ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

М.Н. Боголюбова

СИСТЕМНЫЙ АНАЛИЗ И МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ В МАШИНОСТРОЕНИИ

Рекомендовано Учебно-методическим объединением вузов по образованию в области автоматизированного машиностроения (УМО АМ) в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлениям подготовки: «Технология, оборудование и автоматизация машиностроительных производств», «Конструкторско-технологическое обеспечение машиностроительных производств»

Издательство
Томского политехнического университета
2010

УДК 621.002:51(075.8)
ББК 34.41я73
Б74

Боголюбова М.Н.

Б74 Системный анализ и математическое моделирование в машиностроении: учебное пособие / М.Н. Боголюбова; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2010. – 123 с.

ISBN 978-5-98298-721-1

В пособии изложены основы системного анализа и методы математического моделирования; приведены примеры численных методов решения задач; представлены индивидуальные задания по основным разделам курса.

Предназначено для студентов, обучающихся по направлению 150900 «Технология, оборудование и автоматизация машиностроительных производств».

УДК 621.002:5(075.8)
ББК 34.41я73

Рецензенты

Доктор физико-математических наук, профессор
заведующий лабораторией Института оптики
атмосферы СО РАН

В.П. Лукин

Кандидат технических наук
директор ООО «Сибирская машиностроительная компания»

Э.Н. Панкратов

ISBN 978-5-98298-721-1

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Системный анализ и моделирование процессов и явлений в различных областях науки и техники в настоящее время является одним из основных способов получения новых знаний и высокоэффективных технологий. Математическое моделирование и исследования, проводимые с помощью ЭВМ, являются научно-обоснованными методами оценок характеристик сложных систем, используемых для принятия решений в различных сферах инженерной деятельности.

Методы системного анализа и математического моделирования позволяют прогнозировать поведение сложных систем, изучать влияние управляющих воздействий на функционирование системы, формировать различные структуры систем и оптимизировать их параметры в соответствии с заданными критериями эффективности.

Для проектирования, исследования и эксплуатации сложных конструкторских и технологических систем в машиностроении специалист должен владеть определенными знаниями, необходимыми для разработки моделей и алгоритмов решения поставленных задач, владеть способами их программной реализации на компьютере, а также иметь навыки использования пакетов прикладных программ.

Для этих целей существует хорошо разработанный математический аппарат численных методов. Имеется обширная литература по методам вычислений и программированию на алгоритмических языках, таких как Си, Паскаль, Бейсик и многих других. Все они, как правило, предполагают наличие у пользователя определенных знаний в этой области.

Целью изучения данной дисциплины является ознакомление с рядом разделов прикладной математики и применение системного анализа и методов математического моделирования для решения практических задач.

Данное пособие является руководством к изучению теоретического материала, а также к выполнению индивидуальных заданий по основным разделам курса «Системный анализ и математическое моделирование». Оно включает теоретическую часть, схемы алгоритмов, программы для ЭВМ, примеры решения задач, комплекты индивидуальных заданий по основным темам.

Учебное пособие предназначено для студентов машиностроительных специальностей вузов.

Глава 1. ЭЛЕМЕНТЫ СИСТЕМНОГО АНАЛИЗА

Понятия «система», «системный подход», «системное исследование», «системный анализ», «общая теория систем», «системотехника» широко используются в науке и технике. Их роль обусловлена тем, что системные исследования – это новое направление исследовательской деятельности.

Перед специалистами любого профиля часто возникают вопросы: как решить реальную проблему, как уменьшить сложность возникшей ситуации, как исследовать существующую систему и как осуществить проектирование новой системы. На эти вопросы отвечает прикладная наука, получившая название *системный анализ*.

Под *системным анализом* понимают всестороннее систематизированное (т. е. построенное на определенном наборе правил) изучение сложного объекта в целом вместе со всей совокупностью его сложных внешних и внутренних связей, проводимых для выяснения возможностей улучшения функционирования этого объекта. Предметом системного анализа являются большие и сложные системы, а также методы проектирования, управления и применения их в различных сферах научно-практической деятельности.

Основой системного анализа считают *общую теорию систем* и *системный подход.* Эти понятия будут рассмотрены в последующих разделах.

1.1. Понятие системы

Центральной концепцией теории систем, кибернетики, системного подхода, всей системологии является понятие системы. *Система* (от греч. *systema* – целое, составленное из частей; соединение) – множество элементов, находящихся в отношениях и связях друг с другом, которое образует определенную целостность, единство.

Рассмотрим понятие системы более подробно. Начнем с рассмотрения искусственных систем, создаваемых человеком. Любая деятельность человека носит, как правило, целенаправленный характер. Наиболее четко это прослеживается на примере трудовой деятельности. Цели, которые ставит перед собой человек, редко достижимы только за счет его собственных возможностей или внешних средств, имеющихся у него в данный момент. Примером такой ситуации, требующей создания автоматизированной системы управления, является случай, когда обычные способы сбора и переработки информации не обеспечивают необ-

ходимой полноты и быстроты ее обработки, что значительно снижает качество принимаемых управленческих решений. Такое стечение обстоятельств называется *проблемной ситуацией*. Проблемность существующего положения осознается в несколько «стадий»: от смутного ощущения, что «что-то не так», к осознанию потребности, затем к выявлению проблемы.

Отсюда следует, что *новую* систему порождает наличие неудовлетворенной потребности – проблемной ситуации.

Следующим шагом в построении системы является определение ее цели. *Цель* – это субъективный образ (абстрактная модель) несуществующего, но желаемого состояния среды, которое решило бы возникшую проблему. Есть образное выражение: «Без проблемы нет системы».

Вся последующая деятельность, способствующая решению этой проблемы, направлена на достижение поставленной цели. Другими словами, *система есть средство достижения цели*.

В инженерной практике момент постановки целей (формулировки технического задания) – один из важнейших этапов создания систем. Обычно цели уточняются итеративно, с многократными изменениями и дополнениями.

Далее определим *функцию* системы как способ достижения системой поставленной цели.

На следующем шаге создается *структура* системы, обеспечивающая выполнение функции. *Структура* определяется как совокупность элементов и внутренних связей между ними.

Функционирование и развитие системы удовлетворяется внешней средой. Источник восполнения и дополнения элементов структуры системы через внешнюю среду называют *внешними условиями*.

Таким образом, приведенные выше рассуждения образуют цепочку: *проблемная ситуация – цель – функция – структура – внешние условия*, представляющую собой логическую последовательность действий при построении (синтезе) системы, и носят конструктивный характер.

1.2. Связь между системой и средой

Перейдем от определения системы к визуальному эквиваленту. В самом общем виде систему принято изображать в виде «черного ящика» (рис. 1.1). Ни одна система не является абсолютно замкнутой. Взаимодействие системы со средой представляется внешними связями системы. Эти связи разделяются на входные и выходные. По входным и выходным связям между системой и средой происходит обмен матери-

альных, энергетических или информационных ресурсов: на входе система получает энергию, вещество или информацию из среды, а на выходе среда получает из системы конечный продукт.

Достигнутая цель – это запланированные заранее изменения в окружающей среде, какие-то продукты работы системы, предназначенные для потребления вне ее. Иначе говоря, система связана со средой и с помощью этих связей воздействует на среду. Подчеркнем еще раз, что выход системы в данной графической модели соответствуют слову «цель» в словесной модели системы.

Название «черный ящик» образно подчеркивает полное отсутствие сведений о внутреннем содержании «ящика»: в этой модели задаются, фиксируются, перечисляются только входные и выходные связи системы со средой. Пытаясь максимально формализовать модель, мы приходим к заданию двух множеств *X* и *Y* входных и выходных переменных, но никаких других отношений между этими множествами фиксировать нельзя (иначе это уже будет не «черный», а прозрачный ящик).

Такая модель, несмотря на внешнюю простоту и отсутствие сведений о внутренности системы, часто оказывается полезной.

Рис. 1.1. Модель «черного ящика»

Рассмотрим пример.

Перечислим входы *X* системы «легковой автомобиль».

Понятие входа связано с управляющим воздействием на систему, воздействием, «подталкивающим» систему к цели. Поэтому выделим в автомобиле в качестве входов те его элементы, которые предназначены для управления во время движения: руль, педали сцепления, газа и тормоза, рычаг переключения коробки передач, переключатели сигнализации и освещения, ручка стояночного тормоза.

Затем, учитывая, что регулирующие воздействия приходится осуществлять не только на ходу, в перечень входов автомобиля включаем регулировочные винты, гайки, эксцентрики, смазку, заправку. Нельзя не

учитывать входы в буквальном смысле. Добавляем двери салона и крышки багажника и капота.

Входное воздействие на автомобиль оказывает не только водитель, но и окружающая среда. Можно отметить, что свойства поверхности, по которой движется автомобиль, также оказывают входное воздействие: водителю приходится по-разному действовать при езде по асфальту, песку, гравию. Добавляем к списку входов механическое воздействие грунта на колеса, электрические и магнитные поля и многое другое. Очевидно, что список входов может быть еще продолжен.

Предлагается читателю рассмотреть выходы системы «легковой автомобиль» самостоятельно.

Рассмотренный пример свидетельствует, что построение модели «черного ящика» не является тривиальной задачей, так как на вопрос о том, сколько и какие именно входы и выходы следует включать в модель, ответ не прост и не всегда однозначен.

1.3. Состав системы

Очевидно, что вопросы, касающиеся внутреннего устройства системы, невозможно решить только с помощью модели «черного ящика». Для этого необходимы более развитые, более детальные модели.

При рассмотрении любой системы прежде всего обнаруживается то, что ее целостность и обособленность выступают как внешние свойства. Внутренность же «ящика» оказывается неоднородной, что позволяет различать составные части самой системы. При более детальном рассмотрении некоторые части системы могут быть, в свою очередь, разбиты на составные части и т. д. Те части системы, которые мы рассматриваем как неделимые, будем называть *элементами*. Части системы, состоящие более чем из одного элемента, назовем *подсистемами*. При необходимости можно ввести обозначения или термины, указывающие на иерархию частей (например, «подсистемы такого-то уровня»).

В результате получается *модель состава системы*, описывающая, из каких подсистем и элементов она состоит (рис. 1.2).

Построение модели состава системы только на первый взгляд кажется простым делом. Если дать разным экспертам задание определить состав одной и той же системы, то результаты их работы будут различаться и иногда довольно значительно.

Причины этого состоят не только в том, что у них может быть различная степень знания системы: один и тот же эксперт при разных условиях также может дать разные модели. Существуют, по крайней мере, еще три важные причины этого факта.

Во-первых, разные модели состава получаются вследствие того, что понятие элементарности можно определить по-разному. То, что с одной точки зрения является элементом, с другой − оказывается подсистемой, подлежащей дальнейшему разделению.

Во-вторых, как и любые модели, модель состава является целевой, и для различных целей один и тот же объект потребуется разбить на разные части.

Например, один и тот же завод для директора, главного бухгалтера, начальника пожарной охраны состоит из совершенно различных подсистем. То, что для одного обязательно войдет в модель, может совершенно не интересовать другого.

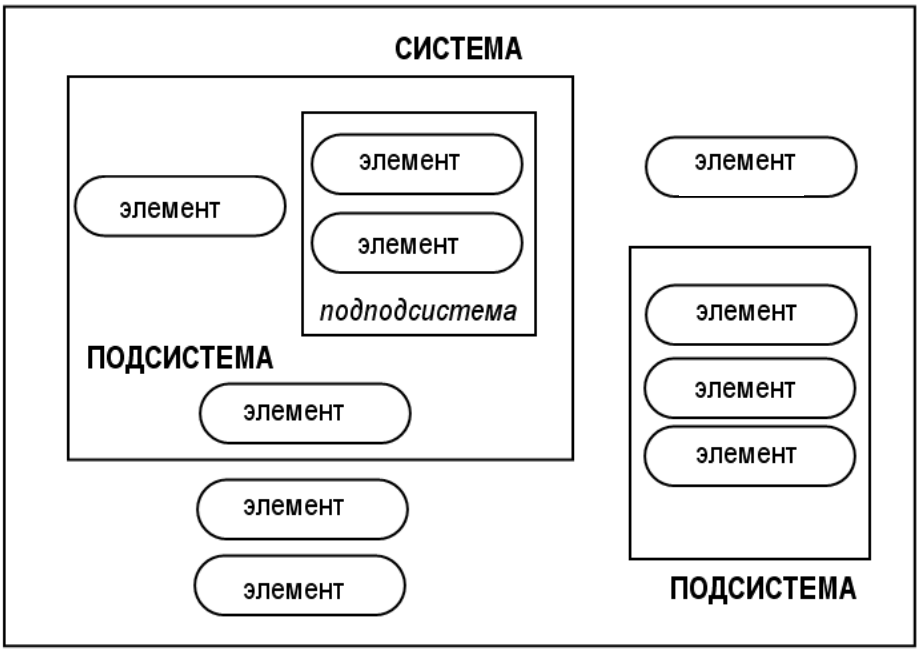


Рис. 1.2. Модель состава системы

В-третьих, модели состава различаются потому, что всякое разделение целого на части, всякое деление системы на подсистемы является относительным и в определенной степени условным.

Другими словами, границы между подсистемами условны, относительны, модельны.

1.4. Структура системы

В некоторых случаях для достижения ряда практических целей достаточно модели «черного ящика» или модели состава. Но очевидно, что есть вопросы, решить которые с помощью этих моделей нельзя. Чтобы получить, например, компьютер, недостаточно иметь «ящик» со всеми отдельными его деталями. Необходимо еще правильно соединить

все детали между собой, т. е. установить между элементами определенные связи – отношения.

Совокупность *необходимых* и *достаточных* для достижения цели отношений между элементами называется *структурой* системы.

Отношения между элементами могут быть самыми разнообразными. Однако можно попытаться их классифицировать и, по возможности, перечислить. Трудность состоит в том, что мы знаем не все реально существующие отношения и вообще неизвестно, каково должно быть конечное их число.

При всем многообразии реальных систем принципиально различных типов моделей систем немного: модель типа «черный ящик», модель состава, модель структуры, а также их разумные сочетания и, прежде всего, объединение всех трех моделей, т. е. структурная схема системы. Это относится как к статическим моделям, отображающим фиксированное состояние системы, так и к динамическим моделям, отображающим характер временных процессов, которые происходят с системой. Можно сказать, что структурная схема («белый ящик») получается как результат «суммирования» моделей «черного ящика», состава и структуры системы [1].

1.5. Типы систем

Существует множество различных вариантов классификации систем. Их можно разбить на два основных типа: предметный и категориальный. В предметном типе выделяются основные виды конкретных систем, существующих в природе и обществе (механические, биологические, социальные и т. п.). В категориальном типе системы классифицируются по общим характеристикам, присущим любым системам, независимо от их материального выражения.

Определим набор категориальных характеристик, с помощью которых следует оценивать основные компоненты систем. Каждый тип систем должен качественно отличаться от других, а так как качественное отличие обеспечивается количеством и составом входящих в систему элементов и характером отношений между ними, то требуется выделить количественные, составные и структурные категориальные характеристики компонентов систем.

Количественно все компоненты могут характеризоваться как *монокомпоненты* и *поликомпоненты*.

По составу компоненты систем классифицируются как *статические* и *динамические*. Для статической системы характерно то, что она

находится в состоянии относительного покоя, ее состояние с течением времени остается постоянным.

Динамические системы изменяют свое состояние во времени и делятся на функционирующие (процесс перехода из состояния в состояние не сопровождается сменой качества – цели) и развивающиеся (изменение состояния приводит к смене качества).

Структурно (по характеру отношений между компонентами системы, а также системы и среды) системы могут классифицироваться как *открытые* и *закрытые*, *детерминированные* и *вероятностные*, *простые* и *сложные*.

Системы делятся на открытые и закрытые по характеру их взаимоотношений со средой. Большинство систем *открытые*, так как они постоянно обмениваются веществом, энергией или информацией со средой.

Система называется *закрытой* (замкнутой), если в нее не поступают и из нее не выделяются вещество, энергия или информация.

Если знание в данный момент времени конечного множества входящих в систему элементов и отношений между ними позволяет установить состояние системы в любой последующий или любой предшествующий моменты времени, то такая система является *детерминированной*. Иначе говоря, поведение детерминированных систем полностью объяснимо и предсказуемо на основе информации об указанном выше множестве.

Для *вероятностной* (случайной, стохастической) системы знание упомянутого множества в данный момент времени позволяет только предсказать вероятность нахождения системы в том или ином состоянии в последующие моменты времени. Это означает, что поведение вероятностной системы определяется не только конечным множеством составляющих данной системы, но и объектами, не входящими в данное множество. О таких системах говорят как о системах *с элементами случайности*.

Существуют разнообразные определения *простых* и *сложных* систем. Например, одни авторы делят системы на *простые* и *сложные* относительно их моделей: если модель достаточно точно (адекватно) отображает поведение системы, то система является *простой* по отношению к модели, а когда модель неадекватна системе, то система является *сложной* по отношению к модели.

Другие авторы предлагают назвать систему простой, если ее результат на выходе, соответствующий поставленной цели, достигается с

помощью заданных средств, если же этих средств недостаточно, то система является сложной [2].

1.6. Специфика системного анализа

Выше в общих чертах была дана характеристика понятия «система» и классификация систем. Понятие системы является главным стержнем в «системном анализе», в «системном подходе», в «общей теории систем», в «системотехнике» и при проведении «системных исследований». Взятые в кавычки термины широко используются в современной науке и технике, поэтому необходимо разъяснить общее и специфичное в трактовке этих терминов как сути соответствующих направлений научно-технических исследований.

Понятие системы имеет чрезвычайно широкую область применения, так как практически каждый объект может быть рассмотрен как система, но в выделенных выше направлениях научно-технических исследований это понятие выполняет основополагающую роль, поскольку является их отличительной особенностью. Общим в данных направлениях является также то, что они базируются на следующих основных системных принципах:

- *целостность* (принципиальная несводимость свойств системы к сумме составляющих ее элементов и невыводимость из последних свойств целого, зависимость каждого элемента, свойства и отношения системы от его места, функций и т. д. внутри целого);
- *структурность* (возможность описания системы через установление ее структуры, т. е. сети связей и отношений системы; обусловленность поведения системы не столько поведением ее отдельных элементов, сколько свойствами ее структуры);
- *взаимозависимость системы и среды* (система формирует и проявляет свои свойства в процессе взаимодействия со средой, являясь при этом ведущим активным компонентом взаимодействия);
- *многоуровневость, иерархичность* (каждый компонент системы, в свою очередь, может рассматриваться как система, а исследуемая система представляет собой один из компонентов более общей системы);
- *множественность* описания системы (в силу сложности практически каждой рассматриваемой системы ее исследование требует построения множества различных моделей, каждая из которых описывает лишь определенный аспект системы).

В разъяснении терминов «системный анализ», «системный подход», «системные исследования» столь много общего, что в широком и нестрогом смысле эти термины часто употребляются как синонимы.

Но все-таки наиболее широким и общим понятием является понятие «*системные исследования*». Этим термином определяется совокупность современных научных и технических работ, концепций и проблем, базирующихся на системном подходе, конкретно – научном знании о системах (конкретные системные концепции) и общей теории систем.

Системный подход ориентирует исследователя на раскрытие целостности объекта и обеспечивающих ее механизмов, на выявление многообразных типов связей сложного объекта и сведение их в единую (системную) теоретическую картину. Системный подход включает такие методы и средства исследования, которые приложимы к любым системам или достаточно широким их классам.

Конкретные системные концепции включают различные специальные теории систем, системные модели и разработки научных и технических дисциплин, системотехнику, исследование операций, эргономику и др.

Особой областью системных исследований является *общая теория систем*, которая стремится выполнить функцию обобщения и выработки принципов построения специального системного знания, или, иными словами, выработать такой язык и понятия, которые легко переводились бы на язык конкретных применений и вместе с тем позволяли бы относить изучаемые или проектируемые системы к тому или иному классу формальных систем, тем самым уже на этой стадии обнаруживая изоморфизм (аналогичность) между системами.

В частности, полагают, что общая теория систем должна стать теоретическим фундаментом *системотехники*.

Системотехнику определяют как комплексную научно-техническую дисциплину, предметом исследования которой являются сложные системы, их проектирование, создание и эксплуатация. Существует множество других определений этой науки, что свидетельствует о ее развивающемся характере.

Под *системным анализом* в узком смысле понимают совокупность методологических средств и процедур, используемых для подготовки, обоснования и осуществления решений по сложным проблемам научно-технического, экономического, социального и т. п. характера. Привлечение методов системного анализа для решения данных проблем оказывается необходимым в силу того, что в процессе принятия решений приходится осуществлять выбор в условиях неопределенности, которая обусловлена наличием факторов, не поддающихся строгой количественной оценке.

Расширение сферы применения системного анализа тесно связано с распространением программно-целевого метода управления. Основные принципы системного анализа заключаются в следующем:

1) процесс принятия решений должен начинаться с выявления и четкого формулирования конечных целей, а также критериев, по которым может оцениваться их достижение;

2) необходимо рассматривать всю проблему как целое, т. е. как единую систему, и выявлять все последствия и взаимосвязи каждого частного решения;

3) необходимы выявление и анализ возможных альтернативных путей достижения цели;

4) цели отдельных подразделений (подсистем) не должны вступать в конфликт с целями всей программы по решению сложной проблемы или, иными словами, системы в целом.

