования, многовариантного проектирования, выбора оптимальных типоразмерных рядов изделий, также обусловливает применение оптимизационных комбинаторных моделей в значительном числе случаев.

СХЕМА ПРИНЯТИЯ РЕШЕНИЙ. АРХИТЕКТУРА СИСТЕМ

Рассмотрим (не претендуя на общность) типовые задачи принятия решений следующего типа [10, 11, 13, 18, 22, 28, 38, 40, 78].

Имеется исходное множество альтернатив (объектов, вариантов, решений и др.). Требуется осуществить:

выделение лучшей альтернативы;

упорядочение альтернатив;

выделение подмножества лучших альтернатив, удовлетворяющего некоторым ограничениям (например, на мощность подмножества, на функции от характеристик альтернатив);

классификацию альтернатив, включая разбиение множества альтернатив на классы, удовлетворяющие некоторым ограничениям (например, на функции от характеристик альтернатив), разбиение альтернатив на взаимно упорядоченные, на заданное число классов.

Следует отметить, что основными результатами решения практических задач могут быть следующие [10]:

формирование общего языка общения для различных групп ЛПР, экспертов;

согласование мнений, взглядов различных групп ЛПР и экспертов;

выявление истинных целей решения.

Общая схема решения задач принятия решений имеет вид [9, 10, 13, 18, 22, 38]:

этап 1. Анализ исследуемой системы, процессов, явлений и выявление проблем;

этап 2. Постановка задачи (структуризация);

этап 3. Получение исходных данных (измерение, оценивание);

этап 4. Решение задачи (с помощью математических методов, ЛПР, экспертов, вычислительной техники);

этап 5. Анализ решения.

К этапу 2 возможен переход от этапов 4 и 5 с целью уточнения постановки. Как указывается в работе [68], процесс решения задачи часто заключается в уточнении исходной постановки. При решении задачи (этап 4) может потребоваться дополнительное измерение, оценивание параметров задачи (переход к этапу 3) [13, 22, 38]. Дополнительно введен этап 1, который часто является определяющим.

На этом этапе проводятся следующие основные действия с выявлением:

уровней рассмотрения, элементов и структуры исследуемой системы (процесса, типов связей);

основных видов ресурсов, критериев качества функционирования;

подсистем, используемых ими ресурсов и критериев качества функционирования подсистем;

основных противоречий (проблем), узких мест, ресурсных ограничений.

Примерная схема этапов 2—5 представлена на рис. 1. В последние годы в широких масштабах проводятся разработка и внедрение СППР или СОПР, которые реализуют приведенную схему решения или ее фрагменты [10, 11, 40, 76, 101, 108, 111—113]. Можно рассматривать следующие основные уровни архитектуры подобных систем [10, 11].

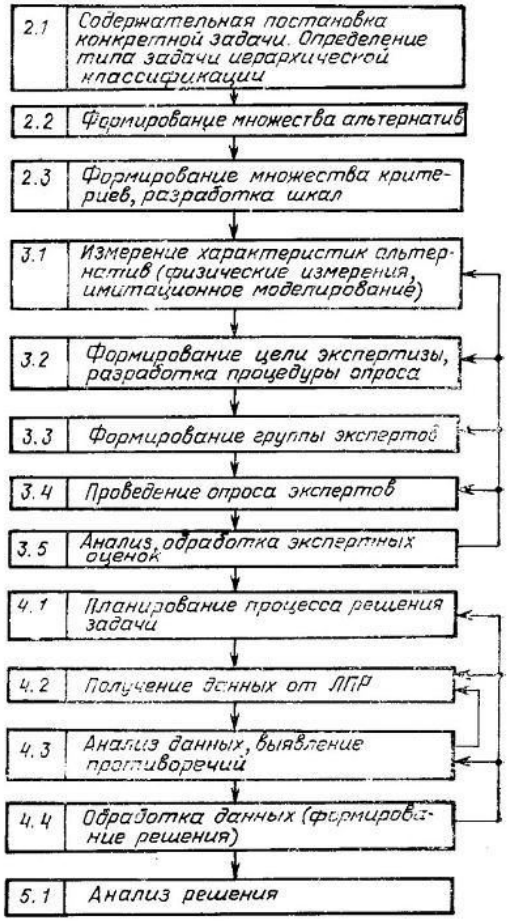


Рис. 1. Фрагменты схемы решения

1. Целевой уровень. Этот уровень соответствует проблемной области и характеризует взаимосвязь понятийной базы (терминологии), целей и задач принятия решений.

2. Уровень постановок задач принятия решений.

3. Уровень процедур, включающий общие (глобальные) процедуры решения, т. е. схемы решения, стратегии [10, 11, 56, 69]. К этому уровню могут быть отнесены процедуры организации экспертиз, обработки результатов измерений характеристик альтернатив, обработки экспертных оценок, организации диалога с экспертами и ЛПР.

4. Уровень формальных моделей. Среди экстремальных моделей особенно актуальны, как показывает практика, комбинаторные оптимизационные модели, в том числе на графах. Такие мо-

дели могут возникать не только как средство формализации исходных задач, но и в рамках различных процедур, например при формировании групп экспертов, вопросов, а также при обработке ответов ЛПР.

5. Алгоритмический уровень, включающий алгоритмы для решения формальных постановок, интерактивные процедуры.

6. Программный уровень, включающий, например, модули для решения формальных оптимизационных задач, модули организации диалога, ведения баз данных и знаний.

7. Уровень ресурсов. На этом уровне в качестве ресурсов можно рассматривать: вычислительные ресурсы (например, производительность и память вычислительных систем); трудозатраты экспертов и ЛПР; затраты на измерение характеристик альтернатив, на организацию экспертизы, на имитационное моделирование.

СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

Процесс реализации СППР обладает рядом особенностей [10, 11, 40, 46, 52].

Типовые постановки задач принятия решений можно рассматривать в двух аспектах: как глобальные, когда постановка соответствует практической задаче принятия решений, и как локальные, когда они представляют собой некоторый фрагмент в рамках общей схемы решения.

Практическое внедрение СППР ставит вопрос об экономном использовании в процессе принятия решений имеющихся ресурсов (временных, вычислительных, трудозатрат ЛПР и экспертов). Это обусловливает соответствующие требования к используемым процедурам, алгоритмам и программам.

Реализация СППР требует организации человеко-машинных процедур с учетом всех возникающих сложностей (ограничений на возможности человека и на время реакции систем для обеспечения диалогового режима в реальном времени; вопросов языка общения и формы представления данных; знаний; учета уровня и особенностей ЛПР и экспертов и др.).

СППР могут включать СОМ. Последние могут использоваться на различных этапах принятия решений, например при формировании множества альтернатив, при организации экспертизы, обработке экспертных оценок и др. [13, 61, 66, 72].

Следует отметить, что участие человека в процессе решения задач обусловлено двумя основными факторами:

при решении слабоструктуризованных задач человек уточняет исходные данные, постановку задачи, формирует решающие правила и др.;

при решении формализованных задач введение человека в контур решения часто связано с вычислительной сложностью исходной постановки, когда участие человека позволяет реализовать экономную процедуру решения.

Участие человека в процессе решения обусловливает важность вопросов общения человека и ЭВМ, причем более общей

представляется проблема выделения фрагментов с участием человека и формальных фрагментов в процессе решения. Это связано с уровнем процедур, декомпозицией исходных задач, построением итеративных многошаговых схем принятия решений. Представляется возможной разработка некоторых типовых схем решения, но для этого желательно наличие готовых фрагментов в виде локальных задач принятия решений, формальных моделей, решаемых на базе эффективных алгоритмов, человеко-машинных процедур. Процесс построения (выбора) схем решений для конкретных задач может основываться на использовании экспертных систем (ЭС). Их наличие желательно на различных этапах принятия решений: при анализе проблемной ситуации; постановке (выборе) задачи; организации оценивания (выборе типа, методов и др.); при решении задач (построении схем решения, стратегий решения, выборе алгоритмов и человеко-машинных процедур), а также при анализе решения. Аналогичная ситуация может иметь место и в рамках СОМ.

При исследовании типовых операций по переработке информации с участием человека [40, 52] выделяются четыре типовые операции переработки информации человеком (по типу объектов): операции с критериями, с оценками альтернатив по критериям, с альтернативами, с переменными. При этом целесообразно учитывать следующие дополнительные факторы [11]:

число представляемых человеку объектов;

вид шкалы, в которой представляется информация или должен быть представлен ответ;

тип вспомогательной информации.

Макроструктуру СППР с точки зрения построения программного обеспечения можно представить в виде трех уровней: управления, функционального и поддержки [60]. Кроме того, рассматриваются специализированные блоки (СБ), которые представляют различного типа пакеты, библиотеки, программные модули и используются на различных уровнях программного обеспечения. Под СБ в данном случае понимаются системы различного назначения (оптимизационные, экспертные, имитационные, диалоговые, информационные). Функциональный уровень может включать набор блоков, реализующих подэтапы схемы принятия решений. Функциональный блок может включать ЭС. На уровне поддержки используются библиотеки и пакеты, реализующие СОМ, системы имитационного моделирования, системы проведения экспертиз, сервис для обеспечения интерактива и библиотеки человеко-машинных процедур, СУБД для управления базами данных и знаний.

СИСТЕМЫ ОПТИМИЗАЦИОННОГО МОДЕЛИРОВАНИЯ

Системы оптимизационного моделирования (СОМ) являются сложными программными комплексами. Они состоят из большого числа модулей, общий объем которых может составлять сотни

тысяч команд на Ассемблере. СОМ обычно включают в свой состав:

оптимизационные программы;

СУБД;

средства формирования, корректировки моделей;

средства анализа результатов оптимизационных расчетов;

средства отображения результатов (промежуточных, окончательных).

В качестве эксплуатационных («промышленных») характеристик пакетов оптимизации выделяют [15, 39]:

надежность;

трудоемкость и устойчивость решения;

разнообразие решаемых задач (включая размерность задач);

наличие развитого сервиса.

СОМ должны удовлетворять различным, часто противоречивым требованиям, например [15, 21, 39, 75, 87, 91]:

полнота функций, необходимых при моделировании (взаимодействие с базой данных, формирование и корректировка моделей, решение задач оптимизации и постоптимизационного анализа, отображение результатов моделирования и др.) и объединение общим управлением;

гибкость, адаптация к виду решаемой задачи;

удобство пользователя и обеспечение диалогового режима работы;

эффективность и обеспечение высоких эксплуатационных характеристик всех элементов;

возможность расширения (открытость), развития и совершенствования отдельных элементов;

возможность встраивания системы в общую технологию обработки информации в конкретной автоматизированной системе.

Общие требования, предъявляемые к СОМ, обусловливают ряд требований к элементам (модулям) [15, 74]:

эффективность программной реализации (трудоемкость решения, объем требуемой оперативной памяти, точность и устойчивость счета и др.);

согласованность модулей по входам и выходам;

возможность использования на разных ЭВМ.

Современные СОМ обладают высокими эксплуатационными характеристиками, но в их развитии наметились противоречия, например [15]:

стремление к универсализму, сложность архитектуры СОМ приводят к излишней громоздкости, недостаточной гибкости;

требование высоких эксплуатационных характеристик и большие затраты на их достижение приводят к тому, что многие интересные алгоритмы не реализуются в промышленных пакетах.

Среди направлений развития СОМ выделяются тенденции укрупнения, усложнения, повышения эффективности и расширения возможностей отдельных программных средств (модулей) поиска оптимальных решений [15, 39]. Развитие СОМ в последнее время

соответствует переходу от задач (методов) исследования операций к задачам (методам) системного анализа и заключается в разработке на основе универсальных пакетов программ по оптимизации гибких интерактивных систем оптимизационного моделирования с поддерживающей их элементной (модульной) базой [15, 87].

В процессе функционирования СОМ на различных этапах (генерации моделей, оптимизации, анализа решений) возникают задачи определения стратегии решений типа выбора (подбора) моделей, схем решения, алгоритмов оптимизации, процедур анализа решений. В простейших случаях решение таких задач может основываться на простых инструкциях и правилах. В связи с усложнением СОМ, расширением сферы их применения, повышения их универсальности, усложнением проблемных ситуаций возникает необходимость в использовании для этих целей специальных автоматизированных блоков [87]. В работе [77] рассматривается переход от одной программы оптимизации к другой на основе применения детерминированных автоматов, вероятностных автоматов с переменной структурой. При определении стратегии решения задач используются знания специалистов в виде эмпирических правил и процедур. Это обусловливает возможность применения на различных этапах определения стратегии решения экспертных систем.

Важным аспектом функционирования СОМ является настройка на проблемную область применения. Это может осуществляться за счет наличия специальных средств настройки на проблемную область: специализации входного языка системы, перестройки массивов и модулей обработки данных. При построении входного языка целесообразно учитывать следующие принципы [87]:

первоначальный входной язык («язык стержень») не должен содержать элементов, отражающих проблемную специфику решаемых задач;

предусматривать возможность конструирования в языке новых операторов, средств представления информации и введения в язык различных служебных слов, связанных со спецификой обрабатываемых данных и класса задач, что обеспечивает настройку системы на проблемную область;

после проблемной ориентации системы, практического использования во входном языке фиксировать новые элементы, образуя тем самым язык-окаймление.

На рис. 2 представлена трехуровневая структура программного обеспечения СОМ, включающая ЭС для реализации «творческих этапов» при определении стратегии решения и средства настройки на проблемную область [49, 60, 75].

Следует отметить, что в области создания СОМ наметилась тенденция, соответствующая ситуации, имеющей место при разработке автоматизированных систем других типов, например информационных.

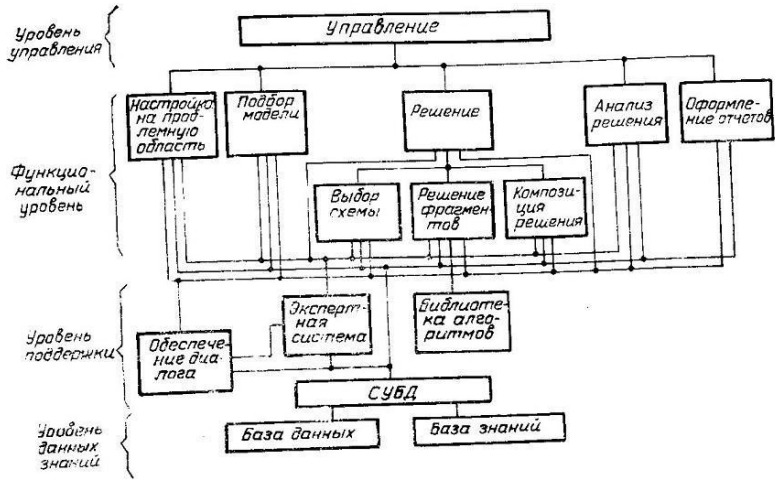


Рис. 2. Структура программного обеспечения СОМ

Процесс создания проблемно-ориентированных СОМ может базироваться на различных специализированных методо-ориентированных пакетах. В результате создаются пакеты программ, называемые в работе [87] универсально-специализированными. Такой подход, в частности, реализован в диалоговых маршрутных системах (ДМ-системах) [50, 51].

ЭКСПЕРТНЫЕ СИСТЕМЫ

В последнее время широко разрабатываются и используются экспертные системы (ЭС). Они находят применение в различных приложениях [3, 20, 24, 41, 44, 111]:

медицинская диагностика и выбор схем лечения;

диагностика ошибок в программном обеспечении, схем повреждений в электронике, сложных организационно-технических системах;

выбор вариантов размещения компонентов вычислительных систем, средств финансирования;

построение эконометрических моделей, поиск информации.

Типовыми задачами, для которых применяются ЭС, являются различного вида диагностика, исследование сложных систем и процессов (организационно-технических, социально-экономических, экологических), поиск информации.

В основном применение ЭС характеризуется двумя свойствами. Во-первых, в указанных приложениях важная роль отводится эвристическим знаниям и подходам в целом. При этом часто невозможно точно поставить задачу или сформулировать правила ее решения, не удается или нецелесообразно решать ее чисто формальными методами. Это имеет место на различных этапах принятия решений (в первую очередь при анализе ситуаций и выявлении проблем, постановке задач, выборе подходов, методов, алгоритмов решения, анализе полученных решений), при формулировании знаний экспертами. Во-вторых, часто решение задачи на основе формального подхода требует такого количества информации, что получение и обработка ее в полном объеме человеку утомительны и трудоемки или вообще невозможны ни для человека, ни для вычислительной системы (по экономическим ограничениям).

В упрощенном виде типовую задачу, решаемую с помощью ЭС, можно представить так. Имеется множество (пространство) входов (неких исходных параметров, симптомов в медицине, параметров состояния технической системы и др.) V, множество (пространство) решений (например, диагнозов, схем лечения, типов методов или алгоритмов и др.) U. Оба множества в значительной степени «размыты», определены неточно в целом или для рассматриваемой конкретной прикладной задачи. Таким образом, требуется: уточнение пространства входов и пространства решений; определение и реализация правил отображения конкретной входной ситуации на пространство решений.

Процесс решения задачи обычно основывается на двух подходах [3]:

исходя из гипотезы о возможном решении в направлении обоснования истинности гипотезы;

исходя из входных данных в направлении поиска решения.

Решение задач связано с использованием базы знаний в виде фактов (декларативных знаний) и методов решения проблемы (процедурных знаний). Декларативные знания организуются в виде фрейм-структур, метаправил, таблиц, словарей. Процедурные знания включают принципы распознавания образов (геология, математика, диагностика ошибок в программном обеспечении) [3] и организуются в основном в виде семантических сетей. При этом в вершинах находятся определенные объекты (понятия) из области применения ЭС, а связи между вершинами выражают реальные логические отношения этих объектов. Процедурные знания представляют функциями, подпрограммами, фрейм-процедурами, продукциями.