1.7. Основные методы системного анализа

Всякая система может изучаться с двух точек зрения: извне и изнутри. Первое означает рассмотрение взаимодействия системы со средой, т. е. рассмотрение функции системы. Второе предполагает изучение внутреннего строения системы, т. е. рассмотрение ее структуры.

Очевидно, что функция системы и ее структура взаимосвязаны. В ряде задач можно ограничиться изучением либо только структур, либо только функций системы. Этому соответствуют структурный и функциональный методы системного анализа. Те же задачи системного анализа, которые требуют одновременного учета устройства системы и ее функций в единстве, решаются на основе структурно-функционального метода. Центральной процедурой в системном анализе является построение модели, отображающей все факторы и взаимосвязи реальной ситуации (системы), которые могут проявиться в процессе осуществления решения (управления). Метод моделирования широко используется в системном анализе и теории систем.

1.7.1. Структурный метод

Структурный метод используется тогда, когда требуется определить, из каких частей состоит система и как эти части связаны между собой. Основу структурного метода образует выявление структуры как совокупности элементов и их отношений, инвариантных при некоторых преобразованиях. Данный метод представляет собой двухэтажную процедуру:

1) выяснение состава системы, т. е. полное перечисление элементов;

2) определение (анализ или синтез) структуры системы.

Нетрудно убедиться в том, что в окружающем нас мире существуют различные по материальному воплощению системы, имеющие оди-

наковые цели. Логично сделать вывод о том, что эти системы должны иметь нечто обязательно общее в своей структуре, что позволяет именовать их системами одного назначения.

Это общее подводится под понятие формальной структуры как совокупности функциональных элементов и их отношений, необходимых и достаточных для достижения системой заданной цели.

Подставляя вместо функциональных элементов соответствующие материализованные элементы, получаем материальную структуру, т. е. материальная структура представляет собой реальное воплощение формальной структуры.

Примером, поясняющим суть введенных понятий, может служить система, целью которой является указание времени (часы).

Формальная структура часов (рис. 1.3) представляет собой совокупность трех функциональных элементов – датчик времени (Д), индикатор (И), эталон времени (Э) – и отношений между ними.

При этом необходимыми и достаточными отношениями между перечисленными тремя элементами являются: синхронизация датчика с эталоном, однозначная связь датчика с индикатором и градуировка индикатора по эталону. Данная формальная структура присуща часам любой природы.

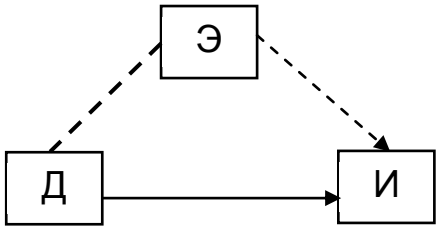


Рис. 1.3. Формальная структура часов

Материальная структура часов определяется конкретной конструкцией часов и является реализацией описанной формальной структуры.

Приведенный пример позволяет сделать два важных вывода:

1) фиксированной цели соответствует одна и только одна формальная структура системы;

2) одной формальной структуре может соответствовать множество материальных структур.

Первый этап структурного анализа − этап выяснения состава системы – не всегда является однозначным, и в настоящее время нам неизвестен общий подход к выяснению состава системы. Полезную роль на этом этапе структурного анализа может выполнить вполне очевидное

утверждение о том, что определение состава формальной структуры эквивалентно выделению дескрипторов (ключевых слов и предложений) в определении цели.

Второй этап реализации структурного метода – определение (анализ или синтез) структуры системы – также не имеет общего решения, пригодного для широкого класса систем. На данном этапе полезно использовать то, что формальная структура является логической структурой цели системы.

1.7.2. Функциональный метод

Слово *функция* происходит от латинского *functio*, что означает совершение, исполнение. Именно такое смысловое содержание этого понятия выявляется при рассмотрении логической последовательности действий при построении системы: *проблемная ситуация − цель − функция − структура − внешние условия.*

Отсюда следует, что всякий объект рассматривается как система только в том случае, если какое-то из его свойств используется для достижения поставленной цели. В итоге мы приходим к следующему определению: *функция системы есть ее свойство в динамике, приводящее к достижению цели*. Данное определение является дескриптивным (описательным).

Определим функцию системы конструктивно. Для этого необходимо указать способ описания функции и способ сравнения различных функций. В этой связи введем ряд понятий. Любой объект, в том числе и системный, выделяется из окружающей среды благодаря его отличительным особенностям, проявляющимся во взаимоотношениях с этой средой. Такие специфические отношения принято называть характеристиками данного объекта.

Задание определенной цели позволяет выделить из всего множества характеристик их конечное число, которое представляется исследователю необходимым и достаточным для математического описания объекта. Такие характеристики называют существенными.

Далее для математического описания объекта необходимо ввести количественную меру для каждой существенной характеристики объекта. Характеристики, для которых можно ввести единый универсальный эталон сравнения, называются количественными характеристиками объекта (системы). Для данной характеристики вводится понятие параметра − числа, выражающего отношение между этой характеристикой и выбранным эталоном. Однако не для всех существенных характеристик объекта удается ввести единый, универсальный эталон сравнения.

Такие характеристики принято называть качественными характеристиками объекта (системы). Удобство параметрического описания объекта столь велико, что всегда предпринимаются попытки устранить трудности, обусловленные отсутствием универсального эталона, и ввести некоторые аналоги количественных параметров. При этом, как правило, идут двумя путями.

Первый состоит в том, чтобы проверять и фиксировать наличие или отсутствие качественной характеристики у данного объекта. Для этого используется два числа (например, 0 и 1); одно из этих чисел соответствует наличию проверяемого качества, а другое – его отсутствию.

По второму пути идут, когда при наличии данного качества у всех сравниваемых объектов существует необходимость сопоставить их между собой по степени выраженности данного качества. В этом случае за эталон принимается качественная характеристика любого из сравниваемых объектов в упорядоченном ряду. Таким образом, имеется возможность введения количественных параметров и для качественных характеристик.

Пусть в некоторый момент времени система характеризуется набором значений *n*-параметров. Тогда эти значения можно рассматривать как координаты определенной точки в *n*-мерном пространстве, которое принято называть пространством состояний. Точка в этом пространстве соответствует состоянию системы.

Функционирование системы проявляется в ее переходе из одного состояния в другое или в сохранении какого-либо состояния в течение некоторого промежутка времени. При этом время не входит в число параметров системы, функция системы проявляется в движении изображающей точки, соответствующей состоянию системы, по некоторой траектории в пространстве состояний.

Функционирование системы представим в следующем виде: X − пространство состояний; $x_i(t_k)$ − значение i-го параметра системы в момент времени t_k; $x(t_k) = \{x_1(t_k), ..., x_n(t_k)\}$ – вектор состояния системы в момент времени t_k; $\overrightarrow{x(t)}$ − траектория движения системы в пространстве состояния, если она определена для всех моментов времени t.

Так как целевое состояние может быть достигнуто движением по разным траекториям, оканчивающимся в целевой точке или области, возникает следующий вопрос: не все ли равно, по какой траектории двигаться к целевому состоянию? Ответ на этот вопрос лежит вне данной системы и определяется двумя внешними факторами:

1) ограничениями, накладываемыми на систему внешней средой;

2) оценкой качества траектории с точки зрения системы высшего уровня, задавшей цель данной системе.

Оценка качества функционирования системы достигается определением предпочтительности одной траектории движения перед другой траекторией. Способ определения предпочтительности траектории системы называется критерием качества функционирования системы.

Общепринятый способ задания качества функционирования состоит в задании целевой функции и ограничении на множестве траекторий. Последнее осуществляется с помощью совокупности характеристических функционалов, определенных на упомянутом множестве. На часть из этих функционалов накладываются ограничения, выполнение которых выделяет множество допустимых траекторий.

Конкретные значения другой части функционалов на одной из допустимых траекторий являются показателями эффективности функционирования системы. Если необходимо выбрать наилучшую из допустимых траекторий, то мы должны определить, что понимается под словом наилучший.

Этот выбор осуществляется путем определения целевой функции в зависимости от показателей эффективности и выдвижения требования о том, чтобы она достигла экстремального значения (наибольшего или наименьшего из возможных значений) при заданных ограничениях.

Траектория, на которой выдвинутое выше требование удовлетворяется, называется оптимальной.

1.7.3. Структурно-функциональный метод

Выше структура и функция системы рассматривались в отрыве друг от друга, однако, при синтезе и анализе реальных систем многие задачи не могут быть решены без рассмотрения связей между структурой и функцией систем.

Особенно явно эта потребность проявляется при проектировании и исследовании управляемых систем.

Задачи, требующие одновременного учета устройства системы и ее функций, решаются на основе структурно-функционального метода. Здесь различают два типа задач:

1) прямая задача – синтез структуры, вновь создаваемой или совершенствуемой системы путем анализа заданной функции;

2) обратная задача – определение функции существующей системы путем анализа ее структуры.

1.7.3.1. Прямая задача

Рассмотрим сначала решение прямой задачи. Она решается в два этапа. На первом этапе осуществляется анализ заданной цели (функции) путем ее декомпозиции, а на втором – синтез структуры.

На первом этапе декомпозиция осуществляется путем выяснения состава необходимых условий для достижения заданной цели. Эти условия формулируются как подцели (цели ближайшего нижнего уровня), перечень которых определяется принципом декомпозиции.

Принцип декомпозиции – это признак, по которому производится разложение на составные части. В свою очередь, подцели также могут быть подвергнуты декомпозиции. В итоге получается некоторая *иерархическая* структура цели, т. е. субординированная совокупность необходимых условий достижения главной цели.

Возникают естественные вопросы:

- на каком уровне декомпозиции следует останавливаться?
- в каком порядке применять принципы декомпозиции?

В каждом конкретном случае число уровней декомпозиции будет различным: декомпозиция прекращается после того, как на конце каждой ветви древовидной структуры подцелей будут получены элементарные цели, достижимые с помощью известных средств.

Относительно порядка применения принципов декомпозиции можно утверждать, что многие системы к этому безразличны, но есть и системы, для которых результат декомпозиции зависит от этого порядка.

Цели, полученные в результате декомпозиции, имеют следующие особенности:

- цели нижнего уровня иерархии подчинены целям верхнего уровня;
- цели верхнего уровня не могут быть достигнуты, пока не достигнуты все цели ближайшего нижнего уровня.

Полученная иерархия целей является основой для синтеза структуры системы на втором этапе решения прямой задачи.

Для достижения основной цели необходимо выбрать такую последовательность обеспеченных структур разных этапов, которая полностью покрывает иерархию целей. Назовем эту последовательность допустимой. Очевидно, чем больше дефицит имеющихся ресурсов, тем больше число таких допустимых последовательностей. Возникает вопрос: какую из допустимых последовательностей обеспеченных структур избрать для реализации? В общем случае для решения задачи выбора оптимальной допустимой последовательности требуется ввести *критерий оптимальности*, базирующийся на количественном сравнении целей, обеспеченных структур и их допустимых последовательностей.

Согласно принципу декомпозиции цели на одном уровне иерархии являются качественно различными. Для их количественного сопоставления необходима *ранжировка*, т. е. присвоение рангов подцелям одного уровня по некоторому общему для них качеству.

Итак, мы пришли к тому, что первоначальная иерархия целей должна быть дополнена:

1) перечнями ресурсов, которые необходимы для обеспечения каждой элементарной цели;

2) ранжировкой подцелей каждого уровня.

Иерархия целей, дополненная сведениями в соответствии с пунктами 1 и 2, называется *деревом целей*.

1.7.3.2. Дерево целей

Дерево целей представляет собой графическое изображение иерархии целей и подцелей данной системы, на котором связанные между собой задачи, соединены линиями. С помощью дерева целей осуществляется *декомпозиция* основной цели на подцели, фиксируется последовательность их достижения. Это дает возможность оценить каждую задачу с точки зрения ее важности для достижения *глобальной* цели. Эти оценки служат основой для распределения ресурсов по подсистемам, решающим отдельные задачи.

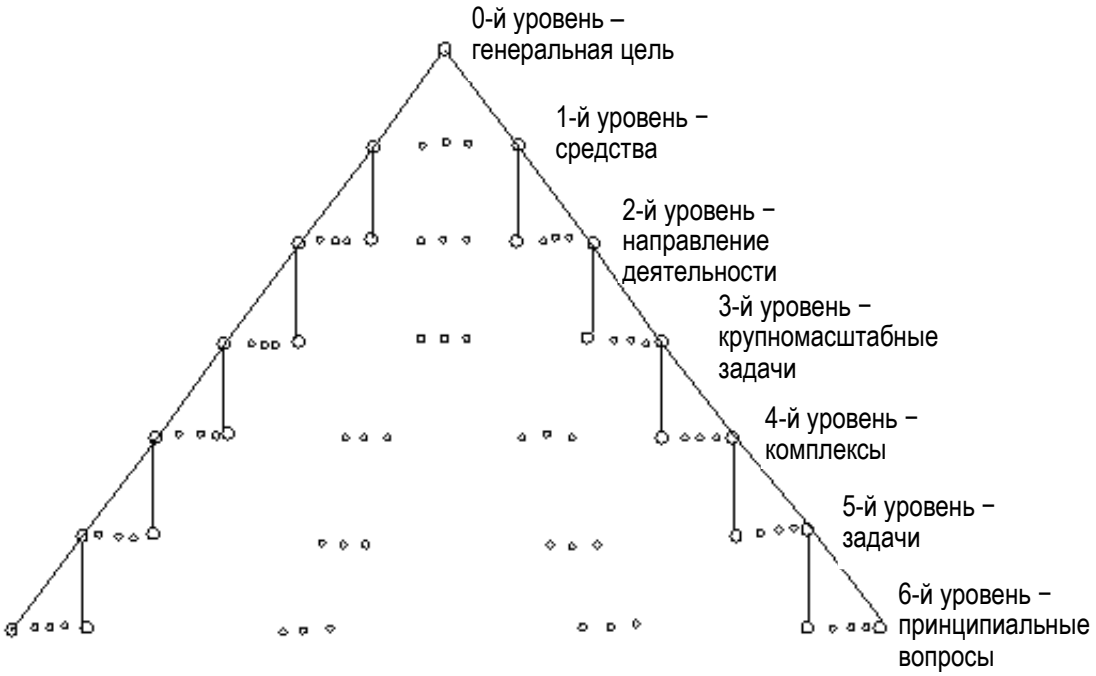


Рис. 1.4. Дерево целей

На рис. 1.4 приведен пример схемы дерева целей для станкостроения.

0-й уровень − наиболее полное удовлетворение потребностей в станках различных типов;

1-й уровень − увеличение объема выпуска станков, повышение качества продукции, соблюдение сроков поставки, расширение номенклатуры, увеличение срока службы;

2-й уровень − строительство новых предприятий, повышение производительности труда, улучшение технических параметров, улучшение экономических показателей, использование новых материалов, улучшение материально-технического снабжения, исследование пиковых нагрузок;

3-й уровень − технологическое и конструкторское проектирование с помощью ЭВМ, САПР, АСТПП и т. п.;

4-й уровень − разработка математической модели, оптимизация по критериям эффективности, разработка алгоритмов и программ;

5-й уровень − идентификация модели, многокритериальные задачи, экстремальные задачи, поиск оптимальных решений различными методами, методы адаптации и обучения;

6-й уровень − формализация описания исходной информации, достоверность описания объектов, использование графических пакетов прикладных программ, выбор языка программирования.

1.7.3.3. Обратная задача

Перейдем к рассмотрению обратной задачи, в которой требуется, зная состав и структуру системы, определить ее цель (функцию).

Пути решения этой задачи существенно зависят от того, как описан состав и структура системы, что именно о них известно, какова при этом степень подробности, точности и формализации, а также каков уровень априорных знаний человека или коллектива, взявшегося за решение обратной задачи. Возможно, что для решения задачи необходимы ресурсы, но их недостаточно, поэтому обратная задача оказывается неразрешимой в смысле определения основной функции системы.

Изложенное выше в какой-то степени объясняет отсутствие общего подхода к решению обратной задачи. Однако, если удается осуществить формализацию постановки обратной задачи и ограничить поиск возможных путей ее решения применением кибернетических методов, то самым существенным признаком, согласно которому отдается предпочтение тому или иному методу, является уровень априорной информации о составе и структуре системы.

В заключение заметим, что довольно широкий круг обратных задач удается представить в следующей постановке: по набору признаков данной системы требуется отнести эту систему к определенному классу, который и определяет ее функцию. Задачи в такой постановке решаются методами распознавания образов.

1.8. Отображение динамики системы

Следующий шаг в исследовании систем состоит в том, чтобы понять и описать, как система «работает», что происходит с ней самой и с окружающей средой в ходе реализации поставленной цели. Очевидно и подход к описанию, и степень подробности описания происходящих процессов могут быть различными. Но общим при этом является то, что разрабатываемые модели должны отражать поведение систем, описывать происходящие с течением времени изменения, последовательность каких-то этапов, операций, действий, причинно-следственные связи.

Системы, в которых происходят какие бы то ни было изменения со временем, будем называть динамическими, а модели, отображающие эти изменения, − динамическими моделями систем.

Для разных объектов и систем разработано большое количество динамических моделей, описывающих процессы с различной степенью детальности: от общего понятия динамики до формальных математических моделей конкретных процессов типа уравнений движения в механике или волновых уравнений в теории поля [1].

1.9. Функционирование и развитие

Различают два типа динамики системы: ее функционирование и развитие.

Под *функционированием* подразумевают процессы, которые происходят в системе (и окружающей ее среде), стабильно реализующей фиксированную цель (функционируют, например, станок, городской транспорт, часы, радиоприемник и т. д.).

Развитием называют то, что происходит с системой при изменении ее целей. Характерной чертой развития является тот факт, что существующая структура перестает соответствовать новой цели, и для обеспечения новой функции приходится изменять структуру, а иногда и состав системы, перестраивать всю систему.

Все указанные типы моделей являются формальными, относящимися к любым системам и, следовательно, не относящимися ни к одной конкретной системе. Чтобы получить модель заданной системы, нужно придать формальной модели конкретное содержание, т. е. решить, какие аспекты реальной системы включать как элементы модели избранного типа, а какие – нет, считая их несущественными.

Этот процесс обычно неформализуем, поскольку признаки существенности или несущественности в очень редком случае удается фор-

мализовать. Столь же слабоформализованными являются признаки элементарности и признаки разграничения между подсистемами.

В силу указанных причин процесс построения содержательных моделей является процессом интеллектуальным, творческим. Тем не менее, интуиции эксперта, разрабатывающего содержательную модель, немало помогают формальная модель и рекомендации по ее наполнению конкретным содержанием.

1.10. Моделирование

Под моделью данной системы следует понимать любую другую систему, обладающую той же формальной структурой при условии, что:

1) между системными характеристиками (функцией, структурой) модели и оригинала существует соответствие;

2) модель более доступна для исследования, чем оригинал.

При этом имеется в виду, что модель обладает только выделенными свойствами, а оригинал – многими другими. По отношению к своей модели систему принято называть *прототипом.*

Из определения модели следует, что для данной системы можно построить сколько угодно моделей с различной природой входящих в них элементов. Например, транспортную сеть можно промоделировать электрической схемой, гидравлической моделью либо в виде графа сети и т. п.

При моделировании абсолютное подобие не имеет места и стремятся к тому, чтобы модель достаточно хорошо отображала исследуемую сторону функционирования объекта. Для правильно построенной модели характерным является то, что она выявляет лишь те закономерности, которые нужны исследователю, и не рассматривает свойства системы, несущественные для данного исследования.

От постановки задачи моделирования до интерпретации полученных результатов существует ряд сложных этапов:

- идентификация реальных объектов;
- выбор вида моделей;
- построение моделей и их машинная реализация;
- взаимодействие исследователя с моделью в ходе машинного эксперимента;
- проверка правильности полученных в ходе моделирования результатов;
- выявление основных закономерностей, исследованных в процессе моделирования.

Каждый из этапов в зависимости от объекта имеет разную значимость и сложность выполнения.

1.10.1. Классификация видов моделирования систем

Классификационные признаки приведены на рис 1.5. В зависимости от характера изучаемых процессов в системе все виды моделирования могут быть разделены на детерминированные и стохастические, статические и динамические, дискретные, непрерывные и дискретно-непрерывные.