В процессе создания и эксплуатации ЭС целесообразно выделить два основных режима:

накопление знаний;

непосредственно эксплуатация.

Очевидно, что накопление, расширение знаний может осуществляться в процессе эксплуатации. В некоторых ЭС эксплуатация не требует больших затрат (трудовых, вычислительных и др.), в то время как накопление знаний чрезвычайно трудоемко и требует значительных вычислительных ресурсов. Как правило, оно включает:

получение информации от экспертов (на базе интерактивных процедур);

обработку знаний, в том числе аппроксимацию и расширение знаний [20];

организацию и ведение баз знаний.

К основным проблемам, возникающим в процессе создания и эксплуатации ЭС, относят [111]:

комбинаторную сложность пространств входа и решений;

трудности представления знаний (фактография проблемной области, эмпирические правила и процедуры, типовые проблемные ситуации и соответствующие им эвристики, глобальные стратегии поиска решений);

вопросы экономической эффективности ЭС;

сложности реализации интерактива (как в процессе накопления знаний, так и при эксплуатации) с учетом уровня, квалификации, ориентации эксперта, пользователя.

Эффективность создания и функционирования ЭС в значительной степени связана с успешным решением возникающих комбинаторных задач. Это относится и к глобальным, внешним ситуациям типа сужения, представления пространств входа и решений, представления знаний, рациональной организации интерактива, и к внутренним проблемам типа рациональной организации данных, экономных алгоритмов обработки, аппроксимации знаний и др.

При использовании ЭС выделяются тенденции их применения в рамках имеющих широкое распространение СОМ, СИМ, на этапах выбора моделей, стратегий решения прикладных задач, анализа получаемых решений. Это распространяется и на СППР. Очевидно, что в рамках ЭС возникают локальные задачи типа принятия решений, комбинаторные оптимизационные модели.

ПРИЕМЫ ПОСТРОЕНИЯ ЭФФЕКТИВНЫХ АЛГОРИТМОВ

СЛОЖНОСТЬ КОМБИНАТОРНЫХ МОДЕЛЕЙ

Долгое время для решения комбинаторных задач в основном использовали различные алгоритмы типа перебора. Положение изменилось в связи с расширением сферы применения вычислительных систем, резким увеличением доли вычислений комбинаторного характера и решением задач большей размерности. Эффективно построенные алгоритмы позволяют получить существенную экономию вычислительных ресурсов (производительность и память ЭВМ) при решении задач средней и большой размерности. Это обусловило проведение многочисленных исследований, направленных на построение эффективных алгоритмов, анализ классов задач и алгоритмов.

Как отмечается в работах [36, 93], Эдмондс был, очевидно, первым, кто выделил «хорошие» алгоритмы, время решения задачи которых в худшем случае ограничено полиномом от размера

входа (размерности задачи — длины двоичной записи всех исходных данных). «Хорошие» алгоритмы стали называть эффективными. Алгоритмы, временная сложность которых не поддается подобной оценке, называются экспоненциальными (это определение включает и такие функции, как $n^{\log n}$, где n — размерность задачи) [36]. При большой размерности задач различие в трудоемкости (времени решения) алгоритмов особенно заметно.

Большинство экспоненциальных алгоритмов представляет собой варианты полного перебора, в то же время полиномиальные алгоритмы можно построить только тогда, когда удается глубоко проникнуть в суть решаемой проблемы. «Хорошие» алгоритмы обычно имеют трудоемкость $O(n^3)$ и лучше. Для некоторых задач известны экспоненциальные алгоритмы, которые хорошо себя зарекомендовали на практике в подавляющем числе случаев применения. Такими алгоритмами являются, например, симплекс-метод для решения задач линейного программирования, метод ветвей и границ и метод динамического программирования для задачи о рюкзаке. Но такие примеры не часты. Задачу, для которой не существует полиномиального алгоритма, называют труднорешаемой.

Однако задача сводится к другой, если метод решения второй задачи можно преобразовать в метод решения первой. Если подобное преобразование можно осуществить за полиномиальное время, то сводимость называется полиномиальной. Класс задач, для которых существуют эффективные алгоритмы, обозначают P. Класс всех переборных задач (в переборных задачах, как правило, имеется конечное множество вариантов, среди которых нужно найти решение) обозначают NP. Неформально—это класс всех задач распознавания, которые можно решить за полиномиальное время с помощью недетерминированного алгоритма. Под задачей распознавания понимается такая постановка, в которой для каждого набора исходных данных нужно дать ответ типа «да» или «нет». Например, задача о коммивояжере может быть представлена в виде задачи распознавания в следующей формулировке: имеется набор городов и известны расстояния между ними. Требуется ответить на вопрос: существует ли маршрут, проходящий через все города не более, чем по одному разу, длина которого не больше некоторой величины b? Несложно показать, что задачу коммивояжера в оптимизационной постановке (определение маршрута с наименьшей длиной) можно за полиномиальное число шагов свести к соответствующей задаче распознавания. Недетерминированность алгоритма неформально можно пояснить следующим образом [79]. Пусть некоторый алгоритм производит вычисления до тех пор, пока не возникнет ситуация, при которой должен быть сделан выбор из нескольких альтернатив. В этом случае детерминированный алгоритм исследует одну альтернативу, а потом возвращается для исследования других альтернатив, т. е. исследование альтернатив производится последовательно. Недетерминированный алгоритм может исследовать все альтернативы одно-

временно, «копируя», в сущности, самого себя для каждой альтернативы. Все копии работают параллельно и независимо. Эти копии могут создавать дальнейшие копии и т. д. Если одна из копий находит решение, то остальные прекращают работу.

Труднорешаемые задачи распознавания свойств, полиномиально сводимые одна к другой (Куком для этих целей была введена эталонная задача о выполнимости [5, 36, 79, 94]), образуют класс NP-полных (универсальных) задач. Задача называется NP-трудной, если детерминированный полиномиальный алгоритм ее решения можно использовать для получения детерминированного полиномиального алгоритма для каждой задачи из NP. Входящие в NP-класс NP-трудные задачи и составляют класс NP-полных задач. Задача коммивояжера в оптимизационной постановке является NP-трудной.

NP-трудными являются многие известные дискретные задачи. В неформальных терминах это означает, что либо все данные задачи неразрешимы за полиномиальное время, либо если хотя бы одну из них можно решить за полиномиальное время, то любая универсальная задача может быть решена также за полиномиальное время. Широкое распространение имеет гипотеза о том, что эти задачи неразрешимы.

КЛАССИФИКАЦИЯ АЛГОРИТМОВ

При построении, выборе алгоритмов решения дискретных задач разработчик оперирует набором некоторых ресурсов, включающим разнообразные трудозатраты человека, включенного в контур решения задачи (вычислительные, затраты на имитационное моделирование и др.). Кроме того, у него имеется некоторый условный ресурс: погрешность решения задачи. Погрешность может быть абсолютной, относительной и комбинированной (когда по одним параметрам допускается абсолютная погрешность, а по другим — относительная). Условным ресурсом является также степень гарантированности использования перечисленных ресурсов. Например, для эффективных алгоритмов гарантируется полиномиальная трудоемкость для наихудшего случая, для некоторых экспоненциальных алгоритмов — полиномиальная трудоемкость в подавляющем большинстве случаев, а также в среднем (например, симплекс-метод для задач линейного программирования). Аналогично может гарантироваться некий порог, ограничивающий погрешность получаемого алгоритмом решения для всех случаев исходных данных, для большинства случаев и др. Таким образом, разработчик имеет возможность манипулировать имеющимся набором перечисленных ресурсов (включая свою способность разбираться в сути физики задачи) с целью построения алгоритма с приемлемыми характеристиками (с учетом затрат на разработку и обоснование алгоритма).

Построение алгоритмов основывается на применении общих принципов и методов, среди которых широко известны, например, эвристические процедуры, методы: ветвей и границ, динамического

программирования, отсечений. При использовании переборных методов делается попытка максимального сокращения объема перебора, хотя при этом, как правило, и признается неизбежность экспоненциального времени работы в худшем случае [36]. Эвристические алгоритмы обычно имеют полиномиальную трудоемкость и основываются на различных разумных, но не всегда достаточно обоснованных соображениях. Такие алгоритмы требуют большой работы по «доводке». Общие принципы, по сути, являются основой и при построении эффективных алгоритмов, позволяющих получать достаточно хорошие решения, например точные, приближенные с гарантированной погрешностью.

Классификация алгоритмов по трудоемкости и точности проводится следующим образом [5, 25, 35, 36, 68, 73, 79, 88, 94, 97]. По точности алгоритмы разделяются на точные и приближенные (с аналитическим или экспериментальным обоснованием погрешности). Обоснование погрешности решения может быть непосредственным или статистическим. При этом рассматривается погрешность (абсолютная или относительная) по целевой функции, по ограничениям на ресурсы, по графовым ограничениям. Пусть T — вычислительная трудоемкость алгоритма (например, число операций); $P(x)$ — полином от x; n — характеристика размерности задачи (длина двоичного представления исходных данных или, в упрощенном виде, число объектов, элементов, переменных и др.); m — дополнительный параметр, характеризующий размерность (число ограничений и др.); ε — относительная погрешность решения по целевой функции; δ — относительная погрешность решения по ограничению (ресурсу).

Классификация алгоритмов по вычислительной трудоемкости осуществляется по виду оценки алгоритма (для наихудшего случая или в среднем, аналитически или экспериментально). При аналитической оценке выделяют четыре основных случая: полиномиальные алгоритмы ($T \sim P(n)$); псевдополиномиальные ($T \sim n^{P(m)}$); экспоненциальные ($T \sim e^{P(n)}$); с гарантированной погрешностью. Среди последних различают ε-приближенные ($T \sim n^{P(1/\varepsilon)}$), полиномиальные ε-приближенные ($T \sim P(1/\varepsilon)P(n)$), полиномиальные ($\varepsilon$, δ)-приближенные ($T \sim P(1/\varepsilon)\ P(1/\delta)\ P(n)$) и др. Аналогично учитывается требуемый при работе алгоритма объем оперативной памяти. В связи с широким распространением в последнее время человеко-машинных процедур часто рассматривается «оракульная» трудоемкость (число обращений к специалистам в процессе решения задачи — информационная сложность [68]).

Разумная декомпозиция исходной задачи, рациональное сочетание различных подходов к построению алгоритмов, процедур решения могут обеспечить приемлемую точность, трудоемкость, гарантированность качества решения и др. Рис. 3 иллюстрирует направления повышения эффективности методов решения комбинаторных задач [88, 98].

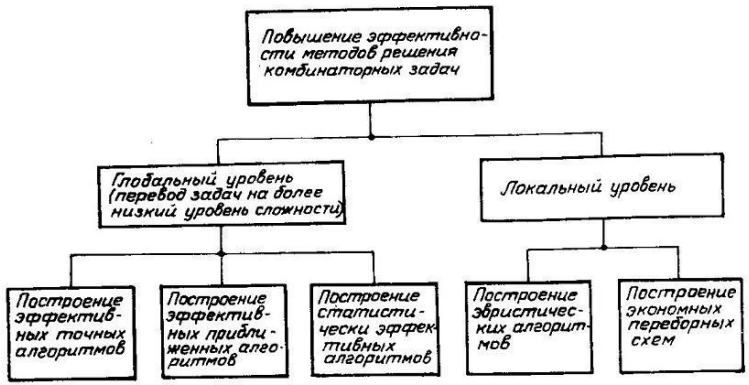


Рис. 3. Направления повышения эффективности методов решения комбинаторных задач

ПРИНЦИПЫ ПОСТРОЕНИЯ ОБЩИХ СХЕМ АЛГОРИТМОВ

Среди принципов построения общих схем алгоритмов, очевидно, центральное место занимает метод частных целей [35], или «разделяй и властвуй» [5]. Метод заключается в сведении исходной задачи к последовательности более простых задач (подзадач). Иногда используется более сложная, чем последовательность, комбинация простых задач. Естественно, что решение исходной задачи должно получаться из решений подзадач. При этом желательно, чтобы подзадачам соответствовали известные и достаточно простые модели. Примерами применения метода частных целей является: решение задачи компоновки программных модулей при организации древовидной оверлейной структуры [54], решение задачи многокритериальной упаковки объектов [99]. В первом случае в качестве простых моделей используется несколько вариантов задачи о рюкзаке и ε-приближенного алгоритма ее решения, во втором — варианты задачи упаковки в контейнеры и алгоритмы их решения. Рассматриваемый метод представляет собой декомпозицию исходной задачи и композицию ее решения из решений подзадач. Это является, по сути, реализацией основных принципов системного анализа. Правда, такой подход не всегда легко применить к конкретной задаче. В работе [35] отмечается, что осмысленное выделение простых подзадач — скорее дело искусства, интуиции. Это требует глубокого проникновения в суть исходной задачи (учет «физики» задачи). При разбиении исходной задачи важным является поддержание равновесия, т. е. разбиения на подзадачи равных размеров (принцип балансировки) [5].

Значительно легче формализуемыми принципами построения общих схем алгоритмов являются два общих метода: метод поиска с возвращением и метод решета [79]. Первый заключается в постоянной попытке расширения частичного решения. На каждом

возможно, происходит возврат к более короткому частичному решению и опять повторяется попытка продолжить его. По сути проводится выделение, продолжение хороших решений. Широко известные методы ветвей и границ являются разновидностью поиска с возвращением с ограничениями [79]. Эти методы представляют собой направленный перебор с отсеиванием «бесперспективных» вариантов [58]. Каждый конкретный вариант метода ветвей и границ характеризуется способом организации перебора (ветвления) и способом оценки перспективности варианта (границы). Для методов ветвей и границ характерна определенная гибкость, позволяющая учитывать специфику конкретной дискретной задачи и использовать в рамках этого подхода другие методы решения [48, 58, 87].

Вариантом поиска с возвращением является и метод динамического программирования. Применение этого метода для решения дискретных задач означает использование декомпозиции: сначала находятся решения подзадач, затем на их основе — решения больших подзадач и т. д. до решения исходной задачи [79]. В основе этого метода лежат принцип оптимальности Р. Беллмана и соответствующее ему функциональное уравнение. На их основе решение многомерной задачи сводится к решению последовательности одномерных. Как отмечается в работе [58], в рамках данного метода трудно учесть специфику конкретной дискретной задачи, что является естественным ограничением его использования. Обобщением метода динамического программирования является более универсальный метод последовательного анализа вариантов, предложенный В. С. Михалевичем [65].

При использовании метода решета рассматривается некоторое конечное множество и исключаются не представляющие интереса элементы [79]. Этот метод является логическим дополнением к методу поиска с возвращением, выделяющим представляющие интерес элементы.

При использовании метода частных целей важными являются предварительный анализ исходных данных, их дифференциация и выделение типовых, стандартных ситуаций. Для некоторых вариантов исходных данных могут храниться готовые решения, для других — использоваться простые модели. Ярким примером такого подхода являются шахматные программы, в которых используются различные заготовки, например дебютные. Такой подход может быть реализован в виде логических таблиц решений [102]. Часто исходные модели целесообразно заменять близкими «хорошими», например однородными (в которых некоторые элементы являются одинаковыми). Для ряда известных задач дискретной оптимизации (унификации, размещения, покрытия и др.) предложен алгоритм, основанный на идее аппроксимации «плохих» (названных нерегулярными) исходных данных «хорошими» (регулярными) [29]. Это позволяет применить достаточно эффективные алгорит-

мы, которые дают решение, близкое к решению, соответствующему реальному входу.

Для задач, направленных на исследование больших систем, часто целесообразно использовать имитационное моделирование [35]. Такое моделирование позволяет изучить все части системы, объединенные в единое целое. При этом можно осуществить управление всеми переменными, измерение параметров системы. Но имитационное моделирование является экспериментальным методом, и достоверность выходных данных зависит от степени адекватности модели. Следует особо отметить его роль при обосновании некоторых алгоритмов.

Одним из важнейших моментов построения общих схем алгоритмов является включение в контур решения человека [68, 84, 87]. Накоплен уже достаточно большой опыт создания и применения человеко-машинных процедур решения дискретных задач. Эти процедуры состоят из последовательности алгоритмов и включают участие человека на этапах перехода от одного алгоритма к другому.

ПРИЕМЫ ПОСТРОЕНИЯ ЛОКАЛЬНЫХ ФРАГМЕНТОВ АЛГОРИТМОВ

В качестве приемов построения локальных фрагментов можно рассматривать как перечисленные выше принципы построения общих схем, так и приемы типа локальной оптимизации (подъема) и др. Деление на принципы построения общих схем алгоритмов и локальных фрагментов в значительной степени условно. Такое разделение целесообразно со следующей точки зрения. Построение общей схемы решения задачи, как правило, основывается на декомпозиции (методе частных целей), причем, как указывалось, разбиение исходной задачи на подзадачи связано с искусством, интуицией разработчика алгоритма. При решении подзадач можно проводить подбор и модификацию известных алгоритмов. На этом уровне имеется возможность более обоснованно осуществлять построение алгоритмов решения (более простые задачи, меньше размерность). Таким образом, то, что на уровне построения общей схемы алгоритма из-за невозможности, сложности или высоких затрат на обоснование является интуицией, на уровне решения подзадач может иметь вид обоснования (аналитического или на основе вычислительных экспериментов).