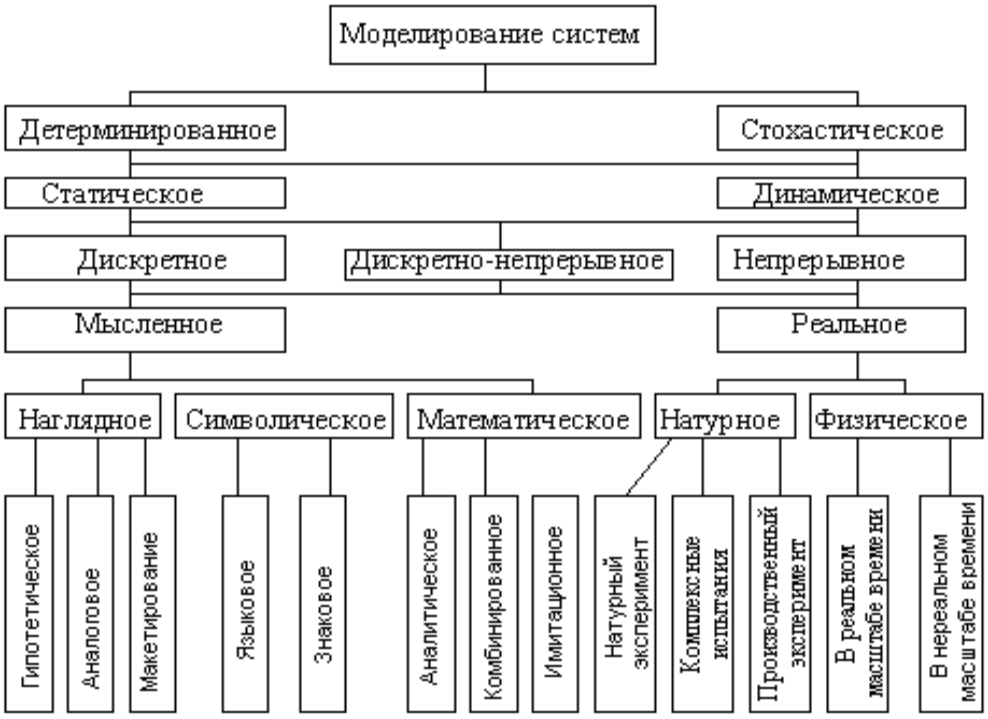


Рис. 1.5. Классификация видов моделирования систем

Детерминированное моделирование отображает детерминированные процессы, т. е. процессы, в которых предполагается отсутствие всяких случайных воздействий.

Стохастическое моделирование отображает вероятностные процессы и события. В этом случае анализируется ряд реализации случайного процесса и оцениваются средние характеристики, т. е. набор однородных реализаций.

Статическое моделирование служит для описания поведения объекта в какой-либо момент времени, а *динамическое моделирование* отражает поведение объекта во времени.

Дискретное моделирование служит для описания процессов, которые предполагаются дискретными, соответственно непрерывное моделирование позволяет отразить непрерывные процессы в системах.

Дискретно-непрерывное моделирование используется для случаев, когда хотят выделить наличие как дискретных, так и непрерывных процессов.

В зависимости от формы представления объекта (системы) можно выделить мысленное и реальное моделирование.

Мысленное моделирование часто является единственным способом моделирования объектов, которые либо практически нереализуемы в заданном интервале времени, либо существуют вне условий, возможных для их физического создания.

Например, на базе мысленного моделирования могут быть проанализированы многие ситуации микромира, которые не поддаются физическому эксперименту. Мысленное моделирование может быть реализовано в виде наглядного, символического и математического.

При *наглядном моделировании* на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и процессы, протекающие в объекте.

В основу *гипотетического моделирования* исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом изучаемого объекта. Гипотетическое моделирование используется, когда знаний об объекте недостаточно для построения формальных моделей.

Аналоговое моделирование основывается на применении аналогий различных уровней. Наивысшим уровнем является полная аналогия, имеющая место только для достаточно простых объектов.

С усложнением объекта используют аналогии последующих уровней, когда аналоговая модель отображает несколько либо только одну сторону функционирования объекта.

Существенное место при мысленном наглядном моделировании занимает *макетирование.* Мысленный макет может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования. В основе построения мысленных макетов также лежат аналогии, однако, обычно базирующиеся на причинно-следственных связях между явлениями и процессами в объекте.

Если ввести условное обозначение отдельных понятий, т. е. знаки, а также определенные операции между этими знаками, то можно реализовать *знаковое моделирование* и с помощью знаков отображать набор понятий – составлять отдельные цепочки из слов и предложений. Используя операции объединения, пересечения и дополнения теории множеств, можно в отдельных символах дать описание какого-то реального объекта.

В основе *языкового моделирования* лежит некоторый тезаурус. Последний образуется из набора входящих понятий, причем этот набор должен быть фиксированным. Следует отметить, что между тезаурусом и обычным словарем имеются принципиальные различия. Тезаурус – словарь, который очищен от неоднозначности, т. е. в нем каждому слову может соответствовать лишь единственное понятие, хотя в обычном словаре одному слову могут соответствовать несколько понятий.

Символическое моделирование представляет собой искусственный процесс создания логического объекта, который замещает реальный объект и выражает основные свойства его отношений с помощью определенной системы знаков или символов.

1.10.2. Математическое моделирование

Для исследования характеристик процесса функционирования любой системы математическими методами, включая и машинные, должна быть проведена формализация этого процесса, т. е. построена математическая модель.

Под *математическим моделированием* будем понимать процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели, позволяющее получать характеристики рассматриваемого реального объекта.

Вид математической модели зависит как от природы реального объекта, так и задач исследования объекта, а также требуемой достоверности и точности решения этой задачи. Любая математическая модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения к действительности.

Математическое моделирование для исследования характеристик процесса функционирования систем можно разделить на *аналитическое*, *имитационное* и *комбинированное*.

Для *аналитического моделирования* характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, конечно-разностных и

т. п.) или логических условий. Аналитическая модель может быть исследована следующими методами:

а) *аналитическим* – когда стремятся получить в общем виде явные зависимости для искомых характеристик;

б) *численным* – когда, не умея решать уравнений в общем виде, стремятся получить числовые результаты при конкретных начальных данных;

в) *качественным* – когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить устойчивость решения).

Наиболее полное исследование процесса функционирования системы можно провести, если известны явные зависимости, связывающие искомые характеристики с начальными условиями, параметрами и переменными системы. Однако такие зависимости удается получить только для сравнительно простых систем.

При усложнении систем исследование их аналитическим методом наталкивается на значительные трудности, которые часто бывают непреодолимыми. Поэтому, желая использовать аналитический метод, в этом случае идут на существенное упрощение первоначальной модели, чтобы иметь возможность изучить хотя бы общие свойства системы. Такое исследование на упрощенной модели аналитическим методом помогает получить ориентировочные результаты для определения более точных оценок другими методами.

Численный метод позволяет исследовать по сравнению с аналитическим методом более широкий класс систем, но при этом полученные решения носят частный характер. Численный метод особенно эффективен при использовании ЭВМ.

В отдельных случаях исследования системы могут удовлетворить и те выводы, которые можно сделать при использовании качественного метода анализа математической модели. Такие качественные методы широко используются, например, в теории автоматического управления для оценки эффективности различных вариантов систем управления.

В настоящее время распространены методы машинной реализации исследования характеристик процесса функционирования больших систем. Для реализации математической модели на ЭВМ необходимо построить соответствующий моделирующий алгоритм.

При *имитационном моделировании* реализующий модель алгоритм воспроизводит процесс функционирования системы во времени, причем имитируются элементарные явления, составляющие процесс, с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о со-

стояниях процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

Основным преимуществом имитационного моделирования по сравнению с аналитическим является возможность решения более сложных задач. Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики элементов системы, многочисленные случайные воздействия и др., которые часто создают трудности при аналитических исследованиях. В настоящее время имитационное моделирование – наиболее эффективный метод исследования больших систем, а часто и единственный практически доступный метод получения информации о поведении системы, особенно на этапе ее проектирования.

Когда результаты, полученные при воспроизведении на имитационной модели процесса функционирования системы, являются реализациями случайных величин и функций, тогда для нахождения характеристик процесса требуется его многократное воспроизведение с последующей статистической обработкой информации, и целесообразно в качестве метода машинной реализации имитационной модели использовать метод статистического моделирования.

Первоначально был разработан метод статистических испытаний, представляющий собой численный метод, который применялся для моделирования случайных величин и функций, вероятностные характеристики которых совпадали с решениями аналитических задач (такая процедура получила название метода Монте-Карло). Затем этот прием стали применять и для машинной имитации с целью исследования характеристик процессов функционирования систем, подверженных случайным воздействиям, т. е. появился метод статистического моделирования.

Таким образом, методом статистического моделирования будем в дальнейшем называть метод машинной реализации имитационной модели, а методом статистических испытаний (Монте-Карло) − численный метод решения аналитической задачи.

Метод имитационного моделирования позволяет решать задачи анализа больших систем, включая задачи оценки: вариантов структуры системы, эффективности различных алгоритмов управления системой, влияния изменения различных параметров системы. Имитационное моделирование может быть положено также в основу структурного, алгоритмического и параметрического синтеза больших систем, когда требуется создать систему с заданными характеристиками при определенных ограничениях, которая является оптимальной по некоторым критериям оценки эффективности.

При решении задач машинного синтеза систем на основе их имитационных моделей помимо разработки моделирующих алгоритмов для анализа фиксированной системы необходимо также разработать алгоритмы поиска оптимального варианта системы.

Далее в методологии машинного моделирования необходимо различать два основных раздела: статику и динамику, основным содержанием которых являются, соответственно, вопросы анализа и синтеза систем, заданных моделирующими алгоритмами.

Комбинированное (аналитико-имитационное) моделирование при анализе и синтезе систем позволяет объединить достоинства аналитического и имитационного моделирования. При построении комбинированных моделей проводится предварительная декомпозиция процесса функционирования объекта на составляющие подпроцессы, и для тех из них, где это возможно, используются аналитические модели, а для остальных подпроцессов строятся имитационные модели.

Такой комбинированный подход позволяет охватить качественно новые классы систем, которые не могут быть исследованы с использованием только аналитического или имитационного моделирования в отдельности.

1.10.3. Другие виды моделирования

При *реальном моделировании* используется возможность исследования различных характеристик либо на реальном объекте целиком, либо на его части. Такие исследования могут проводиться как на объектах, работающих в нормальных режимах, так и при организации специальных режимов для оценки интересующих исследователя характеристик (при других значениях переменных и параметров, в другом масштабе времени и т. д.). Реальное моделирование является наиболее адекватным, но при этом его возможности с учетом особенностей реальных объектов ограничены.

Например, проведение реального моделирования АСУ предприятием потребует, во-первых, создания такой АСУ, а во-вторых, проведения экспериментов с управляемым объектом, т. е. предприятием, что в большинстве случаев невозможно. Рассмотрим разновидности реального моделирования.

Натурным моделированием называют проведение исследования на реальном объекте с последующей обработкой результатов эксперимента на основе теории подобия. При функционировании объекта в соответствии с поставленной целью удается выявить закономерности протекания реального процесса.

Надо отметить, что такие разновидности натурного эксперимента, как *производственный эксперимент* и *комплексные испытания*, обладают высокой степенью достоверности,

С развитием техники и проникновением вглубь процессов, протекающих в реальных системах, возрастает техническая оснащенность современного научного эксперимента. Он характеризуется широким использованием средств автоматизации проведения, применением весьма разнообразных средств обработки информации, возможностью вмешательства человека в процесс проведения эксперимента, и в соответствии с этим появилось новое научное направление – *автоматизация научных экспериментов*.

Отличие эксперимента от реального протекания процесса заключается в том, что в нем могут появиться отдельные критические ситуации и определяться границы устойчивости процесса. В ходе эксперимента вводятся новые факторы и возмущающие воздействия в процессе функционирования объекта.

Одна из разновидностей эксперимента – комплексные испытания, которые также можно отнести к натурному моделированию, когда вследствие повторения испытаний изделий выявляются общие закономерности надежности этих изделий, в характеристиках качества и т. д. В этом случае моделирование осуществляется путем обработки и обобщения сведений, проходящих в группе однородных явлений.

Наряду со специально организованными испытаниями возможна реализация натурного моделирования путем обобщения опыта, накопленного в ходе производственного процесса, т. е. можно говорить о производственном эксперименте. Здесь на базе теории подобия обрабатывают статистический материал по производственному процессу и получают его обобщенные характеристики.

Другим видом реального моделирования является *физическое*, отличающееся от натурного тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием. В процессе физического моделирования задаются некоторые характеристики внешней среды и исследуется поведение либо реального объекта, либо его модели при заданных или создаваемых искусственно воздействиях внешней среды.

Физическое моделирование может протекать в реальном и нереальном (псевдореальном) масштабах времени, а также может рассматриваться без учета времени. В последнем случае изучению подлежат так называемые «замороженные» процессы, которые фиксируются в некоторый момент времени. Наибольшую сложность и интерес с точки зре-

ния верности получаемых результатов представляет физическое моделирование в реальном масштабе времени.

С точки зрения математического описания объекта и в зависимости от его характера модели можно разделить на модели *аналоговые* (непрерывные), *цифровые* (дискретные) и *аналого-цифровые* (комбинированные).

Под *аналоговой* моделью понимается модель, которая описывается уравнениями, связывающими непрерывные величины.

Под *цифровой* понимают модель, которая описывается уравнениями, связывающими дискретные величины, представленные в цифровом виде.

Под *аналого-цифровой* понимается модель, которая может быть описана уравнениями, связывающими непрерывные и дискретные величины.

Особое место в моделировании занимает *кибернетическое* моделирование, в котором отсутствует непосредственное подобие физических процессов, происходящих в моделях, реальным процессам. В этом случае стремятся отобразить лишь некоторую функцию и рассматривают реальный объект как «черный ящик», имеющий ряд входов и выходов, а также моделируют некоторые связи между выходами и входами. Чаще всего при использовании кибернетических моделей проводят анализ поведенческой стороны объекта при различных воздействиях внешней среды.

Таким образом, в основе кибернетических моделей лежит отражение некоторых информационных процессов управления, что позволяет оценить поведение реального объекта.

Для построения имитационной модели в этом случае необходимо выделить исследуемую функцию реального объекта, попытаться формализовать эту функцию в виде некоторых операторов связи между входом и выходом и воспроизвести на имитационной модели данную функцию, причем на базе совершенно иных математических соотношений и, естественно, иной физической реализации процесса.

Контрольные вопросы и упражнения

1. Дайте определение следующим понятиям: система, системный анализ, системный подход, системотехника.
2. Приведите примеры:
   а) системы, которая предназначена для выполнения определенной цели, но которую можно использовать и для других целей;
   б) системы, спроектированной специально для реализации одновременно нескольких различных целей;
   в) разных систем, предназначенных для одной и той же цели.

3. Обсудите проблему множественности входов и выходов на примере знакомой вам системы (станка с ЧПУ, гибкого производственного модуля, технологического процесса и т. п.). Перечислите при этом нежелательные входы и выходы. Выделите главную цель системы, дополнительные цели и ограничения.
4. Сравните формальную структурную схему какого-нибудь известного вам объекта с его реальной структурой. Обсудите расхождения.
5. Завод специализируется на сборке тракторов из готовых деталей. Какие существенные характеристики можно указать для данной системы? Что является входными и выходными величинами данной системы? Какие возмущающие воздействия могут возникнуть в этой системе?
6. Объясните понятия: состав системы, структура системы, функция системы, окружающая среда.
7. Назовите логическую последовательность действий при построении новой системы.
8. Приведите принципы классификации систем. Назовите основные типы систем.
9. Рассмотрите функционирование и развитие систем на примере конкретной системы.
10. Назовите основные методы исследования систем.
11. Какие задачи решаются при структурно-функциональном методе исследования систем.
12. Разработайте дерево целей для автоматизации технологической подготовки производства.
13. Объясните понятие «оптимизация решения задачи».
14. Приведите принципы классификации моделей систем.
15. Объясните особенности математического, имитационного, кибернетического моделирования.
16. Приведите другие виды моделирования.

Глава 2. РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

2.1. Решение нелинейного уравнения с одной переменной

Решение нелинейных уравнений с одной переменной представляет одну из важных задач прикладного анализа, необходимость в которой возникает в многочисленных и разнообразных разделах механики, техники, физики и других областях.

В общем случае нелинейное уравнение может быть записано в виде

$$F(x) = 0,$$

где функция $F(x)$ определена и непрерывна на конечном или бесконечном интервале $[a, b]$.

Корнем уравнения называется всякое число ξ, принадлежащее интервалу $[a, b]$, обращающее функцию $F(x)$ в нуль, то есть такое ξ, при котором $F(\xi) = 0$. В графической интерпретации это точка пересечения графика функции $F(x)$ с осью абсцисс (рис. 2.1).

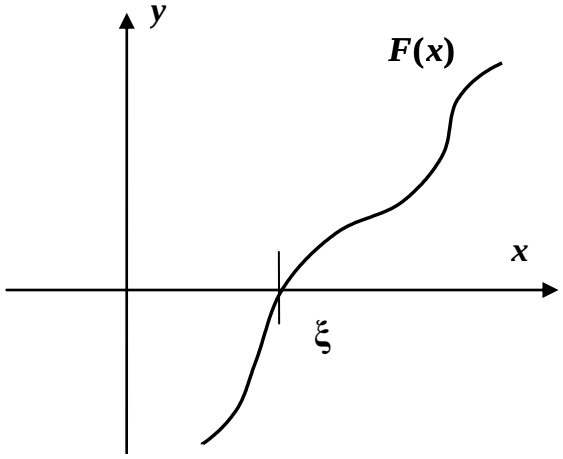


Рис. 2.1. График нелинейной функции $F(x)$

То есть нахождение корней уравнения $F(x) = 0$ сводится к отысканию координаты точки пересечения графика функции с осью абсцисс.

Нелинейные уравнения с одной переменной разделяются на алгебраические и трансцендентные. Уравнение называется алгебраическим, если путем алгебраических преобразований можно преобразовать его к каноническому виду:

$$F(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n = 0,$$

где a_0, a_1, ..., a_n – коэффициенты уравнения; x – неизвестное.

Алгебраическое уравнение имеет, по крайней мере, один корень, вещественный или комплексный. Алгебраические уравнения имеют способы решения аналитическими методами (точные методы). Они не являются предметом нашего рассмотрения, с ними можно ознакомиться в специализированной литературе.

Если функция не является алгебраической, то она называется трансцендентной. То есть уравнение невозможно привести к канонической форме. Примерами трансцендентных уравнений являются:

$$x - 10 \sin x = 0; \quad 2^x - \cos x = 0.$$

На практике подавляющее большинство нелинейных уравнений с одной переменной не решается путем аналитических преобразований (точными методами), поэтому их решают численными методами. Решить такое уравнение – значит установить, имеет ли оно корни, сколько корней и найти значение корней с заданной точностью.

Для решения трансцендентных уравнений разработано множество численных методов [5, 6]. Эти методы позволяют получить решение уравнения с заданной точностью.

Если обратиться к графической интерпретации, то это будет означать, что находится интервал величиной ε, внутри которого находится точное решение (рис. 2.2).

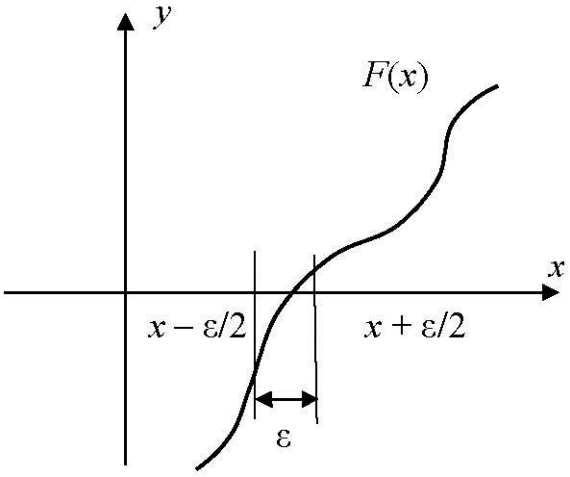


Рис. 2.2. Графическая интерпретация решения уравнения F(x) = 0

Задача численного нахождения действительных и комплексных корней уравнения $F(x) = 0$ состоит из двух этапов:

а) отделение корней, то есть нахождение достаточно малых окрестностей рассматриваемой области, в которых содержится одно значение корней;

б) уточнение корней, т. е. вычисление корней с заданной степенью точности.

Наибольшее распространение на практике получили численные методы решения рассматриваемого уравнения, такие как метод половинного деления, метод хорд, метод касательных (Ньютона), комбинированный метод, метод простой итерации и др.

Предметом нашего дальнейшего рассмотрения является метод половинного деления. С остальными методами решения нелинейных трансцендентных уравнений с одной переменной можно ознакомиться в специальной литературе, список которой приводится в конце пособия.

Использование численного метода половинного деления предполагает большое количество интенсивных математических вычислений. Использование ЭВМ на этом этапе представляется наиболее продуктивным. Программирование рассматриваемых алгоритмов не должно вызвать затруднений.

Для рассматриваемого метода приведен алгоритм решения, который можно без труда перевести на любой доступный язык программирования. Для облегчения задачи программирования приводится листинг программ на языке Си. Однако следует обратить внимание на то, что использованная при подготовке программы версия языка программирования может не совпадать с той, которой вы будете пользоваться, и может потребоваться внесение корректив в текст программы, обусловленных требованиями синтаксиса языка программирования.

2.1.1. Отделение корней

Первый этап численного решения уравнения $F(x) = 0$ состоит в отделении корней, т. е. в установлении промежутков, содержащих только один корень. Отделение корней во многих случаях можно провести графически. Принимая во внимание, что действительные корни уравнения – это точки пересечения графика функции $F(x)$ с осью абсцисс, достаточно построить график $F(x)$ и отметить на оси абсцисс отрезки, содержащие по одному корню. Построение графиков часто удается сильно упростить, заменив уравнение $F(x) = 0$ равносильным ему

$$f_1(x) = f_2(x).$$

Два уравнения называются равносильными (эквивалентными), если всякое решение каждого из них является решением и для другого, т. е. множество решений этих уравнений совпадают.

Если такое упрощение удается, то строятся графики функций $f_1(x)$ и $f_2(x)$, а потом на оси абсцисс отмечаются отрезки, локализующие абсциссы точек пересечения этих графиков. Примером может служить

уравнение $\sin 2x - \ln x = 0$. Для графического отделения корней необходимо построить графики функций

$$f_1(x) = \sin 2x \text{ и } f_2(x) = \ln x,$$

которые изображены на рис. 2.3.

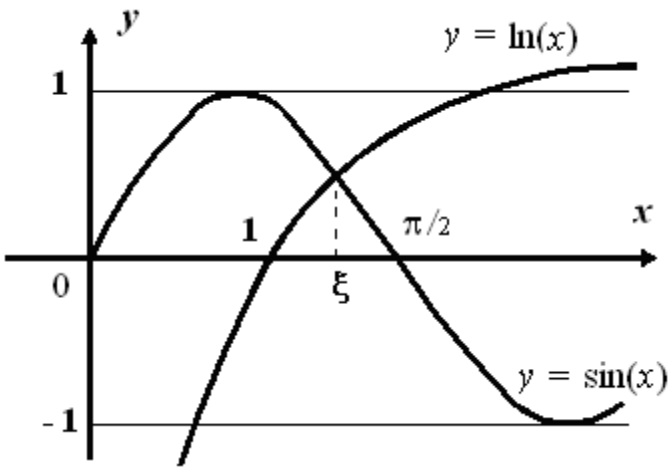


Рис. 2.3

Из графиков видно, что уравнение $\sin 2x - \ln x = 0$ имеет единственный корень, который принадлежит отрезку $[1; \pi/2]$.

При сомнительных случаях графическое отделение корней следует подтвердить расчетами. При этом следует пользоваться следующими очевидными положениями:

1. Если непрерывная на отрезке $[a; b]$ функция $F(x)$ принимает на его концах значения разных знаков, т. е. $F(a) \times F(b) < 0$, то уравнение имеет на этом отрезке по меньшей мере один корень.

2. Если функция $F(x)$ строго монотонна на интервале $[a; b]$, то уравнение $F(x) = 0$ имеет этом интервале единственный корень.

Для проверки произведенного отделения корней вычислим значения функции $F(x) = \sin 2x - \ln x$ на концах интервала [1; 1,5]:

$$F(1) = 0{,}909298; \; F(1{,}5) = -0{,}264344.$$

Построенный график наглядно показывает, что уравнение $\sin 2x - \ln x = 0$ имеет на интервале [1; 1,5] единственный корень.

Рассмотренный прием позволяет сузить интервал, полученный графическим способом. Так, вычислив $F(1{,}3)$ получим $F(1{,}3) = 0{,}253138 > 0$. Значит, интервалом отделения корней можно считать [1,3; 1,5]. Для дальнейшего сужения интервала, очевидно, потребуются интенсивные вычисления. Именно здесь наиболее целесообразно применять компьютер.

Для решения задачи отделения корней с использованием ЭВМ применяется следующий алгоритм.

Пусть имеется уравнение $F(x) = 0$, причем можно считать, что все интересующие корни находятся на конечном интервале $[A; B]$, в котором функция $F(x)$ определена и непрерывна. Требуется отделить корни уравнения, т. е. указать все интервалы $[a; b] \subset [A; B]$, содержащие по одному корню.

Будем вычислять значения $F(x)$, начиная с точки $x = A$, двигаясь вправо с некоторым шагом h (рис. 2.4).

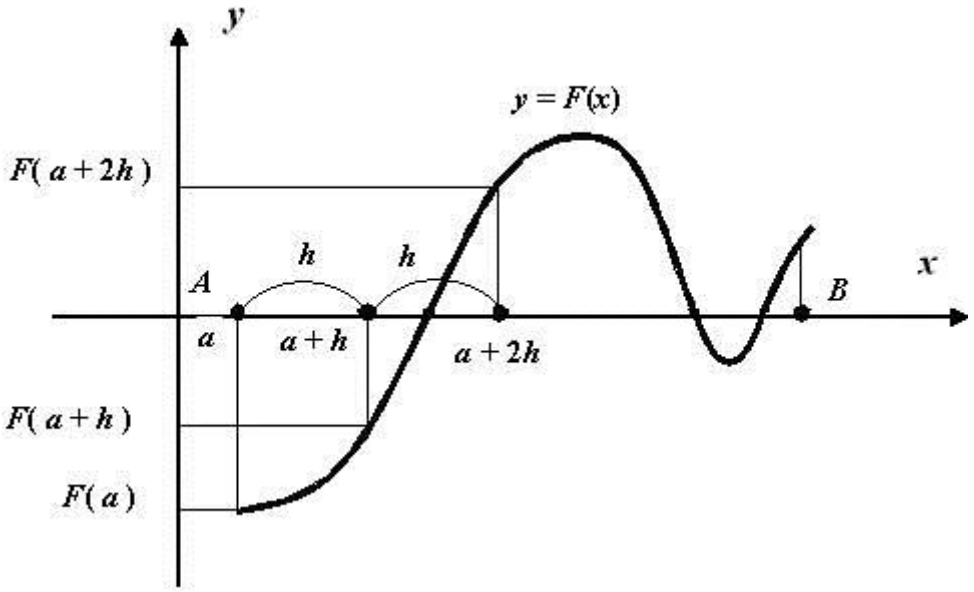


Рис. 2.4. Вычисление значений $F(x)$ с шагом $\Delta x = h$

Как только обнаружится пара соседних значений $F(x)$, имеющая разные знаки, и функция монотонна на этом интервале, так соответствующие значения аргумента x можно считать концами интервала, содержащего корень. Очевидно, что надежность рассмотренного подхода к отделению корней уравнения зависит как от характера функции $F(x)$, так и от выбранной величины шага h.

Действительно, если при достаточно малом значении h на концах выбранного интервала $[x; x + h]$ функция $F(x)$ принимает значение одного знака, естественно ожидать, что уравнение $F(x) = 0$ на этом интервале корней не имеет.

Это, однако, не всегда так: при несоблюдении условия монотонности функции $F(x)$ на интервале $[x; x + h]$ могут оказаться корни уравнения, как это показано на рис. 2.5.

Не один, а несколько корней может оказаться на интервале $[x; x + h]$ и при соблюдении условия $F(x) \times F(x + h) < 0$.

Предвидя подобные случаи (рис. 2.6), следует выбирать при отделении корней достаточно малые значения h.

Ниже приводится схема алгоритма (рис. 2.7) и текст программы на языке Си для отделения корней уравнения $\sin x - 0{,}2x = 0$.

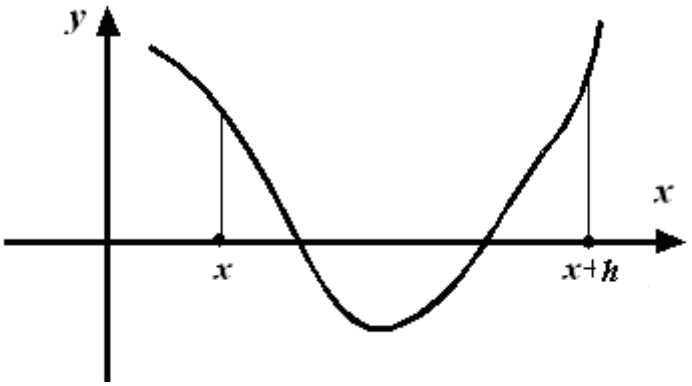


Рис. 2.5

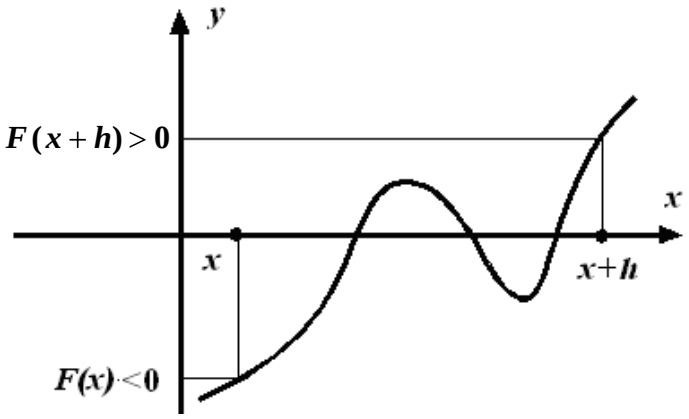


Рис. 2.6

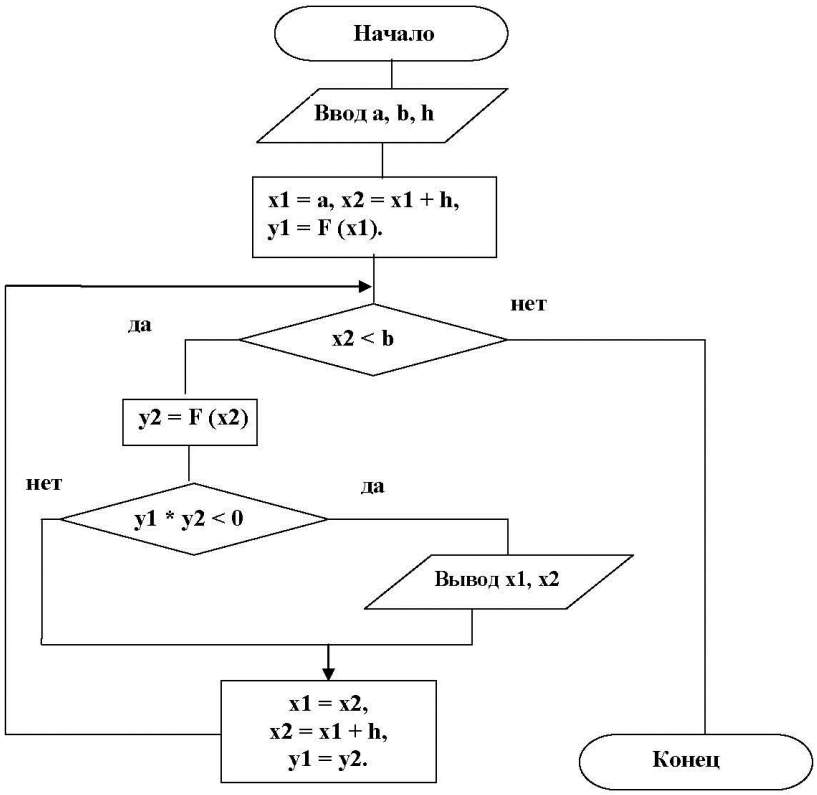


Рис. 2.7. Схема алгоритма отделения корней

Программа отделения корней

```
#include<stdio.h>
#include<math.h>
main()
{int k=0;
float A,B,h,x1,x2,y1,y2;
clrscr();
puts(«Введите A, B, h:»);
scanf(«%f%f%f»,&A,&B,&h);
x1=A; x2=x1+h;
y1=sin(x1)-0.2*x1;
while(x2<=B)
{y2=sin(x2)-0.2*x2;
if(y1*y2<0)
{k=k+1;
printf(«\n%d-й корень x1=%f x2=%f\n»,k,x1,x2);}
x1=x2; x2=x1+h; y1=y2;}
getch();}
```

Для того чтобы использовать эту программу для отделения корней другого уравнения, необходимо отредактировать программу, т. е. вписать выражение исследуемой функции.

Пример

Введите A, B, h:

–10 10 0,1

Решение

1-й корень	x1 = –2,599998	x2 = –2,499998
2-й корень	x1 = –0,099998	x2 = 0,000002
3-й корень	x1 = 2,500002	x2 = 2,600002

2.1.2. Уточнение корней

После того как произведено отделение корней уравнения $F(x) = 0$, т. е. известен каждый интервал $[a; b]$, на котором имеется единственный корень, причем функция $F(x)$ на этом интервале непрерывна, необходимо произвести уточнение корней. Из большого количества существующих методов уточнения корней рассмотрим один – уточнение корней методом половинного деления.

Разделим интервал $[a; b]$ пополам точкой $c = (a + b)/2$.

Если $F(c) \neq 0$, то возможны два случая: либо $F(x)$ меняет знак на интервале $[a; c]$ (рис. 2.8), либо на интервале $[c; b]$ (рис. 2.9).

Выбирая в каждом случае тот из интервалов, на котором функция меняет знак, и продолжая процесс половинного деления дальше, можно дойти до сколь угодно малого интервала, содержащего корень уравнения.

Рассмотренный метод можно использовать как метод решения уравнения с заданной точностью. Действительно, если на каком-то этапе процесса получен интервал $[a; b]$, содержащий корень, то, приняв приближенно $x = (a + b)/2$, получим ошибку, не превышающую значения $d = (a – b)/2$.

Метод связан с трудоемкими вычислениями, однако с успехом может использоваться для проведения расчетов на компьютере.

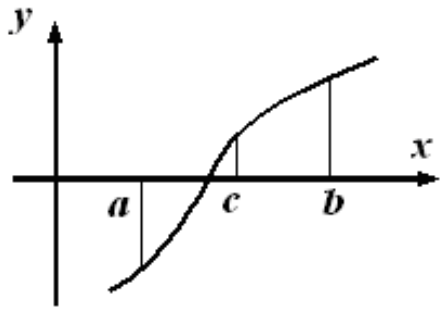


Рис. 2.8

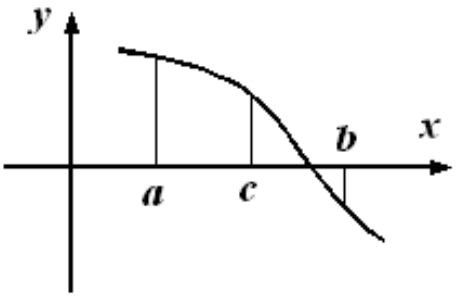


Рис. 2.9

Ниже приводится схема алгоритма уточнения одного корня уравнения $F(x) = 0$ на интервале $[a; b]$ до заданной точности e методом половинного деления (рис. 2.10) и пример программы на языке Си, реализующей этот алгоритм для функции $\sin x – 0{,}2x = 0$ на интервале [2,5; 2,6] с точностью до 10^{-6}.

```
Программа уточнения корней
#include<stdio.h>
#include<math.h>
#define y(x) sin(x)-0.2*x;
main()
{
float x1,x2,x,e,c,y1,y2,d;
printf(«Введите x1, x2, точность e\n»);
scanf(«%f%f%f»,&x1,&x2,&e);
do{
```

```
c=(x1+x2)/2;
y1=y(x1);
y2=y(x2);
if(y1*y2<0)x2=c;else x1=c;}
while(x2-x1>e);
x=(x1+x2)/2;
d=(x2-x1)/2;
printf(«x=%f d=%f\n»,x,d);
getch();
}
```

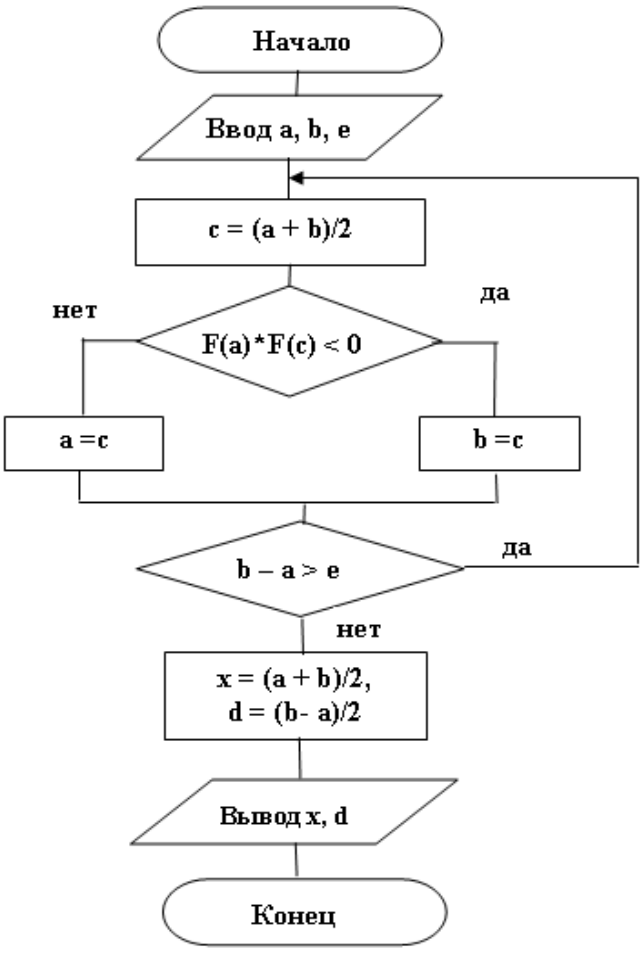


Рис. 2.10. Схема алгоритма уточнения корней

В программе заданная точность обозначается *e*, а граница погрешности найденного корня − *d*.

Выполнив программу для значений

A = 2,500002, *B* = 2,600002, *e* = 0,000001, получим ответ

x = 2,550002, *d* = 0,000001.

Округляя результаты, окончательно получаем:

x = 2,5500, *d* = 0,0000.

Для того чтобы использовать эту программу для уточнения корней другого уравнения, необходимо отредактировать программу, вписав выражение требуемой функции.

Контрольные вопросы

1. Какое уравнение называется трансцендентным?
2. Что значит решить трансцендентное уравнение?
3. Что называется корнем уравнения?
4. Объясните алгоритм отделения корней.
5. В чем заключается уточнение корней нелинейного уравнения?
6. Поясните метод половинного деления.
7. Как определяется погрешность вычисления корней уравнения при применении метода половинного деления?

2.2. Безусловная оптимизация функций. Минимум функции одной переменной

Рассмотрим задачу поиска минимума функции одной переменной $F(x)$ на интервале $[a; b]$, где расположено несколько локальных минимумов, среди которых необходимо определить глобальный (рис. 2.11).

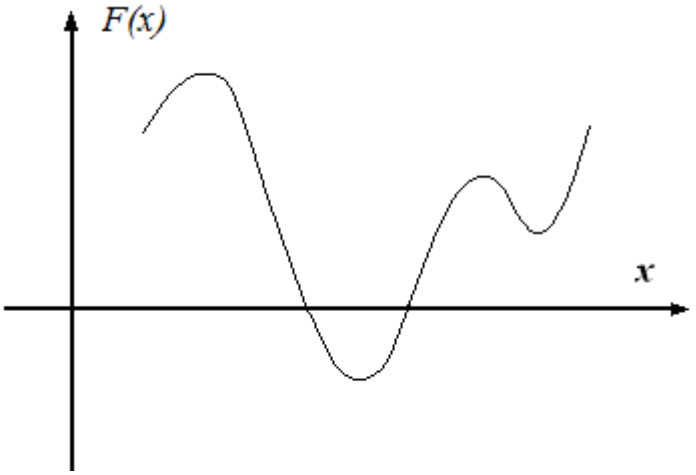


Рис. 2.11. Поиск минимума функции

Минимум дифференцируемой функции $F(x)$ определяется из уравнения

$$F'(x) = 0.$$

Известно, что корень этого уравнения является точкой минимума функции $F(x)$, причем мы имеем минимум, если $F''(x) > 0$ и, максимум, если $F''(x) < 0$.

Если невозможно получить аналитическую формулу для производной $F'(x)$, то ее значения могут быть определены дифференцированием полинома, аппроксимирующего функцию $F(x)$.

Когда функция $F(x)$ недифференцируема и вычисление ее значений для аппроксимации производной связано со значительными затратами времени, минимизацию осуществляют по алгоритмам, не связанным с решением уравнения $F'(x) = 0$. Рассмотрим один из методов, основанных именно на этих принципах, – *метод золотого сечения*.

Поиск минимумов (максимумов) функции $F(x)$ разделяется на два этапа, которые аналогичны задачам отделения и уточнения корней при решении нелинейной функции одной переменной.

2.2.1. Отделение унимодальных функций

На первом этапе выделяются интервалы аргумента x, в которых существует единственная точка x^*, где функция $F(x)$ принимает экстремальное значение. Функция на каждом таком интервале называется *унимодальной*.

Первый этап минимизации близок по идеологии к задаче отделения корней уравнений (п. 2.1) и не поддается строгой алгоритмизации. Применяются те же подходы, которые уже рассматривались при отделении корней нелинейного уравнения с одной переменной, т. е. это графическое представление функции либо анализ упрощенных математических моделей.

Ниже приводится пример программы на языке Си, реализующей этот алгоритм для функции: $y = 10\cos x - 0{,}1x^2$ на интервале [–10; 10] с шагом $h = 0{,}1$.