Методы локальной оптимизации или градиентные (подъема, увеличения, спада) заключаются в пошаговом построении допустимого решения [58, 87]. При этом на каждом шаге определяется экстремум некоторой функции, зависящей от определенного числа компонент, составляющих допустимое решение (ищется способ увеличения, улучшения значения целевой функции на решении, если решается задача на максимум, с помощью локальных изменений) [94]. Когда возникает ситуация, при которой невозможно улучшить решение посредством локальных изменений, то алгоритм останавливается. Гарантировать оптимальность полученного решения нельзя. Используемый для решения многих задач теории расписаний перестановочный прием является, по сути, методом локальной оптимизации, приводящим к глобальному оптимуму [53, 92, 93, 95]. Методы подъема часто используют при решении различных разновидностей задачи о коммивояжере. Методы подъема являются достаточно грубыми. При их использовании запоминается некоторая цель и предпринимается все возможное для ее приближения, что делает эти методы несколько недальновидными [35].

В последнее время большое внимание уделяется greedy (жадным) алгоритмам. Такие алгоритмы в основном используются при необходимости максимизировать некоторую функцию, определенную на взвешенном графе [94]. Пусть, например, требуется найти во множестве со взвешенными элементами максимальное по весу подмножество, удовлетворяющее некоторым ограничениям. При применении жадного алгоритма следует упорядочить элементы по невозрастанию их весов и просматривать их по порядку, начиная с первого, строя подмножество элемент за элементом. При просмотре в состав подмножества добавляется элемент, если расширенное за счет этого элемента подмножество удовлетворяет ограничениям. В противном случае этот элемент пропускается. Такие методы обычно легко реализуются и работают быстро; их трудоемкость составляет порядка $O(n\log n)$. Жадным алгоритмом, например, решается задача построения минимального остового дерева. Общая постановка жадных алгоритмов и методов увеличения дается в теории матроидов.

Важен также метод рекурсии [94], в основе которого — сведение задачи к одной или более подзадачам. Подзадачи в свою очередь сводятся до тех пор, пока не станут в такой степени малыми, что можно будет их решать непосредственно. Затем в обратном порядке проводится объединение решений подзадач.

Сама по себе рекурсия не приводит к «хорошим» алгоритмам [5], но облегчает их понимание и в сочетании с другими методами (декомпозиция, балансировка и др.) позволяет получать алгоритмы одновременно и эффективные и элегантные. Метод динамического программирования, в сущности, можно рассматривать как специальный вид рекурсии [94].

При решении целочисленных задач линейного программирования могут использоваться методы отсечения. Они заключаются в сведении исходной задачи к последовательности обычных задач линейного программирования, каждая из которых получается из исходной путем «отсечения» некоторых нецелочисленных вершин [58].

ПРЕДСТАВЛЕНИЕ КОМБИНАТОРНЫХ ОБЪЕКТОВ, ОПЕРАЦИИ НАД НИМИ

Описание дискретных, комбинаторных объектов и рассмотрение типовых задач их обработки содержатся в работах [5, 35, 36, 62, 73]. В качестве математических объектов при построении ал-

горитмов используются элементы (целые числа, имена в виде набора символов) и множества (в основном конечные). Множества обычно подразделяют на неупорядоченные, упорядоченные и более сложные объекты (мультимножества, совокупности множеств, например графы). Важным объектом являются деревья, с помощью которых эффективно (по трудоемкости и памяти) могут проводиться некоторые операции, например поиск, исключение, добавление вершин. В качестве основных операций над множеством используются [79, 94]:

просмотр множества;

установление принадлежности элемента множеству;

исключение, добавление элемента (в том числе исключение после заданного, вставка на k-е место, удаление с k-го места и др.);

выбор элемента с k-м по величине значением;

определение объединения, пересечения, разности множеств.

При обработке графовых объектов используются бинарные и унарные операции [85]. Пусть имеется два графа $G_1=(V_1, E_1)$ и $G_2=(V_2, E_2)$, где V_1 и V_2 — множества вершин; E_1 и E_2 — множества дуг. В качестве бинарных операций рассматриваются:

объединение (U):

$$G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2);$$

пересечение ($\cap$):

$$G_1 \cap G_2 = (V_1 \cap V_2, E_1 \cap E_2);$$

кольцевая сумма ($+$):

$$G_1 + G_2.$$

Кольцевая сумма представляет собой граф, порожденный на множестве ребер E_1+E_2, который не имеет изолированных вершин и состоит только из ребер, присутствующих либо в графе G_1, либо в графе G_2, но не в обоих одновременно. Указанные три операции являются коммутативными.

В качестве унарных операций используются:

удаление или добавление вершины, ребра;

замыкание или отождествление (пара вершин в исходном графе G v_i и v_j замыкаются или отождествляются, если заменяются такой вершиной, при которой все ребра в графе G, инцидентные v_i и v_j, становятся инцидентными новой вершине);

стягивание (удаление ребра и отождествление его концевых вершин);

групповое замыкание или отождествление (аналогично замыканию или отождествлению, но новой вершиной заменяется не пара вершин, а некоторый подграф исходного графа);

расщепление (замена некоторой вершины в исходном графе двумя с ребром между ними; эта операция является обратной для операции замыкания);

групповое расщепление (операция, являющаяся обратной групповому замыканию или отождествлению).

Кроме того, используется большое число типовых операций на графах (например, поиск некоторых вершин) и большое число типовых задач на графах (построение транзитивного замыкания, покрытие остовным деревом; определение меры близости к структуре заданного типа и др.). При работе с деревьями (лесами) особое значение имеют операции просмотра вершин (прохождения леса), например [5, 43, 79, 94]:

в глубину (в прямом порядке);

снизу вверх;

в горизонтальном или симметричном порядке (для бинарных деревьев).

С точки зрения реализации как представления комбинаторных объектов, так и построения алгоритмов имеет значение необходимость проведения корректировки (исключение, замена, вставка элемента или фрагмента комбинаторного объекта). Без проведения корректировки объект является статическим, в противном случае — динамическим, и в этом случае требуются особые средства представления и обработки.

Основой представления комбинаторных объектов является представление последовательностей (конечных упорядоченных множеств) в виде списков (рис. 4):

последовательного распределения;

связанного распределения с включением циклического или дважды связанного списка.

Последовательное распределение представляет собой точный список элементов последовательности, расположенных в памяти по порядку (рис. 4, *а*). Оно легко осуществимо и требует небольших расходов памяти, обеспечивает легкий (прямой) поиск элементов, может быть использовано для представления многомерных объектов (многомерных массивов). К недостаткам относятся неудобство удаления или добавления новых элементов. Эти недостатки устраняются при использовании связанного распределения (рис. 4, *б*, *в*). В этом случае каждый элемент списка содержит и

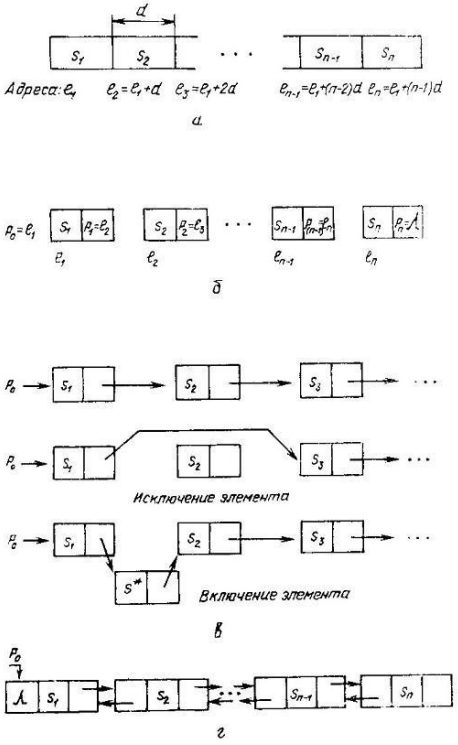


Рис. 4. Варианты представления последовательностей:

а — последовательное распределение; *б* — связанное распределение; *в* — исключение, включение элемента при связанном распределении; *г* — дважды связанный список

элемент последовательности, и указатель (адрес) следующего элемента. В циклическом списке в последнем элементе указывается адрес первого. В дважды связанном списке (рис. 4, *г*) в каждом элементе содержатся два указателя: на последующий и на предыдущий элементы. Это облегчает (повышает эффективность) проведение некоторых операций, например включение нового элемента перед некоторым заданным; исключение элемента без предварительного знания его предшественника и др.

Динамические последовательности (изменяются вследствие исключения некоторых элементов и включения новых) часто представляются в виде специальных списков: стеков и очередей. Стек представляет последовательность, все включения и исключения в которой проводятся на правом конце (вершине) стека (левый конец называется основанием) [79]. Таким образом, включение и исключение элементов производятся в соответствии с правилом «Первым пришел — последним ушел». Работа с очередью осуществляется по принципу «Первым пришел — первым ушел»: включение элементов осуществляется на правом конце списка (в конце очереди), а исключение — на левом (в начале). Иногда используются деки (двусторонние очереди), в которых и включение, и исключение элементов производятся на обоих концах списка. Стеки обычно реализуются и как последовательная реализация, и как связанное распределение. Для очереди последовательное распределение достаточно сложно (из-за перемещения границы начала очереди), что в основном обусловливает использование связанного распределения.

Иногда для представления множеств используются двоичные (характеристические) векторы, являющиеся последовательностью фрагментов (например, разрядов, байтов, ячеек) памяти (последовательное распределение), соответствующих каждому возможному элементу. (Каждый фрагмент указывает на содрежание элемента в множестве).

При выполнении операций над множествами (поиск элемента, проверка принадлежности, вызов, вставка, исключение и др.) трудоемкость имеет порядок $O(n)$, где n — число элементов. За счет использования специальных структур данных многие операции могут осуществляться с трудоемкостью $O(\log n)$. В основном специальные структуры используются для динамических последовательностей и имеют вид сбалансированных деревьев [5, 79, 94]:

бинарные, сбалансированные по высоте (*АВЛ*-деревья);

сбалансированные по весу;

сбалансированные сильно ветвящиеся деревья.

Такие структуры позволяют производить с логарифмической трудоемкостью [5, 35, 43, 79, 94] следующие операции: вставку, вызов, удаление элемента с (на) k-го места в списке; добавление элемента в очередь с приоритетом; вызов элемента с наименьшим значением; удаление элемента, положение которого известно; объединение очередей.

Для представления графов применяются [94]:

матрица смежностей или соединений;

матрица весов (для взвешенных графов);

список ребер в виде двух соответствующих друг другу массивов вершин (в основном для разреженных графов);

структура смежностей (массив, элементами которого являются списки последователей для каждой вершины).

При представлении деревьев используются матрицы смежностей, структуры смежностей, массив отцов (в качестве элемента для каждой вершины указывается предшествующая вершина), бинарные деревья.

ПРИМЕРЫ ОПТИМИЗАЦИОННЫХ КОМБИНАТОРНЫХ МОДЕЛЕЙ

ЗАДАЧИ ПРЕОБРАЗОВАНИЯ СТРУКТУР

В основу процесса решения значительной части задач принятия решений можно положить преобразование графовых структур различного типа [10, 16, 56, 61, 66]. Пусть на входе имеется полученный на основе предпочтений ЛПР, многокритериальных оценок альтернатив орграф предпочтений G. Требуется получить некоторый результирующий, финитный орграф, соответствующий постановке задачи принятия решений [10, 56]. Можно выделить четыре уровня базовых моделей (структур):

исходные данные, например получаемые от ЛПР и/или экспертов (уровень 0);

первичные аппроксимирующие структуры (уровень 1);

промежуточные структуры (уровень 2);

результирующие, целевые, финитные структуры (уровень 3).

Примерами структур нулевого уровня являются орграфы произвольного вида (наличие в них контуров соответствует противоречиям в исходных данных); структур первого уровня — бесконтурные графы (частичные порядки); второго уровня — слоистые графы; третьего уровня — цепочки, двухуровневые графы. Можно ввести следующие базовые типы структур.

Произвольный орграф без контуров (частичный порядок) R, в том числе:

граф линейного порядка (цепочка) C;

граф древесного порядка T;

упорядоченная структура в виде параллельно-последовательного графа P;

звездный граф Z.

Слоистая структура S, в том числе:

двухдольный бесконтурный граф K;

структура S', верхние слои которой содержат по одному элементу.

В слоистых структурах множество вершин V разбито на непересекающиеся слои V_i, $i = \overline{1,m} : V = \bigcup_{i=1}^{m} V_i$, для каждого

жащих слоев доминирует над любой вершиной из нижележащих слоев (доминирование может пониматься как в поэлементном, так и в групповом смысле [5, 10]).

Могут быть введены дополнительные условия типа эквивалентности вершин внутри слоев (линейный квазипорядок) или их несравнимости. Кроме того, интерес представляют структуры, являющиеся объединениями или композицией базовых структур, например NT, NS, и др.

Для каждой типовой задачи принятия решений можно рассмотреть отнесение базовых структур предпочтений к уровням базовых моделей. Тем самым строятся типовые многошаговые схемы решения задач принятия решений на базе локальных задач, сформулированных в виде преобразования (аппроксимации) графов предпочтений. При этом используются следующие средства:

удаление, добавление или переориентация дуг;

удаление вершин с дугами;

замена некоторого подграфа одной вершиной (стягивание, конденсация, склеивание вершин);

удаление вершин с заменой исключаемых дуг, некоторого подграфа и с возможной заменой исключаемых дуг;

добавление вершины с дугами или некоторого подграфа.

Преобразование графа формализуется в виде задачи поиска наилучшей аппроксимирующей структуры из заданного класса.

Наилучшей структурой может считаться:

ближайшая структура данного типа;

структура данного типа, у которой наиболее ярко проявляется определенное свойство или комплекс свойств.

В первом случае на множестве структур вводится метрика (или хотя бы псевдометрика) ρ и ставится задача

$$\rho(x, y) \to \min_{y \in Y},$$

где x — исходная структура;

y — аппроксимирующая структура;

Y — множество структур заданного типа.

Во втором случае на множестве Y вводится функционал $\Phi(y)$, оценивающий желательность свойства аппроксимирующих структур или их сходство с x, и рассматривается задача

$$\Phi(y) \to \max_{y \in Y}.$$

Схема решения типовых задач принятия решений можно рассмотреть на примерах. Для линейного упорядочения альтернатив возможны следующие пути, использующие построение:

непосредственно линейного упорядочения (преобразование типа $G \to C$);

предварительного частичного порядка, т. е. устранение противоречий в исходной структуре $G \to R$, $G \to T$, $G \to P$, $G \to P \to T$ или $G \to R \to P$ с последующим преобразованием его в линейный порядок $R \to C$, $T \to C$ или $P \to C$;

линейного порядка по схеме последовательного выделения, при котором в «хвост» формируемой цепочки последовательно добавляются альтернативы $G \to S' \to \ldots \to S' \to C$;

предварительной слоистой структуры с дальнейшим упорядочением элементов внутри слоев $G \to S \to C$.

При решении задачи выделения подмножества лучших альтернатив (формирования портфеля заказов) возможно использование следующих стратегий:

непосредственное выделение подмножества лучших элементов $G \to K$;

решение задачи по схеме последовательного выделения $G \to S' \to \ldots \to S' \to K$;

построение предварительного частичного порядка $G \to R$, $G \to T$, $G \to P$, $G \to P \to T$ или $G \to R \to P$ с последующим его преобразованием к заданной структуре $R \to K$, $T \to K$, $P \to K$;

построение заданной структуры на основе предварительного линейного порядка $G \to C \to K$;

построение заданной структуры на основе слоистой $G \to S \to K$.

Анализ стратегий для задач принятия решений позволяет выявить основные типы комбинаторных моделей преобразования структур. В связи с этим актуально создание библиотек алгоритмов, программ, интерактивных процедур для их решения.

ЗАДАЧИ ПОСТРОЕНИЯ СЛОИСТЫХ СТРУКТУР

При исследовании, построении алгоритмов решения практически всех комбинаторных моделей может возникнуть необходимость разбиения исходного набора объектов на упорядоченные по важности группы (слои), групповой оценки объектов в некоторой порядковой шкале. Задача групповой оценки может соответствовать содержательной постановке или являться важным вспомогательным фрагментом решения исходной комбинаторной задачи. Это одна из типовых задач принятия решений [10, 104]. В качестве приложений такой модели можно привести подведение итогов соцсоревнования, оценку качества товаров, отбор статей для опубликования, планирование, проектирование [2, 28, 33, 40, 67].

Специфика задачи построения слоистых структур заключается в том, что в большинстве практических случаев необходимо проектировать конкретную методику решения задачи на основе типовых подходов. Общая схема решения имеет следующий вид.

Этап 1. Анализ ситуации, выявление задачи оценивания (группового упорядочения).

Этап 2. Постановка (структуризация) задачи:

анализ системы целей, способов их достижения;

формирование системы критериев;

формирование множества оцениваемых показателей (декомпозиция критериев).

Этап 3. Оценивание объектов:

оценивание значений показателей на основе измерения, расчетов, экспертиз;

агрегирование данных (формирование оценок по критериям).

Этап 4. Решение задачи (получение информации от ЛПР, анализ и обработка данных, формирование и анализ решения).

Приведенная схема с учетом специфики задачи, процесса решения может включать обратные связи.