Программа отделения унимодальных функций:

```
#include<stdio.h>
#include<math.h>
main()
{
int k=0, l=0;
float A,B,h,x1,x2,x3,y1,y2,y3, xmax1=0, xmax2=0, xmin1=0, xmin2=0;
float min=0,max=0;
clrscr();
printf(«Введите A, B, h:\n»);
scanf(«%f%f%f»,&A,&B,&h);
x1=A; x2=x1+h; x3=x2+h;
y1=10*cos(x1)-0.1*pow(x1,2);
```

```
y2=10*cos(x2)-0.1*pow(x2,2);
while(x3<=B)
{
y3=10*cos(x3)-0.1*pow(x3,2);
if((y2<y1)&&(y2<y3))
{
if(y2<min){min=y2;xmin1=x1;xmin2=x3;}
k=k+1;
printf(«\n%10d_й min x1=%-10.4f x2=%-10.4f Ymin=%-10.4f\n»,k,
x1,x3,y2);
}
if((y2>y1)&&(y2>y3))
{
if(y2>max){max=y2; xmax1=x1;xmax2=x3;}
l=l+1;
printf(«\n%10d_й max x1=%-10.4f x2=%-10.4f Ymax=%-
10.4f\n»,l,x1,x3,y2);}
x1=x2; x2=x1+h; x3=x2+h; y1=y2; y2=y3;
}
printf(«\nГлобальный Ymin=%-10.4f xmin1=%-10.4f xmin2=%-
0.4f\n», min, xmin1, xmin2);
printf(«\nГлобальный Ymax=%-10.4f xmax1=%-10.4f xmax2=%-
4f\n», max, xmax1, xmax2);
getch();
}
```

Решение

Введите A, B, h:
-10 10 0,1
Результат:

1-й min	x1 = –9,7000	x2 = –9,5000	Ymin = –19,0629
1-й max	x1 = –6,3000	x2 = –6,1000	Ymax = 6,1214
2-й min	x1 = –3,3000	x2 = –3,1000	Ymin = –11,0069
2-й max	x1 = –0,1000	x2 = 0,1000	Ymax = 10,0000
3-й min	x1 = 3,1000	x2 = 3,3000	Ymin = –11,0069
3-й ma x	x1 = 6,1000	x2 = 6,3000	Ymax = 6,1214
4-й min	x1 = 9,5000	x2 = 9,7000	Ymin = –19,0629

Глобальный Ymin = –19,0629 xmin1=9,5000 xmin2 = 9,7000
Глобальный Ymax = 10,0000 xmax1= –0,10 xmax2= 0,1000

2.2.2. Метод золотого сечения

На втором этапе осуществляется уточнение расположения минимумов (максимумов) на интервале унимодальности функции.

Пусть функция $F(x)$ унимодальна на интервале $[a; b]$, необходимо построить такую последовательность $\{x_k\}$, чтобы минимум функции $F(x)$ находился в интервале неопределенности $[x_{i-1}, x_i]$, то есть $x_{i-1} < x^* > x_i$. Алгоритм выбора элементов последовательности называют стратегией поиска $\{x_k\}$.

При заданном количестве вычислений функции $F(x)$ оптимальной является стратегия, которая приводит к наименьшему интервалу неопределенности. Установлено, что стратегия поиска минимума будет оптимальной, если для построения последовательности $\{x_k\}$ использовать числа Фибоначчи F_k:

$$F_{k+1} = F_{k-1} + F_k.$$

На каждом шаге метода Фибоначчи интервал $[a; b]$ делится в точке x_1 в отношении двух последовательных чисел Фибоначчи.

Чтобы воспользоваться свойствами унимодальности функции для уменьшения интервала неопределенности, на интервале $[a; b]$ выбирается другая точка x_2, симметрично расположенная относительно середины интервала по отношению к первой точке.

Если $x_1 < x_2$ и $F(x_1) > F(x_2)$,

то на основании унимодальности функции $F(x)$ в качестве следующего интервала неопределенности следует выбрать отрезок $[x_1; b]$, что эквивалентно переносу точки a в точку x_1.

Если $x_1 < x_2$ и $F(x_1) < F(x_2)$,

то очередной интервал неопределенности будет $[a; x_2]$, что эквивалентно переносу точки b в точку x_2.

Недостатком метода Фибоначчи является зависимость точек, где проводятся вычисления функции $F(x)$ от общего числа вычислений N, так как на первой итерации точка x_1 выбирается из условия деления интервала $[a; b]$ в отношении F_{N-1}/F_N.

Для отношения чисел Фибоначчи установлен предел:

$$g = \lim_{N \to \infty} F_{N-1} / F_N = 0,618034...$$

Деление отрезка в таком отношении называют *золотым сечением*, так как в этом случае отношение длины большей части ко всей длине будет равно отношению меньшей части к длине большей части. Дей-

ствительно, принимая длину отрезка за единицу, из определения золотого сечения будем иметь:

$$g = \frac{1-g}{g}.$$

Последнее выражение приводится к квадратному уравнению

$$g^2 + g - 1 = 0,$$

положительный корень которого будет равен:

$$g = (-1 + \sqrt{5})/2 = 0{,}618034.$$

Можно на каждой итерации поиска минимума делить интервал неопределенности $[a; b]$ в одном и том же соотношении g (точка x_1), а другую точку x_2 выбирать симметрично точке x_1 относительно середины отрезка. При этом выбор очередного интервала неопределенности осуществляется на основании свойств унимодальности функции.

Такой метод называют методом *золотого сечения*. Здесь расположение точек деления x_1 и x_2 интервала $[a; b]$ не зависит от общего числа итераций N, и вычисления прекращаются, когда длина интервала неопределенности становится меньше заданной величины.

После N вычислений функции интервал неопределенности местоположения минимума становится равным $(b - a)g^{N-1}$, что позволяет оценить число итераций, необходимое для выполнения алгоритма с заданной погрешностью.

Для использования метода золотого сечения ниже приводится программа на языке Си. Для того чтобы воспользоваться этой программой, необходимо вместо приведенной функции вписать выражение требуемой функции.

Чтобы перейти к максимуму функции $F(x)$, достаточно изменить знак функции на противоположный и использовать тот же алгоритм, что и при отыскании минимума функции.

Ниже приведена программа уточнения минимума и максимума функции: $y = 10\cos x - 0{,}1x^2$.

Метод золотого сечения:

```
#include<stdio.h>
#include<math.h>
#define c 10*cos(x)-0.1*pow(x,2);
#define k 0.618*a+0.382*b;
#define l 0.382*a+0.618*b;
main()
```

```
{
float a,b,x,e,f,f1,f2,y,z;
clrscr();
puts («Введите a, b, e);
scanf(«%f%f%f»,&a,&b,&e);
y=k; z=l;
 x=k; f1=c;
 x=l; f2=c;
while(b-a>e)
{
if(f1<=f2)
{
b=z; z=y; f2=f1; y=k; x=y; f1=c;
}
 else {a=y; y=z; f1=f2; z=l; x=z; f2=c;}
}
x=(a+b)/2;
f=(f1+f2)/2;
printf(«\nXmin = %f    Ymin = %f\n»,x,f);
}
```

Решение

Введите *a*, *b*, *e*:

9,5000 9,7000 0,000001

Результат:

*X*min = 9,616716 *Y*min = –19,064487.

Индивидуальные задания

Тема: Исследование нелинейного уравнения с одной переменной.

Нелинейные уравнения (функции) приведены в табл. 2.1.

Исследование заключается в нахождении корней нелинейного уравнения, определения экстремальных значений функции (F_{max} и F_{min}) и соответствующих им значений аргументов.

Исследование необходимо выполнять в следующей последовательности:

1. Построить график функции на экране дисплея в интервале, указанном в таблице. Если интервал не указан, то исследовать функцию в пределах $-10 < x < 10$ или выбрать пределы изменения x самостоятельно.

Таблица 2.1

Нелинейные уравнения (*функции*)

Номер варианта	Уравнение	Примечания
1	$(0,2x)^3 = \cos x$	
2	$x - 10\sin x = 0$	
3	$2^{-x} = \sin x$	при $x < 10$
4	$2^x - 2\cos x = 0$	при $x > 10$
5	$\lg(x + 5) = \cos x$	при $x < 5$
6	$\sqrt{4x + 7} = 3\cos x$	
7	$x\sin x - 1 = 0$	
8	$8\cos x - x = 6$	
9	$\sin x - 0,2x = 0$	
10	$10\cos x - 0,1x^2 = 0$	
11	$2\lg(x + 7) - 5\sin x = 0$	
12	$4\cos x + 0,3x = 0$	
13	$5\sin 2x = \sqrt{1 - x}$	
14	$1,2x^4 + 2x^3 - 4,1 = 13x^2 + 14,2x$	
15	$2x^2 - 5 = 2^x$	
16	$2^{-x} = 10 - 0,5x^2$	
17	$4x^4 - 6,2 = \cos 0,6x$	
18	$3\sin 8x = 0,7x - 0,9$	на отрезке $[-1; 1]$
19	$1,2 - \ln x = 4\cos 2x$	
20	$\ln(x + 6,1) = 2\sin(x - 1,4)$	

2. Выполнить отделение корней заданного уравнения.

3. Вычислить корни заданного уравнения с использованием метода половинного деления. Вычисление произвести с точностью до 10^{-6}.

4. Выполнить отделение унимодальных функций.

5. Уточнить значение функции и аргумента в экстремальных точках заданной функции с помощью метода золотого сечения

Контрольные вопросы

1. Какую функцию называют унимодальной?
2. На какие этапы разделяется поиск минимума функции $F(x)$?
3. Каковы особенности метода золотого сечения?

2.3. Решение нелинейного уравнения со многими переменными

2.3.1. Метод координатного спуска

Рассмотрим алгоритм поиска минимума многомерной функции на примере функции двух переменных $f(z_1, z_2)$.

График функции $f(z_1, z_2)$ в области ее минимума представим в параметрическом виде, подобно изображению рельефа местности на географических картах, соединяя линиями точки на координатной плоскости $f(z_1, z_2)$, где функция принимает одинаковые значения (рис. 2.12).

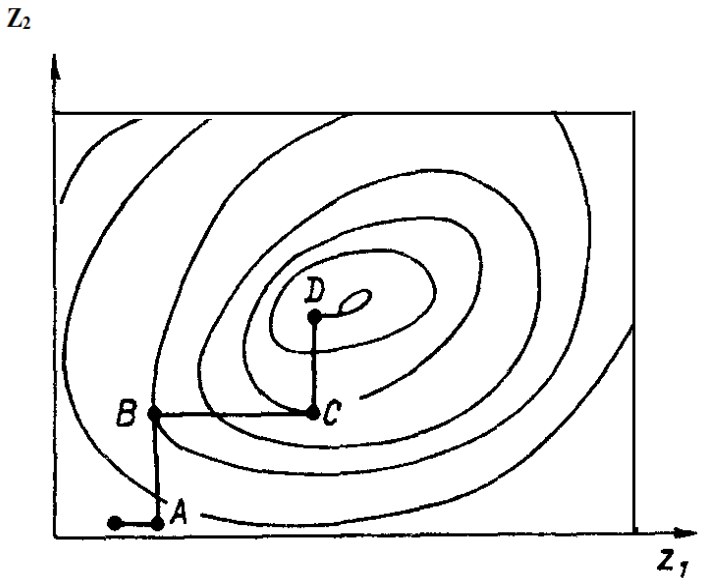


Рис. 2.12. Метод координатного спуска

Предположим, что нам известна прямоугольная область на плоскости (z_1, z_2), где находится минимум функции $f(z_1, z_2)$, т. е.

$$z_1 \in \left[z_{1^0}, z_{1^n} \right], \quad z_2 \in \left[z_{2^0}, z_{2^n} \right].$$

Алгоритм координатного спуска заключается в сведении многомерной задачи к последовательным одномерным задачам, которые решаются методами минимизации функции одной переменной, в частности методом *золотого сечения.*

Вначале в прямоугольной области (рис. 2.12) зафиксируем координату $z_2 = z_2^0$, тогда функция $f(z_2, z_2^0)$ будет зависеть только от одной переменной z_1. Найдем минимум z_1^0 функции $f(z_1, z_2^0)$, изменяя координату z_1 по методу *золотого сечения*. На рис. 2.12 найденный минимум располагается в точке $A(z_1, z_2^0)$. Затем зафиксируем первый аргумент $z_1 = z_1^1$

и найдем минимум z_2^1 функции f относительно второго аргумента z_2 (точка $B(z_1^1, z_2^1)$). Аналогичным способом перейдем последовательно к точкам $C(z_1^2, z_2^1)$, $D(z_1^2, z_2^2)$ и т. д.

Если в области минимума функция $f(z_1, z_2)$ достаточно гладкая, то процесс спуска по координатам будет линейно сходиться к минимуму. В сходящемся процессе с приближением к минимуму функции $f(z_1, z_2)$ расстояния между последовательными точками однокоординатных минимумов будут стремиться к нулю.

Поэтому в качестве критериев окончания итерационного процесса координатного спуска выбираются условия:

$$\left|z_1^k - z_1^{k-1}\right| < \varepsilon_1, \qquad \left|z_2^k - z_2^{k-1}\right| < \varepsilon_2,$$

где ε_1 и ε_2 – заданные допустимые абсолютные погрешности определения местоположения минимума по первой и второй координатам.

Метод координатного спуска легко обобщается на случай функций, имеющих размерность больше двух. Однако следует иметь в виду, что с ростом размерности значительно увеличивается объем вычислений.

В качестве примера находится минимум функции двух переменных.

$$f = 2(x_1 x_2 + 1/x_1 + 1/x_2).$$

Ниже приведены схема алгоритма (рис. 2.13) и программа, составленная на языке Си.

Метод координатного спуска:

```
#include<stdio.h>
#include<math.h>
float fun(float x);
float z[8],z0[8],z1[8],z9[8],f3[8];
float e,e1,a,b,x,x1,x2,y,v,f,f1,f2;
int i,n;
main()
{
clrscr();
puts(«Введите количество координат n и точность вычисления e и
e1»);
scanf(«%d%f%f»,&n,&e,&e1);
puts(«Введите пределы изменения аргументов z0,z9, нач. значение
z1»);
```

```
for (i=1;i<=n;i++)
scanf(«%f%f%f»,&z0[i],&z9[i],&z1[i]);
for (i=1;i<=n;i++)
z[i]=z1[i];
do
{
for(i=1;i<=n;i++)
{a=z0[i]; b=z9[i];
y=0.618*a+0.382*b;
v=0.382*a+0.618*b;
x=y; f1=fun(x);
x=v; f2=fun(x);
while(fabs(b-a>e))
{
if (f1<f2){b=v; v=y; f2=f1; y=0.618*a+0.382*b; x=y; f1=fun(x);}
else {a=y; y=v; f1=f2; v=0.382*a+0.618*b; x=v; f2=fun(x);}
}
f3[i]= f ;
}
}
while (fabs (f3[1]-f3[2]>e1));
for (i=1;i<=n;i++)
printf(«z(%d) = %f \n»,i,z[i]);
printf(«F=%f \n»,f3[i]);
getch ();
}
float fun(float x)
{ z[i]=x;
f=2.0*(z[1]*z[2]+1.0/z[1]+1.0/z[2]);
return(f); }
```

Решение

Введите количество координат *n* и точность вычисления *e* и *e*1
2
0.000001
0.000001
Введите пределы изменения аргументов z0, z9, начальное значение z1
0 3.0 0.5
0 3.0 0.5

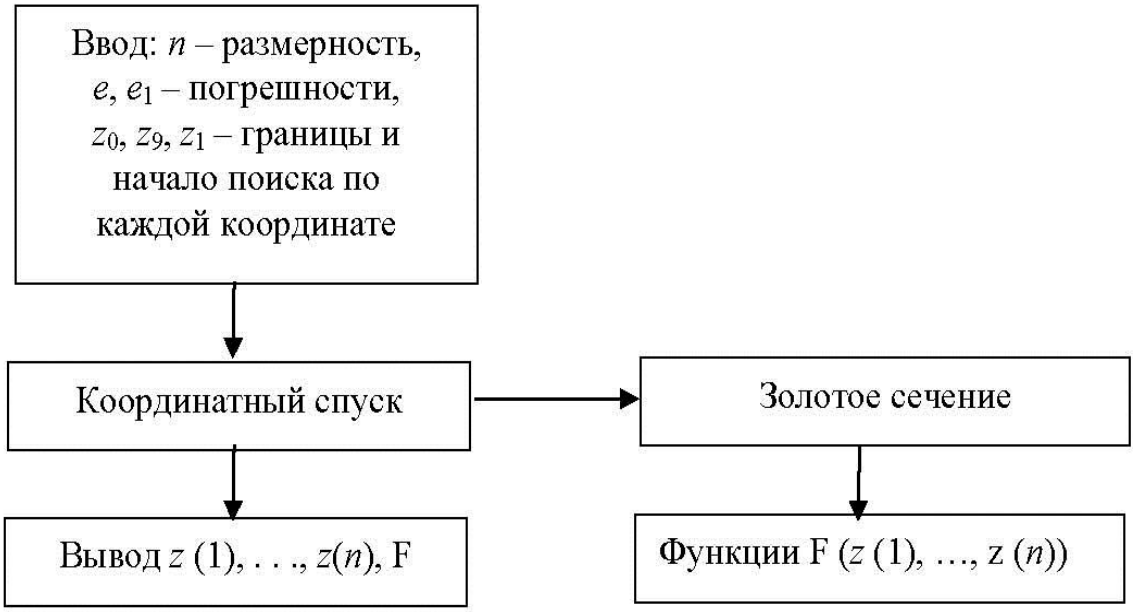


Рис. 2.13. Схема алгоритма метода координатного спуска

Результат:

$z(1) = 1.000321$
$z(2) = 1.000046$
$F\min = 6.000000$

2.3.2. Метод градиентного спуска

Идея метода заключается в следующем. Выбираем начальную точку и вычисляем в ней градиент рассматриваемой функции. Делаем шаг в направлении обратном градиентному. В новой точке снова определяем градиент и двигаемся в новом направлении на определенный шаг. Процесс продолжается до тех пор, пока движение из полученной точки не приводит к возрастанию функции.

Модификацией *градиентного* метода является метод *наискорейшего спуска*. Согласно этому методу движение в направлении обратном градиентному осуществляется до тех пор, пока функция убывает. Затем вычисляется градиент и снова определяется направление спуска.

С помощью *градиентного спуска* минимум гладких функций находится значительно быстрее, чем при использовании координатного спуска. Однако, наряду с вычислением функции *f*, на каждой итерации градиентного метода приходится вычислять составляющие градиента этой функции.

Кроме того, сходимость итерационного процесса может быть медленной для функций, имеющих овражный рельеф. В этом случае изменением масштабов переменных рекомендуется перейти к котлованному рельефу или применить так называемый овражный метод.

Контрольные вопросы

1. Объясните, в чем суть методов:
   а) координатного спуска;
   б) градиентного спуска;
   в) наискорейшего спуска.
2. Сформулируйте математическую модель безусловной оптимизации.

ГЛАВА 3. МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Многие задачи в области проектирования, управления и экономики сводятся к выбору параметров и характеристик, оптимизирующих некоторый показатель качества и одновременно удовлетворяющих определенным условиям (ограничениям). Решением таких задач занимается раздел математики, получивший название *математическое программирование.*

Следует отметить, что математическое программирование представляет собой не аналитическую, а алгоритмическую форму решения задачи, т. е. дает не формулу, выражающую окончательный результат, а указывает лишь вычислительную процедуру, которая приводит к решению задачи.

Математическое программирование включает такие разделы, как *линейное программирование*, *нелинейное программирование*, *динамическое программирование*.

Простейшим случаем задачи математического программирования является задача линейного программирования (ЛП).

3.1. Линейное программирование

Под линейным программированием понимается раздел теории экстремальных задач, в котором изучаются методы нахождения минимума или максимума линейной функции конечного числа переменных при условии, что переменные удовлетворяют конечному числу ограничений, имеющих вид линейных уравнений или линейных неравенств.

К таким задачам, например, относятся задачи нахождения наиболее рационального способа использования сырья и материалов, определения эффективных режимов работы производственного оборудования, повышения пропускной способности транспортных маршрутов и т. п.

3.1.1. Математическая постановка основной задачи линейного программирования (ОЗЛП)

Задача линейного программирования в общем случае формулируется следующим образом.

Найти такие неотрицательные значения переменных x_1, x_2, ..., x_n, при которых линейная функция этих переменных

$$L(x) = c_1x_1 + c_2x_2 + ... + c_nx_n \to \min \qquad (3.1)$$

обращалась бы в минимум при ограничениях:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1,$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m. \qquad (3.2)$$

$$x_1 \geq 0,\ x_2 \geq 0, \ldots,\ x_n \geq 0. \qquad (3.3)$$

Уравнения (3.2) называют системой ограничений данной задачи. Функцию $L(x)$ (3.1) − целевой функцией.

Следует заметить, что систему ограничений в виде неравенств всегда можно свести к системе в виде равенств (способом введения фиктивных добавочных неизвестных). Так, если имеется

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1, \qquad (3.4)$$

то, вводя неизвестное $x_{n+1} \geq 0$, получим:

$$a_{11}x_1 + \ldots + a_{1n}x_n - x_{n+1} = b_1. \qquad (3.5)$$

Значением x_{n+1} в окончательном результате можно пренебречь.

Кроме того, так как min = –max (–f), то любая задача на максимизацию сводится к задаче минимизации и наоборот.

Сделаем несколько пояснений по поводу задачи ЛП.

Характерной особенностью данной задачи является то, что число уравнений меньше числа неизвестных, т. е. ($m < n$). Система уравнений (3.2) для случая, когда число уравнений равно числу неизвестных ($m = n$), рассматривается в обычной алгебре.

Если при этом определитель системы не равен нулю, то система имеет одно-единственное решение.

Если же число уравнений меньше числа неизвестных ($m < n$), то система уравнений (3.2) имеет бесчисленное множество решений, т. е. имеется бесчисленное множество наборов переменных $x_i = \overline{1,n}$, которые удовлетворяют уравнениям (3.2).

Каждый набор переменных x_i, удовлетворяющий системе уравнений (3.2), будем называть *решением.*

Однако на переменные x_i наложено добавочное ограничение, состоящее в том, что эти переменные должны быть неотрицательны ($x_i \geq 0$). В общем случае имеется бесчисленное множество решений,

удовлетворяющих этому добавочному условию. Любое решение системы (3.2) с неотрицательными значениями переменных будем называть *допустимым решением.*

Таким образом, суть задачи линейного программирования состоит в том, чтобы из множества допустимых решений выбрать одно, а именно такое, которое обращает в минимум линейную форму (3.1).

Прежде чем искать решение уравнений (3.2), удовлетворяющее наложенным требованиям:

$$x_i \geq 0, \qquad i = \overline{1,n}, \quad L(x) = \min,$$

попробуем найти какое-либо решение уравнений (3.2).

Поскольку число переменных n в этой системе больше числа уравнений m, то одно из возможных решений можно найти, если $n - m$ каких-либо переменных положить равными нулю. Полученную при этом систему m уравнений с m неизвестными можно решать обычными методами алгебры. При этом, для того чтобы система m уравнений с m неизвестными имела решение, необходимо, чтобы определитель, составленный из коэффициентов при неизвестных, не обращался в нуль. Если это условие не выполняется, то можно приравнять нулю другие $n - m$ переменных. Полученное при этом решение называют *базисным.*

Можем ввести теперь некоторую терминологию, широко употребляемую в задачах линейного программирования.

Базисом называют любой набор m переменных таких, что определитель, составленный из коэффициентов, при этих переменных не равен нулю. Эти m переменных называют *базисными* (по отношению к данному базису). Остальные $n - m$ переменных называются *небазисными* или *свободными.* В каждой конкретной системе уравнений (3.2) может существовать несколько различных базисов с различными базисными переменными.

Если положить все свободные переменные равными нулю и решить полученную систему m уравнений с m неизвестными, то получим *базисное* решение.

Но необходимо отметить, что среди различных базисных решений будут такие, которые дают отрицательные значения некоторых переменных. Эти базисные решения противоречат условию задачи и являются *недопустимыми.*

Допустимым базисным решением является такое базисное решение, которое дает неотрицательные значения базисных переменных. Данные базисные решения являются наиболее простыми из допустимых решений системы (3.2).

Однако на решение задачи накладывается добавочное условие: линейная форма (3.1) должна при найденном решении принимать минимальное значение. Решение задачи при этом добавочном условии усложняется, но понятие допустимого базисного решения играет очень важную роль и при нахождении полного решения задачи.

3.1.2. Примеры задач линейного программирования

Решение задач линейного программирования поясним на ряде примеров.

Пример 1. Задача об использовании ресурсов.

Для осуществления n различных технологических процессов T_1, ..., T_n заводу требуется m видов ресурсов S_i, ..., S_m (сырье, топливо, материалы, инструмент и т. п.). Запасы ресурсов каждого вида ограничены и равны b_1, ..., b_m. Известен расход ресурсов на единицу продукции по каждому технологическому процессу.

Требуется определить, в каком количестве выпускать продукцию каждого вида, чтобы доход от реализации этой продукции был максимальным.

Все имеющиеся данные представим в виде табл. 3.1, положив для конкретности $n = 3$, $m = 4$.

Таблица 3.1

Виды ресурсов	Расход ресурсов на единицу продукции			Запасы ресурсов
	T_1	T_2	T_3	
S_1	a_{11}	a_{12}	a_{13}	b_1
S_2	a_{21}	a_{22}	a_{23}	b_2
S_3	a_{31}	a_{32}	a_{33}	b_3
S_4	a_{41}	a_{42}	a_{43}	b_4
Доход от реализации продукции	c_1	c_2	c_3	c_4

Обозначим:

a_{ij} – расход ресурсов вида S_i на единицу продукции вида T_j;

c_j – доход от реализации единицы продукции вида T_j;

x_j – количество единиц выпускаемой продукции вида T_j.

Ограничениями в этой задаче являются требования: расход ресурсов вида S_i на выпуск всех видов продукции не должен превышать объема имеющихся запасов:

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \ \ i = \overline{1, m}. \qquad (3.6)$$

Эти ограничения легко превратить в уравнения, введя переменные $x_{i+1} \ge 0$, означающие неиспользованные ресурсы вида S_i. При этом вместо (3.6) получим

$$\sum_{j=1}^{n} a_{ij} x_j + x_{i+1} = b_i, \quad i = \overline{1, m}. \qquad (3.7)$$

Доход от реализации выпущенной продукции

$$Q = \sum_{j=1}^{n} c_j x_j . \qquad (3.8)$$

Оптимальным планом выпуска продукции будет такое неотрицательное решение системы уравнений (3.7), при котором целевая функция (3.8) будет максимальна.

Пример 2. Задача о распределении выпуска продукции по предприятиям.

План отрасли предусматривает за время T выпуск следующих видов продукции:

A_1 – в количестве N_1 штук;
A_2 – в количестве N_2 штук;
. . . . . .
A_i – в количестве N_i штук.

Эти виды продукции могут выпускаться на r однородных предприятиях $П_1$, ..., $П_r$. Предполагаем, что ни одно предприятие не может одновременно выпускать несколько видов продукции. Кроме того, задано:

a_{ij} – количество продукции A_i, выпускаемой на предприятии $П_j$, в единицу времени;

b_{ij} – стоимость единицы продукции вида A_i, выпущенной на предприятии $П_j$;

x_{ij} – время работы предприятия $П_j$ по выпуску продукции A_i.

Требуется найти такие значения x_{ij}, при которых стоимость выпускаемой продукции будет минимальной.

Ограничения:

1) время работы каждого предприятия не должно превышать T:

$$\sum_{i=1}^{k} x_{ij} \le T, \ \ j = \overline{1, r}; \qquad (3.9)$$

2) количество выпускаемой продукции должно соответствовать номенклатуре:

$$\sum_{j=1}^{r} a_{ij} x_{ij} = N_i, \ i = \overline{1,k}. \tag{3.10}$$

Целевая функция будет представлять собой общую стоимость выпущенной продукции.

Если принять во внимание, что величина $a_{ij}b_{ij}x_{ij}$ представляет собой стоимость части продукции A_i, выпускаемой предприятием $П_j$, то общая стоимость выпускаемой продукции

$$q = \sum_{J=1}^{r} \sum_{i=1}^{k} a_{ij} b_{ij} x_{ij}. \tag{3.11}$$

Согласно условиям задачи эта величина должна быть минимизирована при выполнении ограничений (3.9) и (3.10).

3.1.3. Решение задачи линейного программирования. Симплекс-метод

Исходя из формулировки задачи линейного программирования, запишем систему ограничений и целевую функцию в виде:

$$x_1 = b_1 - (a_{1,r+1} x_{r+1} + \ldots + a_{1n} x_n),$$

$$x_2 = b_2 - (a_{2,r+1} x_{r+1} + \ldots + a_{2n} x_n),$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$x_r = b_r - (a_{r,\,r+1} x_{r\,+\,1} + \ldots + a_{1r} x_n), \tag{3.12}$$

$$f = c_0 - (c_{r+1} x_{r+1} + \ldots + c_n x_n). \tag{3.13}$$

При этом свободные члены $b_1, b_2, \ldots, b_r$ неотрицательны, базис $Б = \{x_1, x_2, \ldots, x_r\}$, а соответствующее ему базисное решение есть

$$(b_1, b_2, \ldots, b_r, 0, 0, \ldots, 0), \tag{3.14}$$

для которого $f_Б = c_0$.

Как следует из (3.13), уменьшить значение f – значит путем увеличения одного из свободных неизвестных добиться того, чтобы значение выражения в скобках стало положительным (разумеется, при соблюдении условия неотрицательности для базисных неизвестных).

В зависимости от знаков коэффициентов $c_{r\,+\,1}, c_{r\,+\,2}, \ldots, c_n$ здесь могут быть два случая.

1. Все числа c_{r+1}, c_{r+2}, ..., c_n неположительны. Тогда значение f уже более не может быть уменьшено, ибо это могло быть сделано только путем уменьшения значений свободных неизвестных, что невозможно.

Отсюда следует, что базисное решение (3.14) в этом случае является *оптимальным*.

2. Среди чисел c_{r+1}, c_{r+2}, ..., c_n имеются положительные. Пусть, например,

$$c_j > 0, (r + 1 \le j \le n).$$

Это позволяет уменьшить значение f путем увеличения x_j, оставляя значения других свободных неизвестных нулевыми. Считая, что значения всех неизвестных x_{r+1}, x_{r+2}, ..., x_n, кроме x_j, равны нулю, на основании (3.12) и (3.13) имеем:

$$x_1 = b_1 - a_{1j} x_j,$$

$$x_2 = b_2 - a_{2j} x_j,$$

. . . . . .

$$xr = br - arj\ xj, \qquad (3.15)$$

$$f = c_0 - c_j x_j,\ c_j > 0. \qquad (3.16)$$

Из равенств (3.15) следует, однако, что, увеличивая значение x_j, необходимо следить за сохранением условия неотрицательности базисных переменных x_1, x_2, ..., x_r. Легко видеть, что при неотрицательности свободных членов b_1 b_2, ..., b_r последнее полностью зависит от знака коэффициентов при x_j. Здесь, в свою очередь, снова различаются два случая (2*а* и 2*б*):

2*а*) все числа a_{1j}, a_{2j}, ..., a_{rj} неположительны. Тогда значение x_j может быть увеличено неограниченно, что приводит к неограниченному уменьшению f. Таким образом, в этом случае минимум f не достигается, т. е. $\min f = \infty$.

2*б*) среди чисел a_{1j}, a_{2j}, ..., a_{rj} имеются положительные. Пусть, например, $a_{kj} > 0$ $(1 \le k \le r)$. Коэффициентов a_{kj}, удовлетворяющих этому условию, может быть несколько. Найдем для них значения частных вида $\frac{b_k}{a_{kj}}$ и выберем среди этих частных наименьшее − пусть это будет

$$\frac{b_i}{a_{ij}} = k \ge 0.$$

Для сохранения неотрицательности всех базисных неизвестных $x_1, x_2, ..., x_r$ значение x_j может быть увеличено не более чем до k. Коэффициент a_{ij} в этих условиях называют *разрешающим элементом.*

Итак, примем $x_j = k$ и найдем значения базисных неизвестных (при условии, что все небазисные неизвестные, кроме x_j, равны нулю):

$$x_1 = b_1 - a_{1j}\, k,$$

. . . . . .

$$x_i = b_i - a_{ij}k = 0,$$

. . . . . .

$$x_r = b_r - a_{rj}\, k.$$

Неизвестное x_i теперь должно перейти в состав свободных неизвестных, а взамен него в базис следует ввести неизвестное x_j. Новый базис будет иметь вид

$$\text{Б}_1 = \{x_1, x_2, ..., x_{i-1}, x_i, x_{i+1}, ..., x_r\}\,.$$

Значение целевой функции f, соответствующее базису Б_1, равно:

$$f_{\text{Б}1} = c_0 - c_j\, k \le c_0 = f_{\text{Б}}, \text{ т. е. } f_{\text{Б}1} \le f_{\text{Б}}.$$

Последнее означает, что в результате выполнения одного шага процесса (для случая 2*б*) происходит уменьшение (или, по крайней мере, неувеличение) значения целевой функции.

Для перехода к следующему шагу симплекс-метода необходимо преобразовать систему ограничений (3.12) и целевую функцию (3.13) применительно к новому базису. С этой целью из уравнения системы (3.12), отвечающего бывшему базисному неизвестному x_i, выражается новое базисное неизвестное x_j, которое вслед за этим исключается из остальных уравнений системы (подстановкой в эти уравнения полученного выражения для x_i). Точно так же неизвестное x_j исключается из выражения (3.13) для функции f.

Вслед за этим весь рассмотренный этап повторяется сначала. Очевидно, что остановка процесса может произойти в случаях 1 или 2*а*.

3.1.4. Симплекс-таблицы

Ручные вычисления по симплекс-методу удобно оформлять в виде так называемых симплекс-таблиц. Для удобства составления симплекс-таблицы будем считать, что система ограничений (3.12) переписана в виде:

$$x_1 + a_{1,r+1}\,x_{r+1} + \ldots + a_{1n}x_n = b_1\,,$$

$$x_2 + a_{2,r+1}\,x_{r+1} + \ldots + a_{2n}\,x_n = b_2\,,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$x_r + a_{r,r+1}\,x_{r+1} + \ldots + a_{rn}\,x_n = b_r \qquad (3.17)$$

и целевая функция (3.13) определена равенством

$$f + c_{r+1}\,x_{r+1} + \ldots + c_n x_n = c_0. \qquad (3.18)$$

По исходным данным (3.17), (3.18) заполняется начальная симплекс-таблица, форма которой показана в табл. 3.2.

Сформулируем алгоритм симплекс-метода применительно к данным, внесенным в табл. 3.2.

1.Выяснить, имеются ли в последней строке таблицы положительные числа (c_0 не принимается во внимание). Если все числа отрицательны, то процесс закончен; базисное решение (b_1, b_2, ..., b_r, 0, 0, ..., 0) является оптимальным; соответствующее значение целевой функции $f = c_0$. Если в последней строке имеются положительные числа, перейти к п. 2.

2.Просмотреть столбец, соответствующий положительному числу из последней строки, и выяснить, имеются ли в нем положительные числа. Если ни в одном из таких столбцов нет положительных чисел, то оптимального решения не существует.

Таблица 3.2

Представление исходных данных в виде симплекс-таблицы

Базисные неизвестные	Свободные члены	x_1 ... x_i ... x_r	x_{r+1} ... x_j ... x_n
x_1	b_1	1 ... 0 ... 0	$a_{1,r+1}$ ... a_{1j} ... a_{1n}
..	..	... ...	... ...
x_i	b_i	0 ... 1 ... 0	$a_{i,r+1}$ ... a_{ij} ... a_{in}
..	..	... ...	... ...
x_r	b_r	0 ... 0 ... 1	$a_{r,r+1}$ ... a_{ij} ... a_{rn}
Форма f	c_0	0 ... 0 ... 0	c_{r+1} ... c_j ... c_n

Если найден столбец, содержащий хотя бы один положительный элемент (если таких столбцов несколько, взять любой из них), отметить этот столбец вертикальной стрелкой (см. табл. 3.2) и перейти к п. 3.

3.Разделить свободные члены на соответствующие положительные числа из выделенного столбца и выбрать наименьшее частное. Отметить строку таблицы, соответствующую наименьшему частному, горизонтальной стрелкой. Выделить разрешающий элемент a_{ij}, стоящий на пересечении отмеченных строки и столбца. Перейти к п. 4.

Комментарий. Задача теперь состоит в том, чтобы удалить из базиса неизвестное, расположенное против разрешающего элемента в строке (x_i), и ввести вместо него свободное неизвестное, расположенное напротив разрешающего элемента в столбце (x_j). В соответствии с алгоритмом симплекс-метода это означает преобразование системы ограничений и целевой функции.

В данном случае это означает переход к новой симплекс-таблице, в первом столбце которой вместо x_i вписывается обозначение x_j, а заполнение внутренних пустых клеток происходит по правилам, изложенным ниже.

4.Разделить элементы выделенной строки исходной таблицы на разрешающий элемент (на месте разрешающего элемента появится единица). Полученная таким образом новая строка пишется на месте прежней в новой таблице. Перейти к п. 5.

5.Каждая следующая строка новой таблицы образуется сложением соответствующей строки исходной таблицы и строки, записанной в п. 4, которая предварительно умножается на такое число, чтобы в клетках выделенного столбца при сложении появились нули. На этом заполнение новой таблицы заканчивается и происходит переход к п. 1.

Пример

Рассмотрим решение задачи линейного программирования с помощью симплекс-таблиц.

Дана система ограничений:

$$x_1 - x_4 + x_5 = 2,$$

$$x_2 + 2x_4 + 3x_5 = 7,$$

$$x_3 + x_4 - 2x_5 = 1$$

и целевая функция

$$f - x_4 + 2x_5 = 3.$$

Для данного примера заполним симплекс-таблицу (табл. 3.3).

Таблица 3.3

Базисные неизвестные	Свободные члены	x_1	x_2	x_3	x_4	x_5
x_1	2	1	0	0	–1	[1]
x_2	7	0	1	0	2	3
x_3	1	0	0	1	1	–2
Форма f	3	0	0	0	–1	2

Воспользуемся алгоритмом симплекс-метода. Замечаем, что в последней строке табл. 3.3 имеется положительный элемент, выбираем наименьшее частное из $\frac{2}{1}$ и $\frac{7}{3}$ и выделяем разрешающий элемент (на пересечении строки x_1 и столбца x_5). В новый базис вместо неизвестного x_1 войдет неизвестное x_5. Поскольку разрешающий элемент уже равен единице, выделенную строку без изменений переписываем на прежнее место в новую таблицу (табл. 3.4).

К табл. 3.4 снова применяем симплекс-алгоритм, начиная с п. 1.

Поскольку в последней строке есть положительный элемент (в столбце x_4), процесс не заканчивается. В результате преобразований получается табл. 3.5, в последней строке которой уже нет положительных элементов.

Это означает, что последнее базисное решение (0; 0; 5,2; 0,2; 2,2) является оптимальным.

Таблица 3.4

Базисные неизвестные	Свободные члены	x_1	x_2	x_3	x_4	x_5
x_5	2	1	0	0	–1	1
x_2	1	–3	1	0	[5]	0
x_3	5	2	0	1	–1	0
Форма f	–1	–2	0	0	1	0

Соответствующее значение целевой функции $f = –1,2$. Таким образом, задача решена.

Таблица 3.5

Оптимальное решение задачи

Базисные неизвестные	Свободные члены	x_1	x_2	x_3	x_4	x_5
x_5	2,2	0,4	0,2	0	0	1
x_4	0,2	–0,6	0,2	0	1	0
x_3	5,2	1,4	0,2	1	0	0
Форма f	–1,2	–1,4	–0,2	0	1	0

3.1.5. Отыскание исходного базиса

Начиная обсуждать алгоритм решения задачи линейного программирования симплекс-методом (см. п. 3.2), мы предполагали, что начальный базис системы ограничений уже известен. Укажем некоторые приемы нахождения базиса системы m линейных уравнений с n неизвестными ($m < n$).

В некоторых случаях базис усматривается непосредственно. Пусть дана система:

$$2x_1 + 1{,}4x_3 - x_4 = 6{,}8;$$
$$x_2 + 5x_3 - 7{,}1x_4 = 4{,}3;$$
$$8x_3 + 6{,}4x_4 - 4x_5 = -1{,}2.$$

Замечаем, что неизвестные x_1, x_2 и x_5 можно выразить только через неизвестные x_3 и x_4. С этой целью разделим первое уравнение на 2, а третье – на –4 и перепишем для наглядности исходную систему в виде:

$$x_1 = 3{,}4 - 0{,}7\,x_3 + 0{,}5x_4;$$
$$x_2 = 4{,}3 - 5x_3 + 7{,}1\,x_4;$$
$$x_5 = 0{,}3 + 2x_3 + 1{,}6x_4.$$

Неизвестные x_1, x_2 и x_5 образуют базис. При этом существенно заметить, что все свободные члены положительны, чем обеспечивается выполнение необходимого условия (3.5).

Другой метод получения начального базиса – *метод симплексного преобразования.* Поскольку в преобразовании участвуют только числа (коэффициенты и свободные члены), исходную систему (3.1) удобно записывать в виде табл. 3.6.

При этом предполагается, что все свободные члены $b_i (i = 1, 2, ..., m)$ – числа неотрицательные (этого легко добиться, умножая уравнения с отрицательными свободными членами на –1).

Алгоритм симплексного преобразования основан на идее последовательного исключения переменных методом, во многом схожим с методом преобразования симплекс-таблиц.

Таблица 3.6

Получение начального базиса методом симплексного преобразования

x_1	x_2		x_n	Свободные члены
a_{11}	a_{12}	…	a_{1n}	b_1
a_{21}	a_{22}	…	a_{2n}	b_2
…	…	…	…	…
a_{m1}	a_{m2}	…	a_{mn}	b_m

Среди столбцов из коэффициентов при неизвестных выбирается столбец, в котором имеется хотя бы один положительный элемент. Если в таком столбце несколько положительных элементов, то из них выбирается тот, который отвечает наименьшему числу при делении соответствующих свободных членов на положительные элементы выбранного столбца. Выделенный таким образом элемент называется *разрешающим* (см. п. 3 алгоритма симплекс-метода).

Элементы разрешающей строки делятся на разрешающий элемент и переписываются в новую таблицу. Остальные элементы таблицы получаются по правилу, описанному в п. 5 алгоритма симплекс-метода.

Процесс продолжается до тех пор, пока не будет получено неотрицательное базисное решение.

Пример

Найти неотрицательное базисное решение системы

$$x_1 - 5x_5 + 2x_6 = -3,$$
$$x_2 - 2x_5 - x_6 = 5,$$
$$x_3 - 3x_5 + x_6 = -7,$$
$$x_4 + x_5 - 3x_6 = -4.$$

Дальнейшие преобразования по нахождению начального базиса приведены в табл. 3.7.