Стратегии решения задачи группового оценивания (группового доминирования, ранжирования) представлены на рис. 5. При этом использованы следующие модели данных:

исходное множество объектов (альтернатив) $B=\{1, \ldots, n\}$;

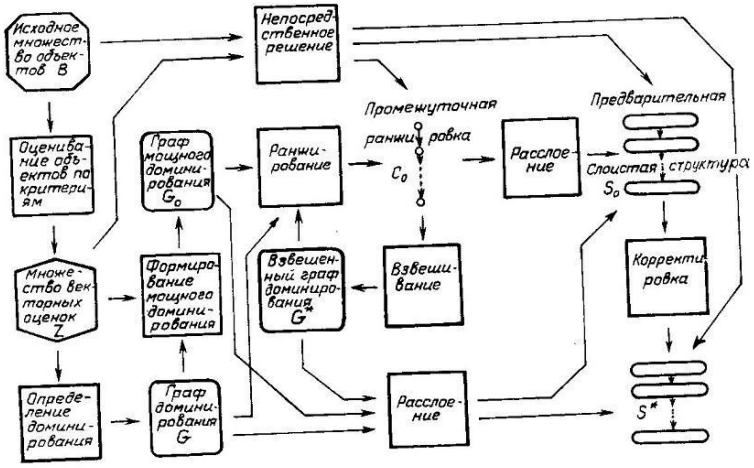


Рис. 5. Стратегии решения

множество векторных оценок объектов Z;

граф доминирования $G=(B, \rightarrow)$, где $\rightarrow$ — множество дуг, направленных к менее предпочтительным элементам;

граф доминирования с большим числом дуг G_0 (насыщенный граф);

взвешенный граф доминирования G^* (каждой вершине приписан неотрицательный вес, характеризующий «важность», каждой дуге — неотрицательный вес, характеризующий степень доминирования);

промежуточное метризированное ранжирование (упорядочение) C_0;

предварительная слоистая структура S_0;

результирующая слоистая структура S^*.

Прямое преобразование $B \rightarrow S^*(S_0)$ может быть реализовано интерактивной процедурой, но это требует использования сложных для ЛПР операций. Это верно и для $B \rightarrow G$. Процесс оценивания объектов по критериям $(B \rightarrow Z)$ осуществляется с помощью измерений, расчетов, экспертного оценивания. Эта задача индивидуального оценивания, требующая, в частности, анализа системы критериев, показателей, организации процесса оценивания.

Переход $Z \rightarrow G$ является несложным формальным преобразованием на основе покомпонентного сравнения векторных оценок с трудоемкостью $O(n^2)$. Преобразование $Z \rightarrow G_0$ может проводиться на основе информации от ЛПР или использования различных механизмов выбора, обобщений мнений экспертов. В частности, такое преобразование может осуществляться парными сравнениями, проводимыми ЛПР, с использованием методов компенсации, методом ЭЛЕКТРА и его модификациями [10, 22, 40, 81].

На основе получаемого графа G часто трудно проводить решение задачи оценивания, так как в нем может содержаться мало информации о предпочтениях. За счет получения дополнительной информации от ЛПР, использования формальных методов можно осуществить переход $G \rightarrow G_0$ (например, итеративное сближение порогов p и q в методе ЭЛЕКТРА). Оценка достаточности мощности множества дуг может проводиться содержательно или на основе формальных критериев.

Стратегия построения S_0 может осуществляться различными способами. Общий вид цепочки преобразований имеет вид: $G_0 \rightarrow C_0 \rightarrow S_0$ или с учетом вспомогательной структуры $G^*: G_0 \rightarrow C_0 \rightarrow G^* \rightarrow \ldots \rightarrow C_0 \rightarrow S_0$. Непосредственно построение G_0, $G^* \rightarrow S_0$ может проводиться, например, последовательным выделением групп наилучших элементов (в частности, по Парето), последовательным распределением объектов по упорядоченным классам на основе информации от ЛПР, на основе модели блочной триангуляции [9, 10, 40].

Для построения промежуточного ранжирования G_0, $G^* \rightarrow C_0$ применяются различные модели: «спортивная» (построчные суммы); о лидере или ее модификации; стохастическая; триангуляционная, а также по ближним или дальним связям [9, 10]. Построение $Z \rightarrow C_0$ может реализовываться на основе сверток оценок по критериям (прямые методы принятия решений); методов, основанных на зависимости критериев (лексикография и др.); компенсации; непосредственной классификации с помощью ЛПР, а также аксиоматических с оценкой полезности [10, 22, 40, 42].

Построение слоистой структуры на основе метризированного ранжирования $C_0 \rightarrow S_0$ может проводиться методами, аналогичными используемым при преобразовании $G^* \rightarrow S_0$, но они будут в значительной степени вырожденными, так как C_0, по сути, является вырожденным частным случаем G^*. В практических задачах может оказаться рациональным разбиение элементов C_0 на слои при помощи ЛПР (на содержательном уровне, в рамках интерактивных процедур), использование эвристических схем решения.

В рамках цепочки преобразований $Z \rightarrow G_0 \rightarrow G^* \rightarrow C_0 \rightarrow S_0 \rightarrow S^*$ может быть организован параллельно-последовательный процесс преобразования информации (многошаговые схемы [105]). Парал-

одинаковых или различных стратегий, методов обработки. Распараллеливание процесса может проводиться на любом этапе на базе использования данных от различных экспертов; различных способов преобразования, например различных порогов в методе ЭЛЕКТРА; на этапе формирования G_0 — на основе учета различных групп критериев. Построение результирующих, агрегированных моделей может проводиться также на всех этапах. Агрегация, как правило, основывается на определении некоторых усредненных моделей данных типа медианы Кемени, выделении некоторого общего ядра, композиции по допустимости и др. [61, 96, 105].

С практической точки зрения наиболее важна агрегация на заключительных этапах цепочки преобразований, т. е. агрегация ранжировок C_0 и предварительных слоистых структур S_0. Актуальными являются исследование, применение агрегации структур с весами, хотя такая операция обычно не рассматривается. Очевидно, во многих случаях целесообразно формировать S_0 на основе использования автоматических процедур, а при переходе к S^* основную роль отводить ЛПР в рамках интерактивной процедуры. Следует отметить, что S^* часто связано с фиксированной порядковой шкалой.

Заслуживает внимания анализ ряда условий, характеристик качества исходной информаци и процесса обработки, например [61, 96]:

устойчивость (результата к малым изменениям исходных данных);

согласованность (при агрегации) и др.

Кроме того, использование в качестве базы для решения задачи группового упорядочения бинарного отношения предпочтения-безразличия позволяет в большинстве случаев обойти вопрос о принципиальном отсутствии оценок по некоторым критериям для некоторых объектов. При этом не требуется определение отсутствующих оценок.

Для задачи группировки объектов (оценки в номинальной шкале) на основе характеристик объектов типа близости, различия, принадлежности и других можно строить аналогичные стратегии.

ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ

Задачи теории расписаний имеют большое значение для практики, в частности для задач планирования в производственных и вычислительных системах. При этом обычно требуется составить расписание обработки множества изделий (работ, заявок, заданий) с помощью имеющихся станков, процессоров, исполнителей.

На содержательном уровне можно выделить следующие основные параметры классификации моделей [47, 55, 92, 93, 95]: число ($1, 2 \geqslant 3$) и тип процессоров (идентичные, однородные, различные);

тип системы (конвейерная, циклическая, произвольная); наличие прерываний, графовых ограничений (например, технологических ограничений предшествования, предпочтений ЛПР), а также директивных сроков начала, окончания обработки изделий; тип критерия (максимальная длина расписания и др.), изделий (простые, составные), измерения характеристик (в количественных, порядковых шкалах); наличие ресурсных ограничений и др.

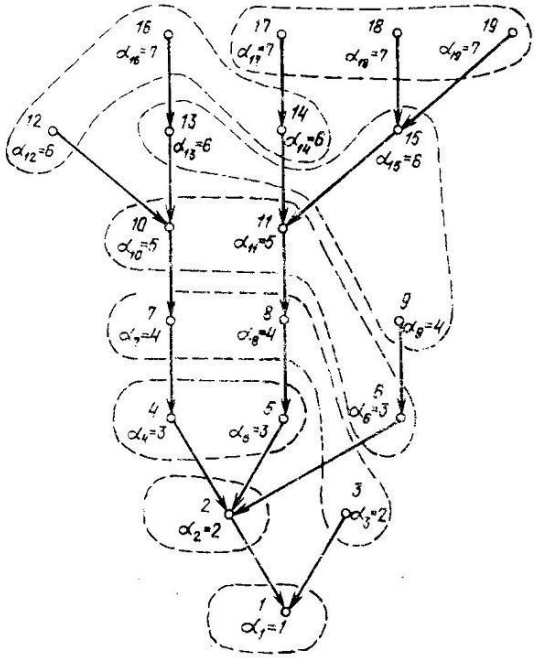
Спектр моделей теории расписаний чрезвычайно велик (от простейших до самых сложных) [47, 55, 92, 93, 95]. Многие модели теории расписаний совпадают или сводятся к другим (почти всем) комбинаторным моделям, например к задачам упаковки, коммивояжера, рюкзачного типа.

Для решения задач теории расписаний используются практически все приемы, в частности перестановочный (попарная перестановка элементов в расписании с целью улучшения значения критериев — локальная оптимизация [93, 95]). В качестве примеров рассмотрим задачу о линии сборки и класс задач теории расписаний, для которых построены эффективные точные алгоритмы [53, 92].

При планировании работы линии сборки иногда возникает задача теории расписаний следующего вида [93]. Имеется множество работ (например, сверление отверстий, нарезка резьбы и др.), причем выполнение каждой из них требует одинакового количества времени. Технологические ограничения на порядок выполнения работ определяют ограничения предшествования (например, сначала сверление, а затем нарезка резьбы). На линиях сборки такое ограничение предшествования имеет вид ориентированного дерева с направлением дуг к корню. Каждую работу может выполнить любой из m идентичных исполнителей. Такая задача может возникать и в других практических ситуациях, например при организации вычислений на ЭВМ, при планировании выполнения сложных проектов. В качестве исполнителей в реальных задачах могут выступать процессоры, машины, люди, организации. Задача заключается в составлении расписания, при котором все работы завершаются за минимальное время.

Алгоритм решения задачи имеет следующий вид. Каждой работе $\in R$ присваивается метка α те $i_i = l_i + 1$, где l_i — длина пути из i в корневую вершину. Если общее число начальных висячих вершин не больше m, то всем работам, соответствующим начальным вершинам, одновременно назначаются исполнители. Если же общее число начальных вершин больше m, то выбирается m начальных вершин таким образом, чтобы значения a_i у выбранных вершин были бы не меньше, чем значения α у невыбранных. В случае совпадения значений a_i выбор осуществляется произвольно. Далее аналогично изложенному обрабатывается оставшийся граф. На рис. 6 представлен пример составления расписания. Приведенный алгоритм иногда называют «алгоритмом длиннейших хвостов», так как физически можно проинтерпретировать его работу следующим образом. Если подвесить дерево за корневую

вершину, то назначать исполнителей надо тем работам, которые соответствуют вершинам, расположенным ниже других. Можно получить оптимальное решение и в случае, когда назначение исполнителей проводится в обратном порядке — от корня к висячим вершинам (по близости к корню).



17	12	13	10	7			
18	16	15	11	8	4		
19	14	9	6	3	5	2	1

Рис. 6. Пример составления расписания для линии сборки

Рассмотрим следующий класс задач теории расписаний. Имеются множество заданий (работ) $R=\{1, 2, ..., n\}$ и ограничение предшествования, заданное направленным ациклическим графом $G=(R, \Gamma)$. Требуется установить оптимальный порядок s^* выполнения работ (расписание), который является допустимым (т. е.

удовлетворяет заданным ограничениям предшествования) и на котором значение функционала минимально:

$$f(s^*) = \min_s f(s) .$$

Задача 1. Имеется одна машина. Каждая работа $i \in R$ характеризуется временем выполнения τ_i и функцией штрафа $\varphi_i(t) = a_i t + b$. Требуется минимизировать суммарный штраф

$$f(s) = \sum_{i=1}^{n} \varphi_i(C_i) ,$$

где C_i — момент окончания выполнения работы i (все работы поступают в момент времени $t=0$).

Задача 2. Отличается от предыдущей тем, что функция штрафа имеет вид: $\varphi_i(t) = a_i \exp(\lambda t) + b_i$.

Задача 3. Это задача Беллмана-Джонсона для случая двух машин. Каждая работа $i \in R$ должна выполняться сначала на первой машине, затем на второй с временами выполнения a_i и b_i соответственно. Требуется минимизировать суммарное время выполнения всех работ.

Задача 4. Имеется одна машина. Каждая работа $i \in R$ характеризуется временем выполнения τ_i и функцией штрафа $\varphi_i(t) = a_i t + b_i$. Требуется минимизировать суммарный штраф

$f(s) = \sum_{i=1}^{n} \varphi_i(t_i)$, где $t_i = \prod_{l=1}^{i} \tau_{s[l]}$, $s[l]$ — номер работы, находящейся в расписании s на l-м месте.

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

Задача 6. Имеется одна машина. Каждая работа $i \in R$ характеризуется стоимостью выполнения ρ_i и вероятностью срыва выполнения работы p_i. Требуется минимизировать среднюю стоимость выполнения всех работ

$$f(s) = \sum_{l=1}^{n} \frac{\rho_{s[l]}}{D_{s[l]}}, \quad \text{где} \quad D_{s[l]} = \prod_{k=1}^{s[l]} (1 - p_k) .$$

Для задачи 3 просто доказывается, что оптимальное расписание можно искать на множестве одномаршрутных расписаний. Таким образом, расписание s — это перестановка n чисел 1, 2,..., n:

$$s = \langle s[1], s[2], ..., s[n] \rangle .$$

Указанные задачи, несмотря на различие в постановках, обладают общими свойствами. Таким образом, можно рассматривать некоторую обобщенную задачу, которая характеризуется следующим. В случае отсутствия ограничений предшествования она может быть решена упорядочением работ по неубыванию одноместных функций — приоритетов $\omega(i)$, вычисляемых для каждой

работы $i \in R$ (на основе перестановочного приема [93]). Приоритеты для рассматриваемых задач [53] имеют вид:

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

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

$$\omega(i) = sign(a_i - b_i)[M - \min(a_i, b_i)],$$

где

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

$$\omega(i) = \frac{a_i \tau_i}{1 - \tau_i};$$

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

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

Дополнительно для задач 1—6 выполняется условие склеивания. В этом случае некоторые наборы (в частности, пары) работ, которые могут выполняться непосредственно одна за одной (и это не ухудшает значение критерия), могут быть заменены одной эквивалентной. В работе [92] это свойство определено как свойство приоритетно-порождаемости у критериев качества. Указанные два свойства позволяют эффективно решать задачи 1—6 для случая, когда G — дерево, параллельно-последовательный граф.

Обычно в моделях теории расписаний используются критерии двух типов:

$$f_1 = \max_{1 \le i \le n} \{\varphi_i(C_i)\} \text{ и } f_2 = \sum_{i=1}^{n} \varphi_i(C_i).$$

В первом случае минимизируется максимальное значение функции штрафа, во втором — суммарное или среднее. В табл. 1 и 2 представлены характеристики сложности ряда однопроцессорных задач теории расписаний [55]. При этом рассмотрены следующие критерии:

$$C_{\max} = \max_{1 \le i \le n} C_i;\ \Sigma C_i = \sum_{i=1}^{n} C_i;$$

$$\Sigma a_i C_i = \sum_{i=1}^{n} a_i C_i;\ \Sigma a_i e^{\lambda C_i} = \sum_{i=1}^{n} a_i e^{\lambda C_i};$$

$$\Sigma U_i = \sum_{i=1}^{n} U_i, \text{ где } U_i = \max(0, C_i - d_i),$$

d_i — директивный срок окончания выполнения задания;

Таблица 1

Число процессоров	Время выполнения	Вид орграфа ограничений: Несвязный (точечный) B								Древовидный T							
		ΣC_i	$C_{\max}$	$\Sigma a_i e^{\lambda C_i}$	$\Sigma a_i C_i$	ΣU_i	$\Sigma a_i U_i$	ΣT_i	$\Sigma a_i T_i$	ΣC_i	$C_{\max}$	$\Sigma a_i e^{\lambda C_i}$	$\Sigma a_i C_i$	ΣU_i	$\Sigma a_i U_i$	ΣT_i	$\Sigma a_i T_i$
1	1	*	*	*	*	*	$O(n \log n)$	*	$O(n^2)$	*	*	*	*	$O(n \log n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
	1,2	*	*	*	*	$O(n^2)$	$O(n^2)$	$O(n \log n)$	$O(n^2)$	*	*	*	*	$O(n \log n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
	$(0, \lambda)$	*	*	*	*	$O(n^2)$	$O(n^2)$	$O(n \log n)$	$O(n^2)$	*	*	*	*	$O(n \log n)$	$O(n^2)$	$O(n^2)$	$O(n^2)$

Таблица 2

Число процессоров	Время выполнения	Вид графа ограничений: Параллельно-последовательный P										Бесконтурный G									
		ΣC_i	$C_{\max}$	$\Sigma a_i e^{\lambda C_i}$	$O(n\log n)$ *	$\Sigma a_i C_i$	ΣU_i	$\Sigma a_i U_i$	ΣT_i	$\Sigma a_i T_i$	ΣC_i	$C_{\max}$	$\Sigma a_i e^{\lambda C_i}$	$\Sigma a_i D_i$	ΣU_i	$\Sigma a_i U_i$	ΣT_i	$\Sigma a_i T_i$
1	$(0, \infty)$	*	*	*	*	*	?	?	—	—	—	*	?	—	—	—	—	—
	1,2	*	*	*	*	*	?	?	—	—	—	*	?	—	—	—	—	—
	1	*	*	*	*	*	?	?	—	—	—	*	?	—	—	—	—	—

$$\Sigma a_i U_i = \sum_{i=1}^{n} a_i U_i ; \quad \Sigma T_i = \sum_{i=1}^{n} T_i ,$$

где $T_i = 0$ при $C_i \leqslant d_i$ и 1 — в противном случае;

$$\Sigma a_i T_i = \sum_{i=1}^{n} a_i T_i .$$

В таблицах приведены оценки сложности задач: «*» — существует эффективный точный алгоритм; «!» — задача является NP-трудной; «?» — вопрос о сложности является открытым; «а» — существует эффективный приближенный алгоритм с гарантированной точностью.