Таблица 3.7

Нахождение начального базиса

Этапы преобразований	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены
1	1	0	–1	0	-2	1	4
	0	1	0	0	-2	-1	5
	0	0	–1	0	3	-1	7
	0	0	1	1	4	-1	3
2	1	0	–1	0	–2	1	4
	1	1	–1	0	–4	0	9
	1	0	–2	0	1	0	11
	4	0	–5	1	–4	0	19
3	3	0	–5	0	0	1	26
	5	1	–9	0	0	0	53
	1	0	–2	0	1	0	11
	8	0	–13	1	0	0	63

Замечаем, что неизвестные x_1, x_2, x_3, x_4 могут быть легко выражены через неизвестные x_5 и x_6 (т. е. исходную систему формально несложно переписать в виде (3.4)), однако при этом не будет выполнено условие неотрицательности свободных членов (3.5). Воспользуемся методом симплексного преобразования.

Перед составлением начальной таблицы вида 3.6, с целью получения неотрицательных свободных членов, третье уравнение умножим на (–1) и прибавим последовательно к первому и четвертому. Уже на третьем этапе преобразований получается одно из неотрицательных базисных решений (0; 53; 0; 63; 11; 26), что при необходимости позволяет составить первую симплекс-таблицу с начальным базисом Б = $\{x_2, x_4, x_5, x_6\}$.

Метод симплексного преобразования составляет фактически идейную основу и другого распространенного метода отыскания начального базиса, особенно удобного для последующего составления симплекс-таблиц, так называемого «метода искусственного базиса».

Индивидуальные задания

Решить один из приведенных ниже вариантов основной задачи линейного программирования. При этом сначала необходимо составить математическую модель в общем виде, а затем уточнить эту модель в соответствии с номером и числовыми данными индивидуального задания.

В табл. 3.8–3.10 приведены данные для трех типов задач.

1. Задача об использовании ресурсов.

Таблица 3.8

Показатели изготовления n видов продукции

Вариант	Виды ресурсов	Расход ресурсов на единицу продукции			Запасы ресурсов b_i	Доход от реализации единицы продукции		
		P_1	P_2	P_3		C_1	C_2	C_3
1	S_1	2	1	1	25	6	5	5
	S_2	1	1	1	14			
	S_3	0	4	2	19			
	S_4	3	0	1	24			
2	S_1	2	5	–	300	5	8	–
	S_2	4	5	–	400			
	S_3	3	0	–	100			
	S_4	0	4	–	200			
3	S_1	2	5	–	30	50	40	–
	S_2	8	5	–	40			
	S_3	5	6	–	30			
4	S_1	2	3	–	19	7	5	–
	S_2	2	1	–	13			
	S_3	0	3	–	15			
	S_4	3	0	–	18			

Окончание табл. 3.8

Вариант	Виды ресурсов	Расход ресурсов на единицу продукции			Запасы ресурсов b_i	Доход от реализации единицы продукции		
		P_1	P_2	P_3		C_1	C_2	C_3
5	S_1	4	2	1	15000	10	15	20
	S_2	6	0	2	17000			
	S_3	0	2	4	10000			
	S_4	8	7	0	20000			
6	S_1	14	12	9	250000	120	100	150
	S_2	16	0	11	170000			
	S_3	0	22	7	150000			
	S_4	10	17	0	240000			

Для изготовления n видов продукции P_1, ..., P_n предприятие использует m видов ресурсов S_1, ..., S_m (сырье, топливо, материалы, инструмент и т. п.). Запасы ресурсов каждого вида ограничены и равны b_1, ..., b_m. На изготовление единицы продукции j-го вида (j = 1, ..., m) расходуется a_{ij} единиц i-го ресурса (i = 1, ..., n). При реализации единицы j-й продукции предприятие получает C_j единиц прибыли.

Необходимо составить такой план выпуска продукции, чтобы при ее реализации получить максимальную прибыль.

2. Задача о загрузке оборудования.

Рассмотрим две постановки этой задачи.

1. Предприятие выпускает n видов изделий P_1, ..., P_n, каждое из которых проходит последовательно обработку на станках типов T_1, ..., T_m. Запас мощности станков, т. е. рабочее время станка, составляет, соответственно, b_1, ..., b_m единиц времени.

Изделие P_i обрабатывается первым станком (типа N_1) a_{i1} единиц времени, вторым станком − a_{i2} единиц времени и т. д. При реализации одно изделие P_i приносит предприятию C_i единиц прибыли (i = 1, ..., n).

Составить такой план загрузки станков, при котором предприятие получит максимальную прибыль.

Конкретные числовые данные приведены в табл. 3.9.

Таблица 3.9

Показатели изготовления n видов продукции

№	Виды ресурсов	Расход ресурсов на единицу продукции			Запасы ресурсов	Доход от реализации единицы продукции		
		P_1	P_2	P_3		C_1	C_2	C_3
1	T_1	2	1	1	25			
	T_2	1	1	1	14			
	T_3	0	4	2	19	6	5	5
	T_4	3	0	1	24			
2	T_1	2	5	–	300			
	T_2	4	5	–	400			
	T_3	3	0	–	100	5	8	–
	T_4	0	4	–	200			
3	T_1	2	5	–	20			
	T_2	8	5	–	40	50	40	–
	T_3	5	6	–	30			
4	T_1	2	3	–	19			
	T_2	2	1	–	13			
	T_3	0	3	–	15	7	5	–
	T_4	3	0	–	18			
5	T_1	4	2	1	150000			
	T_2	6	0	2	170000			
	T_3	0	2	4	100000	100	150	200
	T_4	8	7	0	200000			
6	T_1	7	1	5	290			
	T_2	3	4	8	320			
	T_3	5	0	2	400	8	10	15
	T_4	2	6	0	350			

2. Предприятию необходимо выпустить n видов изделий P_1, …, P_n в количествах, соответственно, N_1, …, N_n единиц. Для этой цели используются m типов станков T_1, …, T_m, каждый из которых может обрабатывать все изделия P_i, (i = 1, ..., n). Производительность каждого станка (количество изделий, обрабатываемых в единицу времени) имеет величину a_{ij}(i = 1, …, n; j = 1, …, m). Запас мощности станков (рабочее время станка) составляет, соответственно, b_1, ..., b_m единиц времени. Конкретные числовые данные приведены в табл. 3.10.

Таблица 3.10

Показатели изготовления n видов продукции

№	Типы станков	Производительность станков				Себестоимость продукции				План выпуска продукции				Запас мощности станков
		P_1	P_2	P_3	P_4	C_1	C_2	C_3	C_4	N_1	N_2	N_3	N_4	B_i
1	T_1	30	20	–	–	6	12	–	–					120
	T_2	1	1	1		8	10	–	–	4000	3000	–	–	100
	T_3	0	4	2		11	7	–	–					160
2	T_1	6	24	–	–	4	47	–	–					6
	T_2	13	13	–	–	13	26	–	–	30	96	–	–	6
3	T_1	30	50	30	20	2	1	0,5	1,2					240
	T_2	60	100	60	40	0,8	1,2	0,9	0,8	3	15	4,5	1,5	150
	T_3	18	30	18	12	0,5	1	0,6	0,9					150
4	T_1	8	4	2	–	4	6	3	–					60
	T_2	4	2	1	–	5	4	2	–	160	100	100	–	70
5	T_1	5	10	20	–	6	3	15	–					40
	T_2	1,7	3,3	5	–	6	3	2	–	300	500	100	–	60
	T_3	5	10	2,5	–	4	2	8	–					30

Требуется составить план загрузки станков, при котором себестоимость выпуска продукции будет минимальной.

Частным случаем задачи линейного программирования являются *задача транспортного типа* и *задача о назначениях*.

Рассмотрим особенности математических моделей этих задач.

3.2. Задача транспортного типа

Имеется m поставщиков и n потребителей однородной продукции, возможности и потребности которых соответственно равны a_i и b_j, где

$$i=\overline{1,m},\quad j=\overline{1,n}.$$

Стоимость перевозки одной единицы продукции из пункта i в пункт j равна c_{ij}. Определить план перевозки продукции от поставщиков к потребителям x_{ij}, минимизирующий общую стоимость всех перевозок.

3.2.1. Математическая постановка задачи

Целевая функция имеет вид

$$Z=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}x_{ij}\rightarrow \min, \tag{3.19}$$

при ограничениях:

$$\sum_{i=1}^{m}x_{ij}\geq b_j,\ j=\overline{1,n}, \tag{3.20}$$

$$\sum_{j=1}^{n}x_{ij}\leq a_i,\ i=\overline{1,m}, \tag{3.21}$$

$$x_{ij}\geq 0,\ i=\overline{1,m},\ j=\overline{1,n}. \tag{3.22}$$

Ограничение (3.20) накладывается на спрос j-го потребителя, ограничение (3.21) − на возможности i-го поставщика.

Если $$\sum_{i=1}^{m}a_i=\sum_{j=1}^{n}b_j,$$

то задача называется закрытой, в противном случае − открытой.

Пример

На станках C_1 и C_2 обрабатывается по 30 деталей типа D_1, D_2, D_3. Известно, что время обработки деталей на станке C_1 составляет, соответственно, 11, 6, 9 минут, а время обработки деталей на станке C_2 – 13, 10, 7 минут. Необходимо обработать по 20 деталей каждого типа.

Составить план обработки деталей с минимальной продолжительностью.

Исходные данные задачи сведены в табл. 3.11.

Введены следующие обозначения:

a_i – количество деталей, которое может быть обработано на каждом из станков;

b_j – план выпуска деталей каждого наименования.

В качестве переменной x_{ij} принимаем количество деталей i-го типа, обрабатываемых на станке j-го типа.

Таблица 3.11

Исходные данные задачи

b_j / a_i	20	20	20
30	11	6	9
30	13	10	7

В клетках, расположенных на пересечении строк a_i и столбцов b_j, находятся значения времени обработки i-й детали на j-м станке t_{ij}.

Целевая функция имеет вид

$$T = 11x_1 + 6x_2 + 9x_3 + 13x_4 + 10x_5 + 7x_6 \rightarrow \min.$$

Ограничения:

$$x_1 + x_2 + x_3 <= 30,$$
$$x_4 + x_5 + x_6 < = 30,$$
$$x_1 + x_4 > = 20,$$
$$x_2 + x_5 > = 20,$$
$$x_3 + x_6 > = 20.$$

Решение

	20	20	20
30	10	20	0
30	10	0	20

Наименьшее количество затрат $T_{\min} = 500$.

Индивидуальные задания

Найти решение задачи определения плана обработки деталей по критерию минимальных затрат, используя математическую модель транспортной задачи.

В табл. 3.12 приведены численные значения переменных a_i, b_j, c_{ij} для вариантов (1–12). Назначение данных в строках и столбцах таблиц аналогичны данным табл. 3.11.

Таблица 3.12

Исходные данные определения плана обработки деталей с минимальной продолжительностью

№ варианта	b_j / a_i	7	5	9	7	2
1	4	26	30	17	10	16
	6	30	37	26	9	23
	10	13	4	32	3	1
	10	3	1	5	14	24
2	b_j / a_i	11	11	11	11	16
	15	27	20	29	26	25
	15	3	14	5	15	24
	15	9	2	32	4	13
	15	20	27	1	27	19

Продолжение табл. 3.12

№ варианта	b_j / a_i	12	12	12	12	12
3	20	25	1	22	19	1
	20	21	28	11	4	3
	20	26	29	33	26	24
	20	1	10	3	29	27
4	b_j / a_i	12	12	12	12	12
	30	26	24	26	29	13
	15	30	29	26	23	17
	4	10	37	30	7	7
	9	16	29	30	3	13
5	b_j / a_i	8	8	8	8	28
	18	31	22	2	13	7
	12	27	20	4	24	9
	17	3	16	35	5	4
	13	28	11	17	20	29
6	b_j / a_i	19	20	19	17	5
	15	20	17	9	20	30
	15	13	14	24	26	26
	19	22	24	40	27	29
	11	25	12	11	34	23
7	b_j / a_i	19	20	19	17	5
	21	40	24	11	12	25
	19	26	14	29	20	24
	15	27	14	24	10	18
	25	6	14	28	18	2
8	b_j / a_i	19	20	19	17	5
	9	15	15	3	6	10
	11	23	18	13	27	12
	14	30	1	15	24	25
	16	8	26	7	38	39
9	b_j / a_i	19	20	19	17	5
	19	17	29	28	8	22
	13	31	27	16	29	13
	20	30	34	7	26	17
	11	19	10	16	2	18

Окончание табл. 3.12

№ варианта	a_i \ b_j	19	20	19	17	5
10	16	10	40	2	5	6
	15	5	39	9	5	7
	14	16	24	24	6	26
	15	3	28	4	35	8
11	a_i \ b_j	19	20	19	17	5
	17	22	11	25	7	21
	22	28	14	8	1	14
	9	13	12	28	15	21
	26	21	3	14	27	43
12	a_i \ b_j	19	20	19	17	5
	28	12	24	4	2	3
	13	20	20	15	27	7
	15	15	15	22	25	19
	30	2	6	3	15	5

3.3. Задача о назначениях

3.3.1. Математическая постановка задачи

Имеется m потенциальных исполнителей ($j = \overline{1,m}$) соответственно одной из имеющихся m работ ($i = \overline{1,m}$). Известны затраты c_{ij} на выполнение j-м исполнителем i-й работы. Требуется назначить каждого исполнителя на одну работу так, чтобы минимизировать суммарные затраты на все работы.

Целевая функция
$$Z = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} \to \min, \qquad (3.23)$$

при ограничениях:
$$\sum_{j=1}^{m} x_{ij} = 1, \ i = \overline{1,m}. \qquad (3.24)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \ j = \overline{1,n}. \qquad (3.25)$$

$$x_{ij} = \begin{cases} 1, \text{ если } i\text{-я работа поручается } j\text{-му исполнителю,} \\ 0 \text{ – в противном случае.} \end{cases}$$

Ограничение (3.24) указывает, что на каждую *i*-ю работу должен быть назначен только один исполнитель. Ограничение (3.25) указывает, что каждый *j*-й исполнитель должен быть назначен для выполнения только одной работы.

Если число работ не равно числу исполнителей, т. е. $i \neq j$, то задача о назначениях называется задачей открытого типа, в противном случае − закрытого.

Индивидуальные задания

В задачах данного раздела найти решение по критерию минимума затрат, используя математическую модель задачи о назначениях.

В таблице 3.13 приведены численные значения исходных данных.

Введены следующие обозначения:

i – выполняемые работы;

j – исполнители;

c_{ij} – затраты на выполнение *j*-м исполнителем *i*-й работы.

Таблица 3.13

Исходные данные, C_{ij}

	C_{ij}	*j*							
			1	2	3	4	5	6	7
№ 1	*i*	1	3	10	5	9	16	8	17
		2	6	8	11	8	18	19	20
		3	7	13	10	3	4	14	18
		4	5	9	6	21	12	17	22
		5	5	4	11	6	13	14	11
		6	17	7	12	13	16	7	9
		7	13	0	8	8	10	12	17
	C_{ij}	*j*							
			1	2	3	4	5	6	7
№ 2	*i*	1	4	5	9	5	6	14	6
		2	8	12	4	13	16	15	16
		3	2	15	8	10	17	7	9
		4	14	8	4	9	5	6	7
		5	3	5	4	12	10	11	13
		6	10	9	11	5	12	6	8
		7	13	4	8	10	5	16	3
	C_{ij}	*j*							
			1	2	3	4	5	6	7
№ 3	*i*	1	5	13	6	10	13	8	9
		2	10	9	7	11	8	12	11
		3	11	5	8	12	4	18	4
		4	12	6	9	8	5	8	5
		5	9	4	4	5	6	6	7
		6	11	8	7	4	7	3	8
		7	6	5	8	12	13	9	14

Продолжение табл. 3.13

№ 4	C_{ij}	*j*						
			1	2	3	4	5	6	7
	i	1	7	8	10	11	7	15	12
		2	1	2	5	6	10	8	4
		3	8	11	9	2	16	3	6
		4	5	2	5	14	3	10	5
		5	8	7	6	7	13	8	14
		6	6	8	17	10	11	9	5
		7	15	18	5	9	12	6	10
№ 5	C_{ij}	*j*						
			1	2	3	4	5	6	7
	i	1	8	4	5	18	6	1	9
		2	9	5	7	2	4	8	4
		3	1	10	5	6	12	9	6
		4	2	4	7	13	10	8	5
		5	12	3	11	9	12	10	11
		6	5	13	8	2	3	12	13
		7	7	6	4	18	5	6	7
№ 6	C_{ij}	*j*						
			1	2	3	4	5	6	7
	i	1	1	17	1	4	5	18	2
		2	21	5	6	7	8	6	16
		3	9	10	9	14	10	15	8
		4	14	2	3	13	6	7	15
		5	10	8	12	1	11	2	4
		6	6	7	1	10	3	8	6
		7	3	19	4	12	13	20	4
№ 7	C_{ij}	*j*						
			1	2	3	4	5	6	7
	i	1	8	4	6	7	8	11	5
		2	13	5	12	13	5	6	8
		3	10	13	14	17	3	4	2
		4	5	6	5	6	4	15	3
		5	19	20	10	7	8	6	7
		6	12	13	1	13	15	8	7
		7	4	1	2	2	4	16	9
№ 8	C_{ij}	*j*						
			1	2	3	4	5	6	7
	i	1	12	13	8	5	5	15	17
		2	1	2	6	8	7	3	4
		3	6	7	2	16	3	9	10
		4	4	5	15	20	19	11	4
		5	10	1	2	18	17	3	5
		6	6	5	4	10	5	7	8
		7	18	19	11	12	14	14	15

Окончание табл. 3.13

	C_{ij}	j							
			1	2	3	4	5	6	7
№ 9	i	1	13	14	5	12	3	6	14
		2	7	1	9	4	11	2	10
		3	12	4	7	6	8	7	4
		4	5	4	6	1	8	6	10
		5	8	9	9	7	10	3	5
		6	7	4	4	5	4	3	8
		7	15	3	3	13	5	6	16
	C_{ij}	j							
			1	2	3	4	5	6	7
№ 10	i	1	1	4	5	8	9	4	5
		2	5	6	7	8	10	11	12
		3	4	18	4	7	6	7	8
		4	5	4	3	6	7	10	11
		5	9	10	8	9	5	13	6
		6	6	8	11	13	7	8	9
		7	12	4	5	6	2	5	4

Пример

Найти оптимальное решение задачи для индивидуального задания № 1 с помощью математической модели и программы ПК.

Решение:

0 0 0 0 0 1 0
1 0 0 0 0 0 0
0 0 0 0 1 0 0
0 0 1 0 0 0 0
0 0 0 1 0 0 0
0 0 0 0 0 0 1
0 1 0 0 0 0 0

Наименьшее количество затрат $L_{\min} = 39.00$

Контрольные вопросы

1. Сформулируйте математическую модель задачи линейного программирования.
2. Что является решением задачи ЛП? Почему на значения переменных накладывается ограничение неотрицательности: $x_1, x_2, ..., x_n \geq 0$?

3. Что является целевой функцией и системой ограничений в модели ЛП?
4. Объясните, почему в математической постановке задачи ЛП количество уравнений системы ограничений меньше количества переменных?
5. Что такое базис, базисные переменные, базисное решение?
6. Что такое свободные переменные?
7. Приведите пример решения основной задачи линейного программирования в машиностроении.
8. Приведите последовательность шагов симплекс-метода решения задачи ЛП.
9. Сформулируйте математическую модель задачи транспортного типа.
10. Сформулируйте математическую модель задачи о назначениях.

ГЛАВА 4. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Динамическое программирование (иначе «динамическое планирование») − это особый метод оптимизации решения задач. Задачи динамического программирования (ДП) являются «многошаговыми», где на каждом из шагов определяется решение некоторой частной задачи оптимизации, обусловленной исходной. Поэтому ДП определяет методы нахождения оптимального решения отдельных классов задач математического программирования, которые могут относиться к задачам как линейного, так и нелинейного программирования.

4.1. Общая постановка задачи динамического программирования. Геометрическая интерпретация

На рис. 4.1 представлен управляемый процесс, который включает ряд последовательных «шагов» или «этапов».