Некоторые многопроцессорные модели теории расписаний рассмотрены в разделе о задаче упаковки.

ЗАДАЧИ РЮКЗАЧНОГО ТИПА

Задачи рюкзачного типа имеют наиболее широкое применение. Простейшие рюкзачные задачи достаточно хорошо исследованы. Они являются одними из самых «легких» в классе NP-трудных задач, поскольку для них построены ε-приближенные алгоритмы с гарантированной относительной погрешностью. При разработке алгоритмов решения задач рюкзачного типа используются методы динамического программирования, ветвей и границ, эвристические (greedy-процедуры, релаксация по Лагранжу и др.).

На содержательном уровне можно выделить ряд параметров классификации задач рюкзачного типа: тип критерия (максимизация — минимизация, учет фиксированных доплат, многокритериальный случай); число ресурсов (1, 2, $\leqslant 3$); тип задачи (обычная, треугольная, лестничная, блочная и др.); наличие графовых ограничений (технологических, предпочтений ЛПР); число рюкзаков (1, 2, $\geqslant 3$); тип измерения характеристик (в количественных, порядковых шкалах).

Некоторые модели задач рюкзачного типа совпадают, например, с моделями упаковки, теории расписаний.

Простейшие модели рюкзачного типа имеют различные приложения, и их целесообразно использовать в качестве базовых фрагментов, подзадач при анализе, решении сложных задач рюкзачного типа. Простейший вариант задачи о рюкзаке имеет вид [25, 36, 83]:

максимизировать

$$\sum_{i=1}^{n} c_i x_i$$

при условиях:

$$\sum_{i=1}^{n} a_i x_i \leqslant b ; \; x_i = 0 \vee 1 ; \; c_i \geqslant 0 ; \; a_i \geqslant 0 ; \; b \geqslant 0 ; \; i = \overline{1, n} . \qquad (1)$$

В практике возникают постановки, в которых ресурсные ограничения имеют более сложный вид. Например, в треугольной задаче

максимизировать

$$\sum_{i=1}^{n} c_i x_i$$

при условиях:

$$\sum_{i=1}^{j} a_i x_i \leqslant b_j ; \; j = \overline{1, n} ;$$

$$x_i = 0 \vee 1 ; \; c_i \geqslant 0 ; \; a_i \geqslant 0 ; \; b_j \geqslant 0 ; \; i = \overline{1, n} .$$

При решении экономических задач, связанных с капиталовложениями, возникает модель с фиксированными доплатами:

максимизировать

$$\sum_{i=1}^{n} (c_i x_i + p_i \operatorname{sign} x_i)$$

при условиях:

$$\sum_{i=1}^{n} (a_i x_i + q_i \operatorname{sign} x_i) \leqslant b ;$$

x_i — целые ; $x_i \geqslant 0$; $c_i \geqslant 0$; $a_i \geqslant 0$; $b \geqslant 0$; $p_i \geqslant 0$; $q_i \geqslant 0$; $i = \overline{1, n}$.

Во многих практических ситуациях требуется не только отобрать объекты, но и выбрать для каждого отобранного объекта некоторый вариант реализации, характеризующийся индивидуальными эффективностью и ресурсными ограничениями. Это приводит к задаче лестничного рюкзака (многовариантной задаче):

максимизировать

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} c_{ij} x_{ij}$$

при условиях:

$$\sum_{i=1}^{n} \sum_{j=1}^{l_i} a_{ij} x_{ij} \leqslant b ;$$

$$\sum_{j=1}^{l_i} x_{ij} \leqslant 1 ; \; i = \overline{1, n} ;$$

$$x_{ij} = 0 \vee 1 ; \; c_{ij} \geqslant 0 , \; a_{ij} \geqslant 0 ; \; b \geqslant 0 ; \; i = \overline{1, n} ; \; j = \overline{1, l_i} .$$

или к задаче блочного рюкзака, в которой второе ограничение имеет вид:

$$\sum_{j=1}^{l_i} x_{ij} = 1 , \; i = \overline{1, n} .$$

В блочной задаче для каждого объекта i проводится выбор одного из возможных вариантов реализации, а в лестничной — одним из вариантов является ситуация, когда объект не отбирается.

Приведенные выше постановки задач несложным образом преобразуются к виду, когда критерий минимизируется [25]. При этом ресурсное ограничение изменяет знак. Такие модели возникают, например, при минимизации затрат.

Характеристики сложности простейших комбинаторных задач рюкзачного типа приведены в табл. 3 [25].

Таблица 3

Тип задачи	Значения коэффициентов целевой функции			
	Максимизация		Минимизация	
	$c_i = 1$	$c_i \in (0, \infty)$	$c_i = 1$	$c_i \in (0, \infty)$
Обычная	* $O(n \log n)$	! a $O(n^2/\varepsilon)$		! a $O(n^2/\varepsilon)$
Треугольная	*	! a $O(n^2/\varepsilon)$		! a $O(n^2/\varepsilon)$
Лестничная	* $O(n \log n)$	! a $O(mn/\varepsilon - n \log n)$		! a $O(mn/\varepsilon + mn \log n + n \log n)$
Блочная	* $O(n \log n)$	! a $O(mn/\varepsilon - n \log n)$		! a $O(mn/\varepsilon + mn \log n + n \log n)$
С фиксированными доплатами		! a $O(n^2/\varepsilon)$		! a $O(n/\varepsilon^2 \log 1/\varepsilon + n \log n)$

Известным частным случаем задачи о рюкзаке является постановка, в которой требуется разбить R на две группы, у которых наиболее близки суммы параметров a_i (задача о камнях). Такая задача является NP-трудной, для нее используются алгоритмы, разработанные для (1).

Важным приложением моделей рюкзачного типа являются задачи организации данных в иерархической памяти вычислительных систем. Подобные задачи часто направлены на экономное использование памяти высокого быстродействия и уменьшение числа обменов, пересылки данных. Примером такой задачи является комбинаторная задача объединения модулей некоторого программного комплекса при оверлейной структуре [54]. Предполагается, что передача по управлению осуществляется с возвратом в вызывающий модуль и связи между модулями имеют вид дерева. При этом учитывается ограничение на общий объем требуемой опера-

тивной памяти. Это приводит к графовому ограничению, которое представляется в виде нелинейной формы.

Рассмотрим формальную постановку. Задано ориентированное дерево $G=(R, \Gamma)$ с направлением дуг от корня $a_0 \in R$ к висячим вершинам, где Γ — многозначное отображение R в R. Каждая вершина (модуль) $a \in R$ имеет вес (требуемый объем оперативной памяти) $v(a) \geqslant 0$, каждая дуга (a, b) ($a, b \in R$ и $b \in \Gamma a$) имеет вес (частота вызова модуля b) $w(a, b)$. Вводятся: для каждого пути

$$L(a_1, a_l) = (a_1, \ldots, a_i, \ldots a_l) \quad (a_j \in \Gamma a_{j-1},\ j = \overline{1, l-1})$$

вес $v(L(a_1, a_l)) = \sum_{i=1}^{l} v(a_i)$, причем для графа G вес

$$v(G) = \max_{b \in B} v(L(a_0, b)),$$

где $B=\{a \in R \mid \Gamma a=\varnothing\}$.

Пусть $G_a=(R_a, \Gamma)$ — поддерево с корнем в $a \in R$ (в R_a кроме a входят все вершины из R, достижимые из a). «Хвост» вершины a определяется как величина

$$\hat{v}(a) = v(G_a) - v(a), \text{ причем } \hat{v}(a) = \max_{b \in \Gamma a} v(G_b).$$

Пусть $\forall a \in R$ a_0 $x(a)$ — булева переменная, равная 1, если дуга, направленная в a, стягивается и вершина a объединяется с непосредственно доминирующей ее, и 0 — в противном случае. Преобразование графа при стягивании дуги (a, b) определяется так: вершина a заменяется на $J(a, b)$ с весом $v(J(a, b)) = v(a)+v(b)$ и $\Gamma J(a, b) = (\Gamma a \cup \Gamma b) \setminus b$, вершина b с идущими из нее дугами исключается.

Пусть вектор x имеет компоненты $x(a)$ $\forall a \in R \setminus a_0$. С учетом стягивания дуг $\forall a \in R$ вес вершины и вес «хвоста» можно рассматривать как функции x: $v(a, x)$, $\hat{v}(a, x)$. Таким образом, модель имеет вид:

максимизировать

$$f(x) = \sum_{a \in R \setminus a_0} x(a) w(a)$$

при условии:

$$v(a_0, x) + \hat{v}(a_0, x) \leqslant \varphi, \qquad (2)$$

где $\varphi \geqslant 0$ (ограничение на объем оперативной памяти).

Другим (вспомогательным) вариантом задачи (2) является случай, когда условие заменяется двумя:

$$v(a_0, x) \leqslant \varphi^{=},\ \hat{v}(a_0, x) \leqslant \varphi^{+},\ \varphi^{-},\ \varphi^{+} \geqslant 0.$$

В случае, когда G — трехуровневое дерево, задача (2) имеет вид (рис. 7):

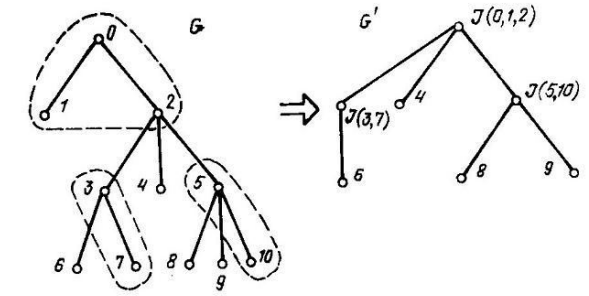
максимизировать

$$f(x) = \sum_{a \in R \setminus a_0} x(a) w(a)$$

при условии:

$$v(a_0) + \sum_{a \in \Gamma a_0} x(a)[v(a) + \sum_{b \in \Gamma a} x(b) v(b) + \max_{b \in \Gamma a}((1 - x(b)) v(b))] +$$

$$+ \max_{a \in \Gamma a_0}\{(1 - x(a))[v(a) + \sum_{b \in \Gamma a} x(b) v(b) + \max_{b \in \Gamma a}((1 - x(b)) v(b))]\} \leqslant \varphi.$$



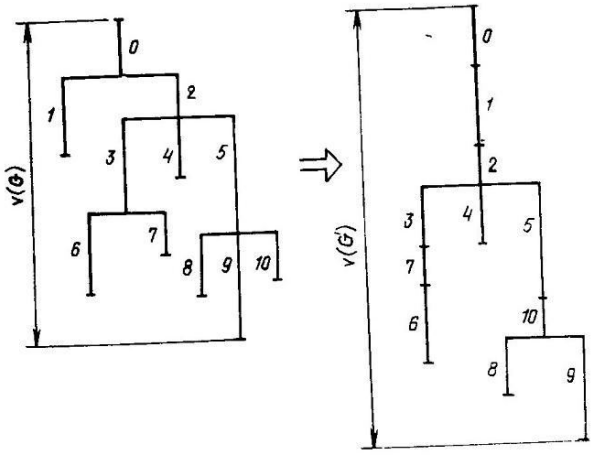


Рис. 7. Пример компоновки модулей оверлейной структуры

Задача (2), за исключением некоторых частных случаев, является NP-трудной, так как обобщает задачу (1). В табл. 4 приведены характеристики известных эффективных алгоритмов для решения различных вариантов задачи (2) [54]. При этом использованы обозначения; μ — число уровней G; η — число висячих вершин в G; a^2 — существует эффективный (ε, δ)-приближенный алгоритм с гарантированной погрешностью по целевой функции и ресурсному ограничению.

Таблица 4

Число уровней дерева μ	Значения весов объектов v	Значения весов дуг w: 1	Значения весов дуг w: $(0, \infty)$
2	1	* $O(n \log n)$	* $O(n \log n)$
	$(0, \infty)$	* $O(n \log n)$	! a $O(n^2/\varepsilon)$
3	1		! a $O(n^3/\varepsilon)$
	$(0, \infty)$		! a^2 $O\left(\frac{n^2 \eta^4}{\varepsilon \delta^3}\right)$
$\geqslant 4$	1		!
	$(0, \infty)$		! a^2 $O\left(\frac{n^2 \eta^5}{\varepsilon \delta^4}\right)$

При решении практических задач G часто имеет вид более сложный, чем дерево. В этом случае рациональным является подход на основе метода частных целей, включая аппроксимацию G более удобными структурами (деревом, несколькими графами). При анализе реальных программных комплексов часто $\mu = 3$.

Во многих практических задачах на исходный набор элементов накладываются некоторые графовые ограничения. Это возникает в приложениях, когда объекты связаны между собой по технологии, например: одни научные темы, строительные заказы могут выполняться совместно или в определенной последовательности. С другой стороны, графовые ограничения возникают при наличии предпочтений ЛПР, многокритериальных оценок объектов в виде бинарного отношения доминирования [10, 56, 61]. Такое графовое ограничение может нарушаться, в отличие от технологических ограничений, нарушение которых недопустимо.

Рассмотрим достаточно общую постановку задачи о рюкзаке с практически типовыми графовыми ограничениями. Графовое ограничение может быть представлено в виде орграфа $G = (R, \rightarrow)$, где $\rightarrow$ — множество дуг, например направленных от более предпочтительных к менее предпочтительным элементам. Постановка задачи имеет вид:

максимизировать

$$\sum_{i=1}^{n} c_i x_i$$

при условиях:

$$\sum_{i=1}^{n} a_{ie} \leqslant b_e, \; l = \overline{1, k} \; ;$$

$$x_i \geqslant x_j, \text{ если } i \rightarrow j \; \forall i, j \in R \; ;$$

$$x_i = 0 \vee 1, \; c_i \geqslant 0, \; a_{ie} \geqslant 0, \; b_l \geqslant 0, \; i = \overline{1, n}, l = \overline{1, k} \; . \qquad (3)$$

Подход к решению задачи может основываться на применении для фрагментов задачи ε-приближенного алгоритма для решения задачи о рюкзаке типа (1) с гарантированной погрешностью. Если $c_i = 1 \; \forall i$, то в качестве базового вместо приближенного алгоритма можно использовать очевидный эффективный алгоритм на основе упорядочения объектов, например по неубыванию требуемых ресурсов. Можно использовать приведенные в табл. 5 базовые типы графа G.

Таблица 5

Тип орграфа	Значения коэффициентов целевой функции: $c_i = 1$	Значения коэффициентов целевой функции: $c_i \in (0, \infty)$
Несвязный B	* $O(n \log n)$	! a $O(n^2/\varepsilon)$
Линейный порядок (цепочка) C	* $O(n)$	* $O(n)$
Слоистый S	* $O(l \log l)$	! a $O(l^2/\varepsilon + n)$
Несколько цепочек NC	* $O(m n^2)$	! a $O(m n/\varepsilon)$
Древовидный с направлением дуг от корня T_1		! a $O(nd/\varepsilon)$
Древовидный с направлением дуг к корню T_2		! a $O(n^2/\varepsilon)$
Лес NT		! a $O(n^2/\varepsilon)$

В табл. 5 для моделей типа (3) с одним ресурсом приведены характеристики известных алгоритмов [10, 54]. В табл. 5 использованы следующие дополнительные обозначения: m — число цепо-

чек; d — число висячих вершин; l — максимальное число элементов в слое.

При сложных типах G достаточно разумным подходом к решению задачи (3) является метод локальных целей, в частности аппроксимация G графом из класса NT или S (разбиение R на взаимно упорядоченные слои) и последовательное решение аналогичных задач меньшей размерности и с более простой структурой графового ограничения.

ЗАДАЧИ ГЕНЕРАЦИИ ВАРИАНТОВ СЛОЖНЫХ ОБЪЕКТОВ

При генерации альтернативных вариантов сложных объектов (планов, проектов, управленческих воздействий) используются различные подходы, например: мозговая атака, синектический метод, морфологический анализ [70, 106]. Большинство подходов являются содержательными, организационными, а морфологический анализ может быть основан на использовании формальных моделей. В рамках морфологического анализа проводятся: исследование проектируемого объекта (выявление основных требований, компонентов и их взаимосвязи, построение соответствующих компонентам множеств вариантов реализации — морфоклассов, разработка системы оценивания); определение допустимых сочетаний элементов морфоклассов; оценивание элементов морфоклассов по локальным критериям; построение допустимых вариантов объекта на базе элементов морфоуровней, их оценка и выбор лучших [23, 70, 107].

Можно выделить основные содержательные факторы классификации задач генерации альтернатив: тип объекта (число морфоклассов, взаимосвязь компонентов, например простая, иерархическая); тип ограничений (связи между морфоклассами, связь между ресурсами различных компонентов); тип критериев (число, взаимосвязь, монотонность и др.); тип задачи, в частности тип генерации (по прототипу, без прототипа); число искомых решений (одно, группа).

При построении сложного объекта часто можно использовать его иерархическую модель, в частности древовидную. Пусть объект состоит из компонентов $A=\{a_0, a_1, \ldots, a_n\}$ и может быть представлен в виде орграфа $G=(A, \rightarrow)$, где $\rightarrow$ — множество дуг, направленных от общих элементов к составляющим их частным; a_0 — корневая вершина. Пусть $R=\{a \in A \mid \Gamma a=\varnothing\}$ — множество висячих вершин, где Γ — отображение A в A, $\forall\, b \in R$ имеется $\Lambda(b)$ возможных вариантов реализации (морфокласс). Все множество вариантов реализации объекта (морфологическое пространство) [106]:

$$\Lambda = \prod_{b \in R} \Lambda(b).$$

С учетом допустимости сочетаний элементов морфоклассов Λ задается множество вариантов реализации объекта. Каждая компонента $a_i \in A$ может характеризоваться системой критериев

$\{K(a_i)\}$. От свойств перехода $\{K(b)\} \rightarrow \{K(a)\}$ $(a, b \in A, b \in \Gamma a)$ и числа критериев существенно зависит сложность решения задачи.

Построение объекта можно представить в виде многоэтапной схемы: построение на базе элементов нижнего иерархического уровня R допустимых вариантов реализации следующего уровня и т. д. Высокая размерность возникающих задач, сложность связи критериев разных уровней требуют упрощения процесса решения.

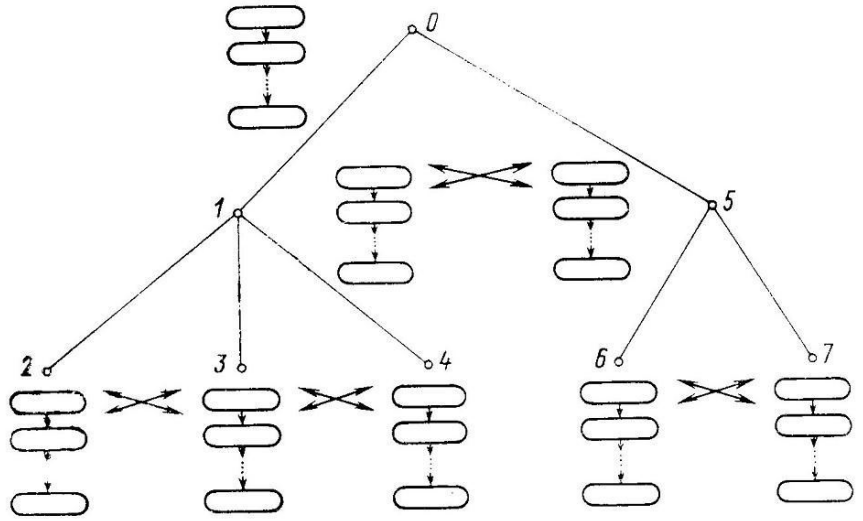


Рис. 8. Пример схемы построения сложного объекта

например за счет оценки на каждом этапе исходных вариантов реализации и использовании в дальнейшем наиболее перспективных. Это можно проводить на основе оценивания элементов морфоклассов по критериям, а затем в некоторой порядковой шкале, формирования элементов следующего иерархического уровня на основе лучших элементов предыдущего уровня и др. (рис. 8). Общая схема решения имеет следующий вид:

1. Построение иерархической модели сложного объекта (трехуровневое дерево).

2. Разработка системы критериев $\{K(a)\} \forall\, a \in A$ (например, стоимость, показатели эффективности, надежность и др.).

3. Формирование множеств вариантов реализации для компонентов нижнего иерархического уровня, т. е. формирование $\forall\, b \in R\, \Lambda(b)$ (Λ (2), Λ (3), Λ (4), Λ (6), Λ (7)).

4. Определение допустимых сочетаний вариантов реализации между собой для компонентов, составляющих компоненты следующего иерархического уровня (Λ (2), Λ (3) и Λ (4) с целью формирования $\Lambda(1)$; Λ (6) и Λ (7) — Λ (5)).

5. Проведение $\forall\ b \in R$ оценивания элементов $\Lambda\ (b)$ по критериям $\{K\ (b)\}$. Групповое ранжирование (расслоение) элементов $\Lambda\ (b)$ (выделение лучших, средних, худших и др.).

6. Построение вариантов реализации компонента следующего уровня на основе первоочередного использования лучших вариантов реализации элементов нижнего иерархического уровня с учетом допустимости их сочетания (построение вариантов реализации компонентов 1, 5). Если использование лучших вариантов невозможно, то замена некоторых из них на следующие в соответствующих ранжировках.

7. Повторение шагов *4—6* для элементов следующих уровней. Работа завершается после ранжирования вариантов реализации корневого элемента (элемента 0).

В случае одного аддитивного критерия и одного аддитивного ресурса и одноуровневой модели задача генерации сводится к модели рюкзачного типа. Пусть объект состоит из компонент $R=\{1, ..., i, ..., n\}$; q_i — число элементов (вариантов реализации) компонента $i\ (q = \max\limits_i q_i)$; x_{ij} — булева переменная $=1$, если в морфоклассе i выбирается элемент j и 0 — в противном случае $(j = \overline{1, q_i})$; $Y_{i,i+1} = \| y_{ek} \|_{q_i \times q_{i+1}}$ — матрица допустимых соответствий элементов морфоклассов i и $i+1$ $(i=\overline{1, n-1})$, где $y_{ek}=1$, если допустимо сочетание элемента l морфокласса i и элемента k морфокласса $i+1$, и 0 — в противном случае. Каждый элемент j морфокласса i характеризуется требуемым объемом ресурса b_{ij} и полезностью $c_{ij} \geqslant 0$. Модель определения лучшего варианта имеет вид:

максимизировать

$$\sum_{i=1}^{n} \sum_{j=1}^{q_i} c_{ij} x_{ij}$$

при условии:

$$\sum_{i=1}^{n} \sum_{j=1}^{q_i} x_{ij} b_{ij} \leqslant B\ ,$$

$$\sum_{j=1}^{q_i} x_{ij} = 1\ \ \forall\ i = \overline{1, n}\ ,$$

$$(x_i\ Y_{i,i+1}) \cup x_{i+1} = \overline{0}\ \ \forall\ i = \overline{1, n+1}\ ,$$

где B — общее ограничение по ресурсу, $x_i = (x_{i1}, ..., x_{iq_i})$, $\cup$ — операция покомпонентной дизъюнкции двух булевых векторов.

Для решения этой задачи можно применять ε-приближенный эффективный алгоритм с трудоемкостью $O\ (n^2 q_0^2 / \varepsilon)$ — модификацию алгоритма разбиения на интервалы. На каждом из n шагов

алгоритма (в соответствии с морфоклассами) для каждого элемента формируется набор решений (сетка по целевой функции с дискретом $\varepsilon' = \varepsilon p_i$, где p_i — максимум целевой функции или его оценка сверху на шаге i). Если допустимо соответствие всех элементов различных морфоклассов между собой, то модель превращается в блочную задачу о рюкзаке.

Можно рассмотреть модификации указанного алгоритма с целью формирования группы наилучших решений, например:

на каждом шаге в ε'-окрестности выбирать $r>1$ лучших решений;

после первого решения задачи дополнительно n раз провести решение, поочередно исключая из исходного множества элементы полученного на первом этапе решения.

Очевидно, что в качестве базового можно использовать и другие алгоритмы и процедуры (переборные, эвристические и др.).

ЗАДАЧИ О НАЗНАЧЕНИИ

Классическая задача о назначении имеет следующий вид [37]. Имеется множество работ $R=\{1, ..., i, ..., n\}$ и множество исполнителей $M=\{1, ..., j, ..., m\}$ $(m=n)$. Эффективность выполнения работ задается матрицей $A = \| a_{ij} \|_{n \times m}$, где a_{ij} — эффективность выполнения работы i исполнителем j. Требуется сделать такое назначение исполнителей на выполнение работ π: $\pi(i) = \pi(j) \Leftrightarrow i=j$, что суммарная эффективность

$$\sum_i a_{i\pi(i)} \to \max\ .$$

Для этой задачи предлагались прямые и двойственные алгоритмы, являющиеся, по сути, аналогами соответственно прямых и двойственных симплекс-алгоритмов [82]. Многие алгоритмы реализованы программно.

Естественным обобщением является несбалансированная задача о назначении, в которой число исполнителей m больше числа работ n, в результате чего используются не все исполнители. Если $n>m$, то получается эквивалентная задача [37]. Известна минимаксная задача о назначении (о назначении на узкие места), в которой в качестве критерия рассматривается

$$\min \max_i a_{i\pi(i)}\ .$$

К такой задаче сводится задача о конвейере, в которой n рабочих распределяются по n рабочим местам [32]. Каждый рабочий j может работать на месте i при скорости $\leqslant b_{ij}$ $(a_{ij} = -b_{ij})$. Требуется расставить рабочих по местам и максимизировать скорость конвейера. В работе [37] исследуется Σ-минимаксная задача, в которой в качестве критерия выступает сумма k $(1 \leqslant k \leqslant n)$ наибольших из выбранных элементов a_{ij}. Во многих практических задачах существуют запреты на назначение некоторых исполнителей на некоторые работы, т. е. из n^2 возможных назначе-

ний разрешены p. Тогда получается задача с запретами. В табл. 6 представлены оценки трудоемкости известных эффективных алгоритмов (q — максимальное число незапрещенных элементов в строке матрицы A) [32, 37, 82]. Иногда могут использоваться другие критерии, например минимизация суммы выбранных элементов a_{ij}, максиминный, но подходы к построению алгоритмов в этих случаях аналогичны.

Таблица 6

Тип задачи	Тип связи исполнитель — работа	
	Без запретов	С запретами
Классическая	$O(n^3)$	
Несбалансированная	$O(n^3)+O(mn)$	$O(n^2 q)$, $O(n^3+nq \log q)$
Минимаксная	$O(n^3)$	$O(np)$
Σ-минимаксная	$O(k m n^2)$	

Часто в задачах о назначении требуется учитывать дополнительно ресурсные ограничения (по каждому исполнителю, по системе в целом), тип назначений (один исполнитель — много работ, много исполнителей — одна работа, много исполнителей — много работ), предпочтения ЛПР или многокритериальность. При этом в основном задача становится NP-трудной [45]. Схема решения задачи о назначении с учетом предпочтений ЛПР и многокритериальностью может быть основана на том, что для каждой работы (или группы работ) исполнители разбиваются на упорядоченные группы (самых предпочтительных и др.). В целом решение сложных постановок задачи о назначении базируется на декомпозиции, эвристических схемах.

ЗАДАЧИ УПАКОВКИ

В последнее время задача упаковки (загрузки) стала часто применяться в практических и исследуемых приложениях [6, 7, 17, 35, 36, 55, 57, 90, 95, 100]. Суть ее заключается в том, что имеется некоторая ограниченная область $\Omega \leqslant R^n$ и множество n-мерных объектов $N=\{t_1, \ldots, t_i, \ldots, t_n\}$, которые размещаются в Ω. Постановки задачи могут иметь следующий вид.

Выбрать из множества N объект t_i (или группу объектов), размещающийся в области Ω с наибольшим коэффициентом заполнения.

Разместить в области Ω наибольшее количество элементов множества N. На возможные значения параметров размещения объектов могут накладываться дополнительные ограничения.

Разместить элементы N в области Ω так, чтобы наилучшим образом выполнялись необходимые требования (например, минимум отклонения центра тяжести объектов от заданной в Ω точки, минимум длины связывающей объекты сети, максимум надежности при минимуме стоимости получаемой конструкции).

Широкое использование получили задачи упаковки, в которых область Ω — один или несколько n-мерных гиперпараллелепипедов (в частности, двумерные, в виде полубесконечной полосы в R^2 с шириной l) и объекты (n-мерные параллелепипеды) размещаются так, чтобы их грани были параллельны граням области Ω (ортогональные упаковки).

Если область Ω — несвязанная, то каждую подобласть исходя из содержательной интерпретации могут называть рюкзаком, контейнером, процессором, исполнителем и др.; размерность, как правило, соответствует числу учитываемых ресурсов. Кроме того, практическая проблема может потребовать учета соответствия объектов подобластям Ω, иерархических ограничений (технологических, предпочтений ЛПР) в виде отношений (например, графов) на объектах. В некоторых задачах объекты состоят из простейших элементов, причем, если два объекта, включающие одинаковые элементы, помещаются в одной подобласти, то вместо двух или более одинаковых элементов учитывается только один. Рис. 9 и 10 иллюстрируют некоторые варианты задачи.

Особый случай задачи упаковки заключается в последовательном рассмотрении объектов с необходимостью оперативной упаковки (режим on line) в отличие от обычной постановки, когда все объекты известны заранее (режим off line) [17].

Среди различных приложений задачи можно выделить [6, 7, 55, 57, 90, 100]: проектирование планов, размещения оборудования производств; раскрой материала; формирование систем обслуживания; использование памяти в ЭВМ.

За исключением некоторых простейших постановок, задачи упаковки являются NP-трудными. Подход к построению приближенных алгоритмов их решения основан на использовании некоторого набора приемов [7, 17, 35, 36, 90, 99]:

упорядочение объектов (например, по невозрастанию или неубыванию общего объема или по конкретной координате — ресурсу);

поочередное размещение объектов в первую подходящую подобласть (в частности, в наилучшую из подходящих подобластей, например по характеристике заполнения);

предварительное разбиение (расщепление) Ω на подобласти определенного вида (на одинаковые или близкие по конфигурации для облегчения построения алгоритма, на соответствующие по конфигурации объектам или группам объектов);

учет (глобально, локально) степени заполнения подобласти;

разбиение исходного множества объектов на группы (близких объектов; объектов, составляющих некоторые заданные конфигурации);

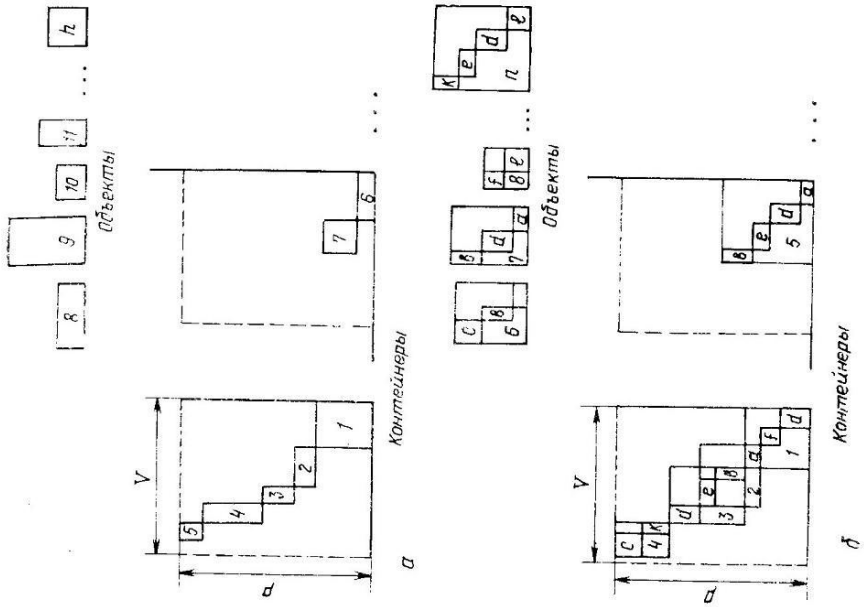


Рис. 10. Варианты двумерной задачи упаковки в контейнеры:

а — двумерная задача упаковки в контейнеры; *б* — двумерная задача упаковки в контейнеры с поворотом элементов

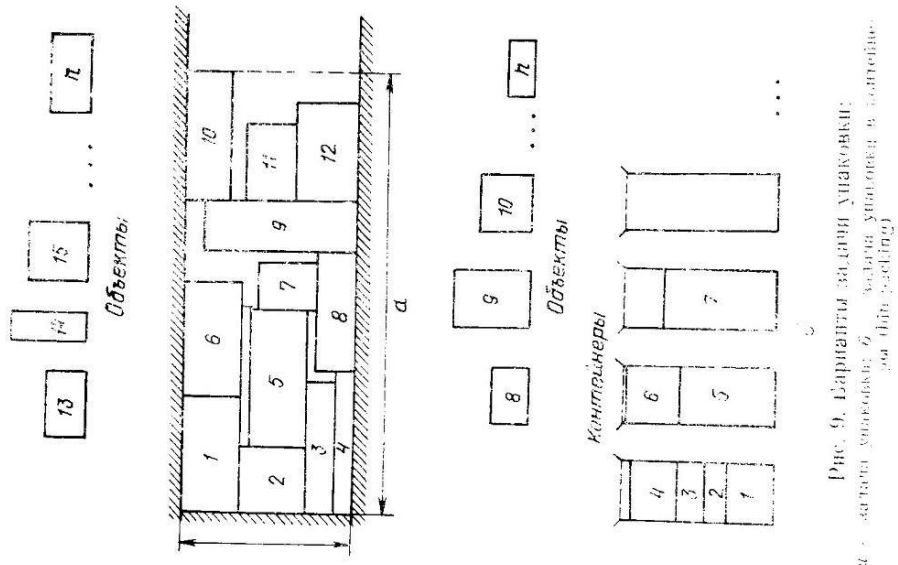


Рис. 9. Варианты задачи упаковки:

а — задача упаковки; *б* — задача упаковки с поворотом (bin packing)

релаксация исходной задачи (отбрасывание условия целочисленности) и др.

При оценке алгоритмов решения задач упаковки обычно используются априорные оценки качества в виде нижней и верхней оценок отношения значения целевой функции, получаемое при работе алгоритма, к оптимальному значению целевой функции ($\overline{U}$ и U соответственно). В работе [17] рассматриваются также асимптотическая оценка (при увеличении числа объектов $n \to \infty$) соответственно нижняя и верхняя $\overline{V}$, V и слабая асимптотическая оценки (когда дополнительно параметры объектов в пределе становятся близкими, равными) соответственно $\overline{W}$, W.

Таблица 7

№ задачи	Тип модели	Тип оценки					
		$\overline{U}$	U	$\overline{V}$	V	$\overline{W}$	W
1	$w_i=h_i=1,2$ $l=m$ on line		$^4/_3$				
2	$w_i=h_i \in (0, \infty)$ $l \in (0, \infty)$ off line		2				$^5/_4$
3	$w_i=1$ $h_i \in (0, \infty)$ $l=2$ off line		$1+\varepsilon$			1	1
4	$w_i=1$ $h_i \in (0, \infty)$ $l=2$ on line	$^3/_2$	$^3/_2$	$\geqslant {}^4/_3$		1	1
5	$w_i=1$ $h_i \in (0, \infty)$ $l=m$ off line		1,176— —1,22			1	1
6	$w_i=1$ $h_i \in (0, \infty)$ $l=m$ on line)	$\geqslant {}^3/_2$	$2-{}^1/m$	$\geqslant \frac{m^2}{m^2m+1}$		1	1
7	$w_i=1$ $h_i=1$ $l \in (0, \infty)$ off line				$\leqslant 1+\varepsilon$ $^{11}/_9$ $^{11}/_9-10^{-7}$		
8	$w_i \in (0, \infty)$ $h_i=1$ $l \in (0, \infty)$ on line		$\leqslant 2,7$	$\geqslant {}^3/_2$	$^5/_3$	$\geqslant {}^3/_2$	$^5/_3$
9	$w_i \in (0, \infty)$ $h_i \in (0, \infty)$ $l \in (0, \infty)$ off line		$\leqslant {}^5/_4$				$^5/_4$

Таблица 6

Число процессоров	Требуемый объем ресурса	Вид орграфа ограничений: Несвязный B			Древовидный T			Параллельно-последовательный P			Бесконтурный G		
		ΣC_i	C_{max}	$\Sigma a_i C_i$	$\Sigma e^{\lambda C_i}$	C_{max}	$\Sigma a_i C_i$	$\Sigma e^{\lambda C_i}$	C_{max}	$\Sigma a_i C_i$	$\Sigma e^{\lambda C_i}$	C_{max}	$\Sigma a_i C_i$
2	1	*	*	*	?	*	?	?	*	?	?	* $O(n^2)$	1
	1,2	*	*	?	?	?	?	?	?	?	?	$O(n^2)$	1
	$(0, \infty)$	*	$1-a$	$1-a$	?	$1-a$	$1-a$	?	$1-a$	$1-a$	?	$1-a$	$1-a$
$\geqslant 3$	1	*	*	*	?	* $O(n)$	?	?	?	?	?	$1-a$	$1-a$
	1,2	*	*	?	?	?	?	?	?	?	?	$1-a$	$1-a$
	$(0, \infty)$	$O(n \log n)$ *	$1-a$	$1-a$	$O(n \log n)$ *	$1-a$	$1-a$	?	$1-a$	$1-a$	?	$1-a$	$1-a$

Следует также отметить, что во многих случаях постановка задачи упаковки совпадает с другими комбинаторными задачами, в частности с задачами рюкзачного типа; многорюкзачными; о назначении; календарного планирования.

Рассмотрим задачу двумерной ортогональной упаковки прямоугольников в ограниченную с одной стороны полосу ширины l, где $l=m$ — целое или $l \in (0, \infty)$. Пусть длина основания прямоугольника i обозначается ω_i (1, 2, произвольная); высота — h_i (1, 2, произвольная). В табл. 7 для девяти типов задач представлены оценки известных алгоритмов [17]. Задачи 3—6 являются, по сути, задачами составления многопроцессорных расписаний с минимизацией общего времени обработки (высота прямоугольников — время выполнения заданий, ширина — требуемая для выполнения задания память), задачи 7, 8 — упаковки в контейнеры. Подробно характеристики задач составления многопроцессорных расписаний даны в табл. 8 [55]. Эти задачи представляют собой распределение заданий (объектов) между процессорами и определение порядка выполнения заданий на каждом процессоре. Графовое ограничение соответствует технологическому ограничению предшествования. В качестве критериев качества использованы

$$\Sigma C_i = \sum_{i=1}^{n} C_i\,,\quad C_{\max} = \max_{1 \leqslant i \leqslant n} C_i\,,$$

$$\Sigma a_i C_i = \sum_{i=1}^{n} a_i C_i\,,\quad \Sigma a_i e^{\lambda C_i} = \sum_{i=1}^{n} a_i e^{\lambda C_i}\,,$$

где C_i — момент окончания выполнения заданий i, т. е. высота верхней грани объекта (прямоугольника).

ЗАДАЧИ УНИФИКАЦИИ

Задача унификации функционально однородной группы изделий формулируется следующим образом [29, 30, 31]. Задан исходный список типоразмеров машин (изделий) $I=\{1, .., i, ..., m\}$. Каждый типоразмер $i \in I$ характеризуется набором эксплуатационных параметров $\{u_{it} \mid t=\overline{1, l}\}$, где u_{it} — значение t-го эксплуатационного параметра i-го типоразмера. Задано множество видов работ (проблемных областей применения и др.) $J=\{1, ..., j, ..., n\}$; каждая работа $j \in J$ предъявляет требования к эксплуатационным параметрам машин $\{v_{jt} \mid t=\overline{1, l}\}$, где v_{jt} — значение эксплуатационного параметра t для работы вида j. Типоразмер i может использоваться на работе j, если $u_{it} \geqslant v_{jt}\ \forall\, t=\overline{1, l}$. На перечне работ J определены объемы работ b_j. Известны затраты на проектирование, опытно-конструкторскую разработку и подготовку производства i-го типоразмера машины r_i, эксплуатационные затраты на выполнение единицы работы j типоразмером i — g_{ij}. Требуется найти ряд типоразмеров $\alpha \leqslant I$, который обеспечит выпол-

нение всех работ в полных объемах с минимальными суммарными затратами в сферах проектирования, производства, эксплуатации. Формально задача состоит в определении

$$\min_{\omega<I} P(\omega) = P(\alpha), \qquad (1)$$

где

$$P(\omega) = \sum_{i \in \omega} r_i + \sum_{j \in J} \min_{i \in \omega} c_{ij},$$

$$c_{ij} = b_i g_{ij}.$$

Результат решения задачи унификации $\alpha = (i_1, \ldots, i_q)$ фиксируется в виде стандартов параметров типов и основных размеров путем перечисления оптимальных значений эксплуатационных параметров машин

$$\{\{u_{i_1 t} \mid t = \overline{1, l}\}, \ldots, \{u_{i_q t} \mid t = \overline{1, l}\}\}$$

В литературе задача унификации также известна как задача размещения.

В рамках приведенной постановки можно непосредственно ввести $\forall\, i \in I$ подмножество работ $J_i \leqslant J$, на которых может быть использован типоразмер i:

$$J_i = \{j \in J \mid u_{it} \geqslant v_{jt} \forall\, t = \overline{1, l}\},$$

в простейшем случае $J_i = J \;\forall\; i \in I$.

Даже такая задача является *NP*-трудной. Для некоторых ее частных случаев, связанных со свойствами матрицы $C = \|c_{ij}\|_{mn}$, разработаны эффективные точные алгоритмы [8, 12, 30]. Эти алгоритмы построены по схеме динамического программирования. В табл. 9 приведены характеристики сложности некоторых частных случаев задачи унификации. Использованные при этом свойства перечислены ниже.

Таблица 9

Свойство матрицы затрат	Оценка сложности
Квазивыпуклая	* $O(m^2 n)$
Квазивогнутая	* $O(m^2 n)$
1 — связная	* $O(mn^2)$
Квазивыпукло-вогнутая	* $O(m^3 n)$ $O(m^3 + mn \log n)$
2 — связная	* $O(m^3 + mn \log n)$
3 — связная	!

Столбец матрицы C называют квазивыпуклым (квазивогнутым), если для любой тройки строк $i<k<l$ выполняется условие $c_{kj} \leqslant \max\{c_{ij}, c_{lj}\}$ ($c_{kj} \geqslant \min\{c_{ij}, c_{lj}\}$). Матрицу C называют квазивыпуклой (квазивогнутой), если каждый столбец ее квазивыпуклый (квазивогнутый)). Матрица является квазивыпукло-вогнутой, если каждый ее столбец либо квазивыпуклый, либо квазивогнутый. Матрица C обладает свойством p-связности, если для любых i, k разность $a_{ij} - a_{kj}$ меняет знак не более p раз при монотонном изменении j.

Для задачи унификации в общем случае предложен корректирующий алгоритм, основанный на следующем [29]. Исходная *NP*-трудная задача называется нерегулярной. Для многих частных случаев нерегулярных задач известны свойства исходных числовых данных, позволяющие использовать эффективные алгоритмы поиска оптимальных решений. Исходные данные для этих задач называют регулярными исходными данными (РИД), а все остальные наборы — нерегулярными исходными данными (НИД).

Схема корректирующего алгоритма имеет вид: аппроксимация НИД при помощи РИД; оценка точности аппроксимации; эффективное решение задачи на РИД; оценка того, на сколько полученное решение на РИД отличается по целевой функции от точного решения на НИД. Если требуемая точность не достигается, то выполняется разбиение задачи с НИД на подзадачи (ветвление, как в методе ветвей и границ [48]). Затем каждая из полученных подзадач решается по той же схеме. Алгоритм завершает свою работу, когда достигается требуемая точность решения задачи с НИД. Корректирующий алгоритм для решения задачи унификации базируется на некоторой модификации метода последовательных расчетов В. П. Черенина [48]. Корректирующий алгоритм реализован программно.

На практике могут возникать более сложные, чем рассмотренные, задачи унификации. Среди факторов, усложняющих модель унификации, можно выделить следующие [12]: изделия могут быть многоразового использования; работы и изделия могут быть взаимосвязаны, например в случае многоуровневости задачи, когда рассматриваются не только машины, но и комплектующие узлы; необходимость перестройки ряда (системы) под влиянием изменения условий функционирования (динамические типоразмерные ряды); необходимость учета предпочтительности типоразмеров исходя из целей более высокого уровня; многофакторность при оценке затрат; оценка параметров модели в качественных шкалах и др.

ЗАДАЧИ КОММИВОЯЖЕРА

Задача коммивояжера изучается уже более века. Классическая постановка задачи в комбинаторной форме имеет следующий вид [36, 62, 83]. Задано множество A, включающее n элементов (городов) и расстояние (длительность, стоимость)

$\rho(a_i a_j) \geqslant 0 \forall a_i, a_j \in A$. Требуется определить на множестве перестановок элементов А П перестановку $\pi^* = \langle a_{\pi^*(1)}, \ldots, a_{\pi^*(n)} \rangle$, для которой длина маршрута, проходящего через все города (не более одного раза), минимальна:

$$\min_{\pi \in \Pi} f(\pi) = f(\pi^*),$$

где $f(\pi) = \sum_{i=1}^{n-1} \rho(a_{\pi(i)}, a_{\pi(i+1)}) + \rho(a_{\pi(n)}, a_{\pi(1)})$, $a_{\pi(i)}$ — элемент из A, стоящий на i-м месте в π.

Данная постановка известна также, как задача определения очередной переналадки оборудования с выпуска одного вида продукции на другой с целью минимизации общих затрат на переналадку. Задача коммивояжера является одной из самых сложных в классе NP-трудных. Постановка задачи, в которой требуется найти ε-приближенное решение, также является NP-трудной. Это относится и к различным модификациям задачи, например: фиксировано начало маршрута; ищется перестановка, максимизирующая путь; имеется несколько коммивояжеров и др. Подходы к решению задачи основаны на применении: метода ветвей и границ; декомпозиции (разбиение множества A на подмножества, решение задачи для каждого подмножества и объединение решений); эвристических методов (например, «идти к ближайшему», двойного обхода минимального остового дерева [36]). В частности, для задачи коммивояжера построен статистически эффективный алгоритм [27], корректирующий алгоритм [29], субоптимальный алгоритм для задачи на максимум с оценкой $\geqslant 3/4$ [86]. Если выполняется неравенство треугольника

$$(\forall a_i, a_j, a_k \ \rho(a_i a_k) \leqslant \rho(a_i, a_j) + \rho(a_j, a_k)),$$

то указанные выше эвристические алгоритмы имеют оценки. Например, полиномиальный алгоритм двойного обихода минимального остового дерева позволяет получить решение, в котором длина маршрута меньше удвоенной оптимальной [36].

Задача коммивояжера может возникать в разнообразных практических ситуациях, например в задаче автоматического проектирования корпуса картера для коробки передач автомобиля [110]. В качестве исходных данных при проектировании рассматриваются система валов в сборе и ее характеристики (расположение валов типа параллельного, перпендикулярного и др.). За счет сечения системы валов получается ряд последовательных проекций системы валов. Частота сечения выбирается с целью фиксации изменения конструкции системы валов (включая изменение диаметров валов). По полученному набору проекций строится корпус картера в виде огибающей, системы касательных. На основе контура картера осуществляется построение внешней оболочки.

Поскольку картер используется для поддержки механизма (системы валов), необходимы опоры для подшипников и стенки связи между опорами, внешней оболочкой. Задача определения набора требуемых стенок сводится к задаче нахождения минимального пути в полном графе, т. е. к задаче коммивояжера. При этом вершины графа соответствуют подшипникам и внешнему корпусу, а веса дуг — стоимостям поверхностей минимальной стенки, связывающей соответствующие два объекта. При решении задачи можно вводить обязательные стенки (вес соответствующей дуги задается равным нулю) и запрещенные (соответственно вес определяется как очень большое число).

ВЫВОДЫ И РЕКОМЕНДАЦИИ

В настоящее время важное значение приобретают интеллектуальные средства формирования, поиска наилучших решений типа систем оптимизационного моделирования, интеллектуальных интерфейсов, систем поддержки (обеспечения) принятия решений, экспертных систем и др. При этом оптимизационные комбинаторные модели занимают особое место. Они соответствуют широкому кругу практических задач планирования, управления, проектирования, обеспечивают рациональную организацию работы вычислительных систем, играют существенную роль в поддержке процесса взаимодействия человека и ЭВМ. Это обусловливает актуальность исследований, пропаганды применения оптимизационных комбинаторных моделей.

Особого внимания требуют этапы построения моделей, алгоритмов решения комбинаторных моделей, так как они, как правило, определяют эффективность всей технологии поиска наилучших решений. Кроме того, возникает потребность в разработке специальных средств поддержки процесса выбора моделей и алгоритмов в виде справочных, экспертных систем.

На основе опыта разработки и эксплуатации автоматизированных систем можно выделить три их основных типа [91, 112, 114]: специализированные системы для решения конкретных прикладных задач; генераторы систем, позволяющие создавать специализированные системы на основе общих методологических принципов, пакетов программ, языка описания систем; наборы типовых средств, позволяющие формировать гибкие системы различного назначения. Это относится и к системам комбинаторного моделирования. Представляется целесообразным наряду с созданием универсальных ППП для решения задач дискретной оптимизации разработать специализированные системы комбинаторного моделирования, включить их в состав базового программного обеспечения различных ЭВМ, включая персональные [65, 87, 109].

ЛИТЕРАТУРА

1. Адельсон-Вельский Г. М., Диниц Е. А., Карзанов А. В. Потоковые алгоритмы. — М.: Наука, 1975. — 119 с.

2. Азгальдов Г. Г. Теория и практика оценки качества товаров. Основы квалиметрии. — М.: Экономика, 1982. — 256 с.

3. Алексеева Е. Ф., Стефанюк В. Л. Экспертные системы — состояние и перспективы. — Изв. АН СССР, Техн. киберн., 1984, № 5, с. 153—167.

4. Антонюк Б. Д. Разработка экспертных систем искусственного интеллекта в США. Препринт. — М.: ВНИИСИ, 1985. — 77 с.

5. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. с англ. / Под ред. Ю. В. Матиясевича. — М.: Мир, 1979. — 535 с.

6. Бакенрот В. Ю., Чефранов А. Г. Эффективность приближенных алгоритмов распределения программ в однородной вычислительной системе. — Изв. АН СССР, Техн. киберн., 1985, № 4, с. 135—148.

7. Применение методов теории расписаний при оптимизации загрузки контейнеров / А. С. Беленький, С. Д. Ильенкова, Е. В. Левнер и др. В сб.: Динамика неоднородных систем. Матеиалы семинара. — М.: ВНИИСИ, 1983, с. 71—78.

8. Белинская И. Г. Об одном классе полиномов от булевых переменных. В кн.: Управляемые системы. — Новосибирск: Ин-т математики СО АН СССР, 1981, вып. 21.

9. Белкин А. Р. Приближенная триангуляция матриц в задачах ранжирования и обработки межотраслевого баланса. — Изв. АН СССР, Техн. киберн., 1981, № 1, с. 26—31.

10. Белкин А. Р., Левин М. Ш. Комбинаторно-графовые модели обработки информации при принятии решений (препринт). — М.: Науч. совет по компл. пробл. «Кибернетика» АН СССР, 1985. — 52 с.

11. Белкин А. Р., Левин М. Ш. К обоснованию построения человеко-машинных процедур. — Тезисы докл. Всесоюз. конф. Теория, методология и практика системных исследований, секц. 6: Человеко-машинные методы исследования сложных объектов. — М.: ВНИИСИ, 1985, с. 82—84.

12. Береснев В. Л., Гимади Э. Х., Дементьев В. Т. Экстремальные задачи стандартизации. — Новосибирск: Наука, 1978. — 333 с.

13. Бешелев С. Д., Гурвич Ф. Г. Математико-статистические методы экспертных оценок. — М.: Статистика, 1980. — 263 с.

14. Бобкова А. И. Состояние и тенденции развития гибкого автоматизированного производства в Японии. Обзорн. информ. Экспресс-информация. — М.: ЦНИИТЭИ приборостроения, сер.: Автоматизированные системы управления, 1985, вып. 10. — 16 с.

15. Системы оптимизационного моделирования / В. А. Большаков, В. Е. Кривоножко, А. И. Пропой, А. Б. Ядыкин. Сб. тр.: Программное обеспечение систем оптимизации. — М.: ВНИИСИ, 1982, вып. 7, с. 3—16.

16. Браверман Э. М., Мучник И. Б. Структурные методы обработки эмпирических данных. — М.: Наука, 1983. — 463 с.

17. Вайнштейн А. Д. Задачи об упаковке прямоугольников в полосу (обзор). В кн.: Дискретные задачи оптимизации (управляемые системы). — Новосибирск: Ин-т математики СО АН СССР, 1984, вып. 25, с. 17—37.

18. Методы и алгоритмы автоматизированного проектирования сложных систем управления / В. Л. Волкович, А. Ф. Волошин, Т. М. Горлова и др. — Киев: Наукова думка, 1984. — 216 с.

19. Вопросы информационной технологии. — М.: ВНИИСИ, сб. тр., 1982, вып. 1.

20. Гаврилова Т. А. Представление знаний в экспертной диагностической системе АВТАНТЕСТ. — Изв. АН СССР, Техн. киберн., 1984, № 5, с. 168—175.

21. Гарусов В. Н. Принципы разработки интерактивной моделенастраиваемой системы программного обеспечения оптимизационных задач большой размерности. — В сб. тр.: Программное обеспечение систем оптимизации. — М.: ВНИИСИ, 1982, вып. 7, с. 16—35.

22. Гафт М. Г. Принятие решений при многих критериях. — М.: Знание, 1979. — 64 с.

23. Метод принятия решений по выбору рациональных компоновок автомобилей / М. Г. Гафт, Н. Н. Миловидов и др. — В кн.: Проблемы и методы принятия решений в организационных системах управления. — М.: ВНИИСИ, 1982, с. 77—88.

24. Геловани В. А., Ковригин О. В., Смольянинов Н. Д. Методологические вопросы построения экспертных интеллектуальных систем. В кн.: Системные исследования / Методол. пробл. Ежегодник. — М.: Наука, 1983, с. 254—278.

25. Генс Г. В., Левнер Е. В. Дискретные оптимизационные задачи и эффективные приближенные алгоритмы (обзор). — Изв. АН СССР, Техн. киберн., 1979, № 6, с. 9—19.

26. Гибкое автоматизированное производство / Под ред. С. А. Майорова и Г. В. Орловского. — Л.: Машиностроение, 1983. — 376 с.

27. Гимади Э. Х., Глебов Н. М., Перепелица В. А. Алгоритмы с оценками для задач дискретной оптимизации. — Сб.: Проблемы кибернетики. — М.: Наука, 1976, вып. 31, с. 35—42.

28. Глотов В. А., Павельев В. В. Векторная стратификация. — М.: Наука, 1985. — 94 с.

29. Гольденгорин Б. И. Корректирующий алгоритм решения некоторых задач дискретной оптимизации. — Док. АН СССР, 1983, т. 270, № 3, с. 525—528.

30. Гольденгорин Б. И., Иоффе А. Л. Полиномиальные алгоритмы для задач унификации с квазивогнутой и квазивыпукло-вогнутой матрицей затрат. — Изв. АН СССР, Техн. киберн., 1985, № 3, с. 224—227.

31. Гольденгорин Б. И. Корректирующие алгоритмы решения многомерных задач унификации. — Изв. АН СССР, Техн. киберн., 1984, № 6, с. 3—7.

32. Гордон А. Я. Один алгоритм решения минимаксной задачи о назначении. В кн.: Исследование по дискретной оптимизации. — М.: Наука, 1976.

33. Горностаева А. Н., Левин М. Ш., Махсон М. А. Задача планирования контроля качества при создании систем машин. В сб.: Тезисы докл. конф. Повышение эффективности деятельности НИИ и КБ с целью ускорения научно-технического прогресса. — М.: МДНТП, с. 83—87.

34. ГОСТ 26228—84. Системы производственные гибкие. Термины и определения. — М.: Изд-во стандартов, 1984. — 5 с.

35. Гудмен С., Хидетниеми С. Введение в разработку и анализ алгоритмов: Пер. с англ. / Под ред. В. В. Мартынюка. — М.: Мир, 1981. — 368 с.

36. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи: Пер. с англ. / Под. ред. А. А. Фридмана. — М.: Мир, 1982. — 416 с.

37. Диниц Е. А. О решении двух задач о назначении. В кн.: Исследования по дискретной оптимизации. — М.: Наука, 1976, с. 333—348.

38. Евланов Л. Г. Теория и практика принятия решений. — М.: Экономика, 1984. — 176 с.

39. Евтушенко Ю. Г. Методы решения экстремальных задач и их применение в системах оптимизации. — М.: Наука, 1982. — 432 с.

40. Емельянов С. В., Ларичев О. И. Многокритериальные методы принятия решений. — М.: Знание, 1985. — 32 с.

41. Кашаев Ю. Х., Потапов В. А., Пуринь Я. Я. Современные гибкие производственные системы и их компоненты./Обзорн. инф. Сер. С-1. Станкостроение. — М.: ВНИИТЭМР, 1985. — 68 с.

42. Кини Р. Л., Райфа Х. Принятие решений при многих критериях: предпочтения и замещения. — М.: Радио и связь, 1981. — 560 с.

43. Кнут Д. Искусство программирования. — М.: Мир, Т. 3, 1978. — 844 с.

44. Ковригин О. В., Смольянинов Н. Д., Чмырь А. Я. Экспертные медицинские диагностирующие системы. — Изв. АН СССР, Техн. киберн., 1982, № 5, с. 199—216.

45. Коган Д. И., Лиогонький М. И. Задача о назначениях с учетом индивидуальных предпочтений. — Кибернетика, 1983, № 6, с. 80—84.

46. Котик М. А., Емельянов А. М. Ошибки управления: психологические причины, метод автоматизированного анализа. — Таллин: Валгус, 1985. — 391 с.

47. Конвей Р. В., Маквелл В. Л., Миллер Л. В. Теория расписаний: Пер. с англ. / Под ред. Г. П. Башарина. — М.: Наука, 1975. — 360 с.
48. Корбут А. А., Финкельштейн Ю. Ю. Дискретное программирование. — М.: Наука, 1969. — 368 с.
49. Корягин Д. А., Карпов В. Я., Самарский А. А. Принципы разработки пакетов прикладных программ для задач математической физики. — ЖВМ МФ, 1978, т. 18, № 2, с. 458—467.
50. Криштопа И. В., Микаилов Г. Э., Перевозчикова О. Л. Основные характеристики диалоговых маршрутных систем. — Кибернетика, 1984, № 6, с. 42—48.
51. Функционирование и конструирование диалоговых маршрутных систем / В. В. Крижановский, И. В. Криштопа, М. Н. Мусаев и др. — Автоматика и телемеханика, 1983, № 1, с. 150—159.
52. Ларичев О. И., Никифоров А. Д. Анализ процедур решения многокритериальных задач математического программирования. — Тезисы докл. Всесоюз. конф. Проблемы и методы принятия решений в организационных системах управления. — М.: ВНИИСИ, 1984, с. 70—71.
53. Левин М. Ш. Об эффективном решении некоторых задач теории расписаний на сетях. — Кибернетика, 1980, № 1, с. 131—135.
54. Левин М. Ш. Одна экстремальная задача организации данных. — Изв. АН СССР, Техн. киберн., 1981, № 5, с. 103—112.
55. Левин М. Ш. Детерминированные задачи планирования при идентичных процессорах и одновременном поступлении заявок. — Изв. АН СССР, Техн. киберн., 1982, № 4, с. 51—57.
56. Левин М. Ш. Комбинаторные модели при принятии решений. — Сб. тр: Процедуры оценивания многокритериальных альтернатив. — М.: ВНИИСИ, 1984, вып. 9, с. 35—41.
57. Левин М. Ш., Махсон М. А. Вопросы методологии и практики системы государственного надзора / Обзорная информ., сер.: Управление качеством продукции. — М.: ВНИИКИ, 1985, вып. 1. — 32 с.
58. Леонтьев В. К. Дискретные экстремальные задачи. В кн.: Итоги науки и техники, сер.: Теория вероятностей. Математическая статистика. Теоретическая кибернетика, т. 16. — М.: ВИНИТИ, 1979, с. 39—102.
59. Липаев В. В. Распределение ресурсов в вычислительных системах. — М.: Статистика, 1979. — 247 с.
60. Липаев В. В., Филиппович В. В. Принципы и правила модульного построения сложных комплексов АСУ. — УСиМ, 1975, № 1, с. 15—22.
61. Литвак Б. Г. Экспертная информация: методы получения и анализа. — М.: Радио и связь, 1982. — 184 с.
62. Майника Э. Алгоритмы оптимизации на сетях и графах: Пер. с англ. / Под ред. Е. К. Масловского. — М.: Мир, 1981. — 323 с.
63. Мануэль Т. ЭВМ пятого поколения — осторожный оптимизм. — Электроника, 1984, т. 57, № 24, с. 63—73.
64. Мартин Дж. Планирование развития автоматизированных систем: Пер. с англ. / Под ред. В. М. Савинкова. — М.: Финансы и статистика, 1984. — 196 с.
65. Михалевич В. С., Кукса А. И. Методы последовательной оптимизации в дискретных сетевых задачах оптимального распределения ресурсов. — М.: Наука, 1983. — 207 с.
66. Миркин Б. Г. Анализ качественных признаков и структур. — М.: Статистика, 1980. — 320 с.
67. Митрофанов Б. В., Пантелеев Ю. Р. Задача синтеза системы электроснабжения. — Сб. тр.: Проблемы и методы автоматизированного проектирования и исследования сложных систем. — М.: ВНИИПАС, 1985, вып. 1, с. 96—105.
68. Немировский А. С., Юдин Д. Б. Сложность задач и эффективность методов оптимизации. — М.: Наука, 1979. — 384 с.
69. Нильсон Н. Искусственный интеллект. Методы поиска решений: Пер. с англ. / Под ред. С. В. Фомина. — М.: Мир, 1973. — 270 с.
70. Одрин В. М., Картавов С. С. Морфологический анализ систем. — Киев: Наукова думка, 1977. — 148 с.

71. Основы создания больших АСУ / Под ред. В. А. Барановка. — М.: Сов. Радио, 1979. — 360 с.
72. Панкова Л. А., Петровский А. М., Шнейдерман М. В. Организация экспертизы и анализ экспертной информации. — М.: Наука, 1984.—180 с.
73. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность: Пер. с англ. — М.: Мир, 1985. — 512 с.
74. Парасюк И. Н., Сергиенко И. В. Модульный подход к построению семейства пакетов прикладных программ. — Программирование, 1981, № 6, с. 29—34.
75. Подружко А. С., Творогов В. Б. Программный комплекс человеко-машинной системы оптимизации для моделирования развития экономики. Сб. тр.: Программное обеспечение систем оптимизации. — М.: ВНИИСИ, 1982, вып. 7, с. 65—76.
76. Поспелов Г. С., Поспелов Д. А. Искусственный интеллект — прикладные системы. — М.: Знание, 1985. — 48 с.
77. Растригин Л. А. Структурная адаптация пакета программ оптимизации. — В кн.: Анализ и моделирование экономических процессов. Межвузовский сборник. — Горький: Горьковский ун-т, 1980, с. 3—9.
78. Райфа Г. Анализ решений. — М.: Наука, 1977. — 408 с.
79. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. — М.: Мир, 1980. — 476 с.
80. Репо А. И. Становление информационного общества. — Проблемы информационных систем. — М.: МЦНТИ, 1984, № 3, с. 11—14.
81. Руа Б. Классификация и выбор при наличии нескольких критериев (метод ЭЛЕКТРА). В кн.: Вопросы анализа и процедуры принятия решений. — М.: Мир, 1976, с. 80—107.
82. Рубинштейн М. И. Об алгоритмах решения задачи о назначении. — Автоматика и телемеханика, 1981, № 7, с. 145—154.
83. Рыбников К. А. Введение в комбинаторный анализ. — М.: Изд-во МГУ, 1985 — 308 с.
84. Самойленко С. И. Алгоритмы целенаправленного стохастического поиска. — М.: Научный совет по компл. проблеме «Кибернетика» АН СССР, 1973. — 22 с. (рукопись депонирована в ВИНИТИ 18 янв. 1974 г., № 111-74 деп.).
85. Свами М., Тхуласираман К. Графы, сети и алгоритмы: Пер. с англ. / Под ред. В. А. Горбатова. — М.: Мир, 1984. — 455 с.
86. Сердюков А. И. Алгоритм с оценкой для задачи коммивояжера на максимум. В кн.: Дискретные задачи оптимизации (управляемые системы). — Новосибирск: Ин-т математики СО АН СССР, 1984, вып. 25, с. 80—86.
87. Сергиенко И. В., Лебедева Т. Т., Рощин В. А. Приближенные методы решения дискретных задач оптимизации. — Киев: Наукова думка, 1980.— 276 с.
88. Современное состояние теории исследования операций. / Под общей ред. Н. Н. Моисеева. — М.: Наука, 1979. — 464 с.
89. Солоненко В. Г. Автоматизированные интегрированные системы со сквозным циклом «Проектирование — изготовление». — Обзорн. инф. Сер.: Робототехника. Радиоэлектроника. — Минск: БелНИИНТИ, 1984. — 40 с.
90. Стоян Ю. Г., Емец О. А. О комбинаторных задачах размещения прямоугольников. — Экономика и математические методы, 1985, т. XXI, вып. 5, с. 869—881.
91. Тамм Б. Г., Тыугу Э. Х. Пакеты программ. — Изв. АН СССР, Техн. киберн., 1977, № 5, с. 111—124.
92. Танаев В. С., Гордон В. С., Шафранский Я. Н. Теория расписаний. Одностадийные системы. — М.: Наука, 1984. — 384 с.
93. Танаев В. С., Шкурба В. В. Введение в теорию расписаний. — М.: Наука, 1975. — 256 с.
94. Тарьян Р. Э. Сложность комбинаторных алгоритмов. — Киберн. сб Новая серия, вып. 17. — М.: Мир, 1980, с. 61—113.
95. Теория расписаний и вычислительные машины: Пер. с англ. / Под ред. Э. Г. Кофмана. — М.: Наука, 1984. — 334 с.
96. Анализ нечисловой информации (препринт) / Ю. Н. Тюрин, Б. Г. Литвак, А. И. Орлов и др. — М.: Науч. совет по компл. проблеме «Кибернетика», 1981.— 80 с.

97. Финкельштейн Ю. Ю. Приближенные методы и прикладные задачи дискретного программирования. — М.: Наука, 1976. — 264 с.
98. Фридман А. А. О некоторых современных направлениях в дискретной оптимизации. — Экономика и математические методы, 1977, т. XIII, вып. 5, с. 1115—1131.
99. Фуремс Е. М. Модели упаковки в многокритериальных задачах принятия решений при ограниченных ресурсах. Препринт. — М.: ВНИИСИ, 1985. — 64 с.
100. Проблема упаковки объектов по контейнерам при наличии многих критериев / Е. М. Фуремс, Г. Н. Васюнин, О. И. Ларичев, Ю. Я. Чернов. В кн.: Проблемы и методы принятия решений в организационных системах управления. — М.: ВНИИСИ, 1982, с. 84—91.
101. Шейнин Р. Л., Черешкин Д. С. Некоторые вопросы создания систем обеспечения управленческих решений. — М.: ВНИИСИ, сб. тр., 1982, № 1, с. 53—63.
102. Эванс И. О. Описание структуры решения задачи при помощи логических таблиц. В кн.: Современное программирование. — М.: Сов. радио, 1967, с. 40—62.
103. ЭВМ пятого поколения. Концепции, проблемы, перспективы./Под ред. Т. Мото-ока: Пер. с англ. — М.: Финансы и статистика, 1984. — 110 с.
104. Юдин Д. Б. Математическое программирование в порядковых шкалах. — Изв. АН СССР, Техн. киберн., 1982, № 2, с. 3—17.
105. Юдин Д. Б., Шоломов Л. А. Обобщенное математическое программирование и функции выбора. — Изв. АН СССР, Техн. киберн., 1985, № 3, с. 3—16.
106. Якимец В. Н. Исследование морфологического пространства вариантов системы. В сб. тр.: Модели и методы формирования и многокритериального выбора предпочтительных вариантов систем. — М.: ВНИИСИ, 1981, вып. 1, с. 6—23.
107. Якимец В. Н. Формирование лексикографически упорядоченных морфологических вариантов систем. — В сб. тр.: Модели и методы формирования и многокритериального выбора предпочтительных вариантов систем. — М.: ВНИИСИ, 1981, вып. 1, с. 96—105.
108. Coleman D. Decision support system. — Data Process., 1984, 26, No. 8, p. 35—36.
109. Dvorak S., Musset A. Basic in action. — London e. a.: Butterworths, 1984. — 291 p.
110. Reynier M., Fouet J. M. Automated design of crank cases: «The CARTER system». — Computer-aided design. 1984, vol. 16, No. 6, p. 308—311.
111. The hand book of artifical Intellegence. Vol. 2, Ed. by Barr, E. E. Feigenbaum. Los Altos. (Calif.), Kauffman, 1982. — 434 p.
112. Third International Conference on Decision Support Systems. — Inf. and Manag., 1984, 7, No. 4, p. 226—233.
113. O'Keefe R. M. Expert System and operation research mutual benefits.— J. Oper. Res. Soc., 1985, 36, No. 2, p. 125—129.
114. Walkinshaw I. Development and use of a flexible DSS. — Data process, 1984, 26, No. 8, p. 37—40.

СОДЕРЖАНИЕ