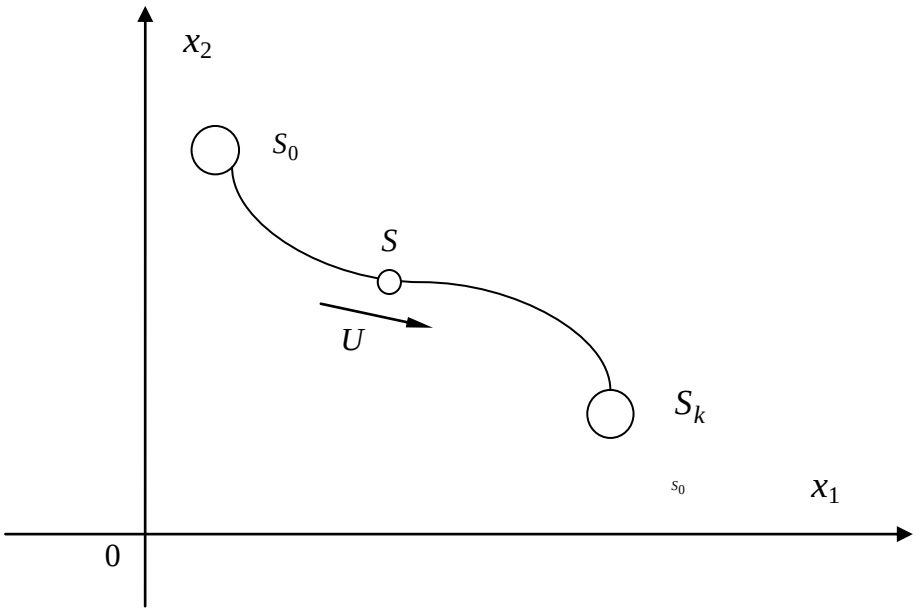


Рис. 4.1. Геометрическая интерпретация задачи динамического программирования: $\tilde{S}_0$ – область возможных начальных состояний системы; $\tilde{S}_k$ – область допустимых конечных состояний системы; U – управление

Предположим, что состояние системы характеризуется некоторой точкой S, на плоскости $x_1 0 x_2$, и эта точка под действием управления U перемещается из $\tilde{S}_0$ в $\tilde{S}_k$.

Каждому управлению U движения точки соответствует значение функции $W(U)$ (длина пути, ресурсы и др.). Задача состоит в том, чтобы

из всех допустимых траекторий движения точки S найти такую, которая получается в результате реализации управления U^*, обеспечивающего оптимальное значение функции $W(U^*)$.

4.2. Принцип оптимальности динамического программирования

Каково бы ни было состояние системы перед очередным шагом, надо выбрать управление на этом шаге таким образом, чтобы выигрыш на данном шаге плюс оптимальный выигрыш на всех последующих шагах были максимальными. При этом неважно, каким образом мы получаем частное оптимальное решение на предыдущем шаге. Переход из одного состояния в другое определяется функционалом любого типа.

Для шага k целевая функция F_k будет представлена как

$$F_k - F_{k-1} + f(U_k), \qquad (4.1)$$

где F_{k-1} − оптимальное значение функции на предыдущем шаге, $f(U_k)$ − управление на k-м шаге.

В задачах динамического программирования решение осуществляется в два этапа.

1 этап

Процесс выполняется «от конца к началу», т. е. сначала находится условно-оптимальное управление на последнем шаге, затем на двух последних, на трех и т. д. вплоть до первого.

2 этап

На основе данных первого этапа находится оптимальное управление. Последовательность шагов осуществляется «от начала к концу».

Рассмотрим действие принципа оптимальности ДП на примерах.

Пример 1

Состояние некоторой системы характеризуется сетевым графом, представленным на рис. 4.2. Вершины графа определяют переход из одного состояния в другое, а дуги соответствуют управлению U.

Требуется найти минимальное значение целевой функции (функции затрат) перехода системы из состояния $S1$ в $S13$.

Решение оформим в виде таблицы (табл. 4.1), в которой будем хранить оптимальное управление S_i и соответствующее ему значение целевой функции F_k для всех возможных состояний системы после каждого шага. Число строк таблицы равно числу шагов n процесса управления (n = 4). Число столбцов таблицы равно числу возможных состояний системы S_i.

В ячейках таблицы содержатся следующие данные: в числителе записывается оптимальное управление для каждого шага (S_i), а в знаменателе – соответствующее значение целевой функции F_k (4.1).

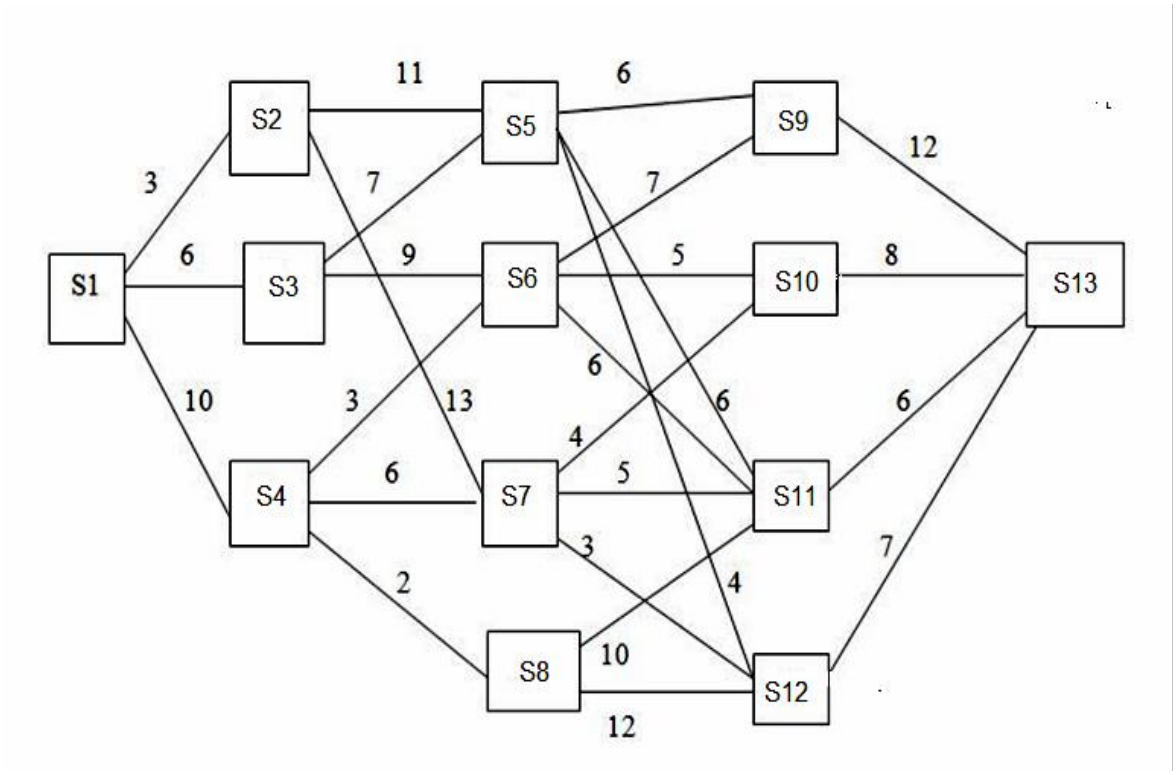


Рис. 4.2. Сетевой граф системы

1 этап

Решение начинается с последнего шага n, т. е. с $S13$.

На последнем шаге n условно-оптимальным управлением будет $S11$ и целевая функция $F\min = 6$.

На шаге $(n - 1)$ условно-оптимальным управлением будет $S7$ и $F\min = 10$.

Для шага $(n - 2)$ условно-оптимальным управлением будет $S4$ и $F\min = 15$.

Для шага $(n - 3)$ условно-оптимальным управлением будет $S1$ и $F\min = 24$.

Для определения оптимальной траектории управления системой необходимо выполнить второй этап вычислительного процесса, т. е. рассмотреть прохождение шагов с 1-го и по 4-й на основании данных табл. 4.1.

2 этап

Для первого шага имеем оптимальное управление $S1$ и значение целевой функции $F_1 = F\min = 24$.

Для второго шага имеем управление $S3$ и $F_2 = 18$.

Для третьего шага имеем управление $S5$ и $F_3 = 11$.

Для четвертого шага имеем управление $S12$ и $F_4 = 7$.

Для управления $S13$ $F = 0$.

Таблица 4.1

Оптимальное управление и соответствующее ему значение целевой функции S_i/F_k

шаг	S_i												
	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$	$S9$	$S10$	$S11$	$S12$	$S13$
1									13/12	13/8	13/6	13/7	
2					12/11	11/12	12/10	11/16					
3		5/22	5/18	6/15									
4	3/24												

Таким образом, оптимальным управлением системы будет:

$$U^* = (S1, S3, S5, S12, S13),$$

при котором целевая функция принимает минимальное значение:

$$W(U^*) = F\min = 6 + 7 + 4 + 7 = 24.$$

Пример 2

Космическая ракета состоит из m ступеней, а процесс ее вывода на орбиту – из m шагов, в конце каждого из которых очередная ступень сбрасывается.

На все ступени выделен общий вес

$$G = G_1 + G_2 + ... + G_m,$$

где G_i – вес i-й ступени.

В результате i-го шага (сброс i-ступени) ракета получает приращение скорости Δi.

При этом Δi зависит от веса i-й ступени и суммарного веса всех оставшихся + вес кабины.

Скорость движения ракеты

$$V = \sum_{i=1}^{m} \Delta i,$$

где Δi – выигрыш на i-м шаге.

Управление x представляет собой совокупность весов всех ступеней G_i:

$$x = (G_1, G_2, ..., G_m).$$

Оптимальным управлением x^* будет распределение весов по ступеням, при котором скорость V максимальна.

Как нужно распределить вес G между ступенями, чтобы скорость ракеты V при ее выводе на орбиту была максимальной?

Решение данной задачи осуществляется по аналогии с примером 1.

4.3. Математическая формулировка принципа оптимальности

Динамическое программирование представляет собой поэтапное планирование многошагового процесса, при котором на каждом этапе, учитывая развитие всего процесса, оптимизируется только один шаг, т. е. при принятии решения учитывается будущее.

Однако в каждом процессе имеется последний шаг k-й, принятие решения на котором не зависит от будущего. Поэтому на этом шаге выбирают управление, позволяющее получить наибольший эффект.

Спланировав этот шаг, к нему можно присоединить предпоследний $(k - 1)$-й шаг, к которому, в свою очередь, $(k - 2)$-й и т. д. В конечном итоге приходят в начальное состояние системы S_0. Процесс динамического программирования «разворачивается» как бы от конца к началу.

Таким образом, для того чтобы спланировать k-й шаг, надо знать состояние системы на $(k - 1)$-й шаге. Далее, исходя из характера данного процесса, делают различные предположения о возможных состояниях системы на этом k-м шаге. Для каждого предположения выбирают оптимальное управление на последнем k-м шаге. Такое оптимальное управление называется условно-оптимальным (УОУ), причем на каждом шаге требуется находить условно-оптимальное управление для любого из возможных исходов предшествующего шага.

Введем обозначения:

$F_n(X^{(0)})$ – максимальный доход, получаемый за n шагов при переходе системы S из начального состояния $X^{(0)}$ в конечное $X^{(n)}$, при реализации оптимальной стратегии управления

$$U^* = (U_1^*, U_2^*, ..., U_n^*).$$

$F_{n-k}(X^{(k)})$ – максимальный доход, получаемый при переходе системы S из состояния $X^{(k)}$ в конечное состояние $X^{(n)}$ при реализации оптимальной стратегии управления на оставшихся $n - k$ шагах. Тогда

$$F_n(X^{(0)}) = \max_{U_{k+1}\ \text{ï ðè}\ (k=\overline{0,n-1})}\left[W_1(X^{(0)},U_1)+...+W_n(X_{(n-1)},U_n)\right];$$
$$F_{n-k}(X^{(k)}) = \max_{U_{k+1}\ \text{ï ðè}\ (k=\overline{0,n-1})}\left[W_{1+k}(X^{(k)},U_{k+1})+...+F_{n-k-1}(X^{(k+1)})\right]. \quad (4.2)$$

Выражение (4.2) представляет собой математическую запись принципа оптимальности и носит название основного ***функционального уравнения Беллмана*** или рекуррентного соотношения.

Используя данное уравнение, находим решение задачи динамического программирования.

Полагая $k = n - 1$ в уравнении (4.2), получим:

$$F_1(X^{(n-1)}) = \max\,[W_n(X^{(n-1)}, U_n) + F_0(X^{(n)})], \quad (4.3)$$

где $F_0(X^{(n)})$ неизвестно либо равно 0.

Всевозможные допустимые состояния на $(n - 1)$ шаге:

$$X_1^{(n-1)},\ X_2^{(n-1)},\ ...,\ X_m^{(n-1)}.$$

Используя уравнение (4.3), находим условно-оптимальные решения:

$$U_n^0(X_1^{(n-1)}),\ U_n^0(X_2^{(n-1)}),\ ...,\ U_n^0(X_m^{(n-1)}),\ ...$$

и соответствующие значения функции (4.3)

$$F_1^0(X_1^{(n-1)}),\ F_1^0(X_2^{(n-1)}),\ ...,\ F_1^0(X_m^{(n-1)}),\ ...$$

Таким образом, на n-м шаге находим условно оптимальное управление при любом допустимом состоянии системы S после $(n - 1)$-го шага. Иными словами, в каком бы состоянии система ни оказалась после $(n - 1)$-го шага, нам уже известно, какое следует принять решение на n-м шаге. Известно также и соответствующее значение функции (4.3).

Рассмотрим функциональное уравнение при $k = n - 2$:

$$F_2(X^{(n-2)}) = \max_{U_{n-1}}\left[W_{n-1}(X^{(n-2)},U_{n-1}) + F_1(X^{(n-1)})\right]. \quad (4.4)$$

Чтобы найти значение F_2 для всех допустимых значений $X^{(n-2)}$, необходимо знать

$$W_{n-1}(X^{(2)},\ U_{n-1}) \text{ и } F_1(X^{(n-1)}).$$

Значение $F_1(X^{(n-1)})$ уже определили. Поэтому нужно произвести вычисления для $W_{n-1}(X^{(n-2)},\ U_{n-1})$ при некотором наборе допустимых значений $X^{(n-2)}$ и соответствующих управлений U_{n-1} для каждого $X^{(n-1)}$.

Каждое из таких управлений совместно с уже выбранным управлением на последнем шаге обеспечивает максимальное значение дохода на двух последних шагах.

Последовательно дойдем до первого шага. На этом шаге известно, в каком состоянии может находиться система (S_0). Остается лишь вы-

брать управление, которое является наилучшим с учетом условно оптимальных управлений, уже принятых на всех последующих шагах.

Таким образом, чтобы найти оптимальную стратегию управления – определить искомое решение задачи, нужно теперь пройти всю последовательность шагов от начала к концу, а именно:

- 1_шаг – оптимальное управление $U_1^* = U_1^0$;
- 2_шаг – находим X_1^*, в которое переводит управление U_1^*;
- 3_шаг – $U_2^* = U_2^0$;
- 4_шаг – зная U_2^*, находим X_2^*, а значит определяем U_3^* и т. д.;
- 5_шаг – находим оптимальную стратегию управления $U^* = (U_1^*,\ U_2^*,\ ...,\ U_n^*)$ и максимально возможный доход.

4.4. Задача замены оборудования

В определенный момент времени на предприятии установлено новое оборудование. От времени использования оборудования зависят его производительность, затраты на содержание и ремонт и прибыль. Затраты, связанные с приобретением и установлением оборудования, идентичного с установленным, составляют 40 тыс. руб.

Составить план замены оборудования в течение 5 лет, при котором общая прибыль за данный период времени максимальна.

Исходные данные приведены в табл. 4.2.

Таблица 4.2

Исходные данные плана замены оборудования в течение 5 лет

Показатели	Время τ, в течение которого используется оборудование (лет)					
	1	2	3	4	5	6
Годовой выпуск продукции $R(\tau)$ в стоимостном выражении, (тыс. руб)	80	75	65	60	60	55
Ежегодные затраты $Z(\tau)$, связанные с содержанием и ремонтом оборудования (тыс. руб)	20	25	30	35	45	55
Прибыль $R(\tau) - Z(\tau)$	60	50	35	25	15	0

Решение

$k = \overline{1,5}$ – количество лет;

U_1 – не заменять оборудование;
U_2 – заменять оборудование;
U_k – управление к началу k-го года.
} управление

Прибыль за k-й год составит:

$$F_k(\tau^{(k)}, U_k) = \begin{cases} R(\tau(k)) - z(\tau(k)), & \text{при } U_1 \text{ (не заменять);} \\ \\ R(\tau(k) = 0) - z(\tau(k) = 0) - Cn, & \text{при } U_2 \text{ (заменять),} \end{cases} \quad (4.5)$$

где U_k – управление к началу k-го года;

$\tau^{(k)}$ – возраст оборудования к началу k-го года;

C_n – стоимость нового оборудования.

Уравнение Беллмана будет иметь следующий вид:

$$F_k(\tau^{(k)}) = \max \begin{cases} R(\tau^{(k)}) - z(\tau^{(k)}) + F_{k+1}(\tau^{(k+1)}), \text{при } U_1; \\ \\ R(\tau^{(k)} = 0) - z(\tau^{(k)} = 0) - C_n + F_{k+1}(\tau^{(k+1)} = 1), \text{ при } U_2. \end{cases} \quad (4.6)$$

Решение начинаем с определения условно-оптимального решения (U^0) для последнего (5-го) года.

$$F_5(\tau^{(5)}) = \max \begin{cases} R(\tau^{(5)} = 1) - z(\tau^{(5)} = 1) + F_6(\tau^{(6)}); \\ \\ R(\tau^{(5)} = 0) - z(\tau^{(5)} = 0) - C_n + F_6(\tau^{(6)} = 1). \end{cases} \quad (4.7)$$

Так как $F_6(\tau^{(6)}) = 0$, то

$$F_5(\tau^{(5)}) = \max \begin{cases} R(\tau^{(5)} = 1) - z(\tau^{(5)} = 1) & - U_1; \\ \\ R(\tau^{(5)}) = 0) - z(\tau^{(5)} = 0) - C_n & - U_2. \end{cases} \quad (4.8)$$

Допустимые состояния оборудования к началу 5-го года будут:

$$\tau_0^{(5)} = 0 - \text{новое оборудование.}$$

$$\tau_1^{(5)} = 1,\ \tau_2^{(5)} = 2,\ \tau_3^{(5)} = 3,\ \tau_4^{(5)} = 4.$$

Подставляя $\tau_1^{(5)} = 1$ и данные таблицы (4.2) в уравнение (4.8), получим

$$F_5(\tau_1^{(5)}) = \max \left\{ \begin{array}{l} 75 - 25 \\ \\ 80 - 20 - 40 \end{array} \right\} = 50 \qquad - U_1;$$

$$F_5(\tau_2^{(5)}) = \max \left\{ \begin{array}{l} 65 - 30 \\ \\ 80 - 20 - 40 \end{array} \right\} = 35 \qquad - U_1;$$

$$F_5(\tau_3^{(5)}) = \max \left\{ \begin{array}{l} 60 - 35 \\ \\ 80 - 20 - 40 \end{array} \right\} = 25 \qquad - U_1;$$

$$F_5(\tau_4^{(5)}) = \max \left\{ \begin{array}{l} 60 - 45 \\ \\ 80 - 20 - 40 \end{array} \right\} = 20 \qquad - U_2.$$

Допустимые состояния оборудования к началу 5-го года сведены в табл. 4.3.

Таблица 4.3

Условно-оптимальное решение для последнего (5-го) года

Возраст оборудования $\tau^{(5)}$, лет	Значение функции $F(\tau^{(5)})$, тыс. руб.	Условно-оптимальное решение U^0
1	50	U_1
2	35	U_1
3	25	U_1
4	20	U_2

Рассмотрим возможные состояния к началу 4-го года:

$$\tau_0^{(4)} = 0,\ \tau_1^{(4)} = 1,\ \tau_2^{(4)} = 2,\ \ \tau_3^{(4)} = 3.$$

$$F_4(\tau_1^{(4)}) = \max \left\{ \begin{array}{l} R(\tau_1^{(4)}=1) - z(\tau_1^{(4)} = 1) + F_5(\tau_2^{(5)} = 2), \\ \\ R(\tau_0^{(4)} = 0) - z(\tau_0^{(4)} = 0) - C_n + F_5(\tau_1^{(5)} = 1). \end{array} \right. \qquad (4.9)$$

$$F_4(\tau_1^{(4)}) = \max \left\{ \begin{array}{l} 75 - 25 + 35 \\ \\ 80 - 20 - 40 + 50 \end{array} \right\} = 85 \qquad - U_1.$$

По аналогии с $F_4(\tau_1^{(4)})$ вычислим значение функций $F_4(\tau_2^{(4)})$ и $F_4(\tau_3^{(4)})$

$$F_4(\tau_2^{(4)}) = \max \begin{Bmatrix} 65 - 30 + 25 \\ \\ 80 - 20 - 40 + 50 \end{Bmatrix} = 70 \qquad - U_2;$$

$$F_4(\tau_3^{(4)}) = \max \begin{Bmatrix} 60 - 35 + 20 \\ \\ 80 - 20 - 40 + 50 \end{Bmatrix} = 70 \qquad - U_2.$$

Допустимые состояния оборудования к началу 4-го года сведены в табл. 4.4.

Таблица 4.4

Условно-оптимальное решение для 4-го года

Возраст оборудования $\tau^{(4)}$, лет	Значение функции $F(\tau^{(4)})$, тыс. руб.	Условно-оптимальное решение U^0
1	85	U_1
2	70	U_2
3	70	U_2

Определим УОР для каждого из допустимых состояний оборудования к началу 3-го года.

$\tau_1^{(3)} = 1, \tau_2^{(3)} = 2.$

$$F_3(\tau_1^{(3)}) = \max \begin{cases} R(\tau_1^{(3)}=1) - z(\tau_1^{(3)}=1) + F_4(\tau_2^{(4)} = 2); \\ \\ R(\tau_0^{(3)} = 0) - z(\tau_0^{(3)} = 0) - C_n + F_4(\tau_1^{(4)} = 1). \end{cases}$$

$$F_3(\tau_2^{(3)}) = \max \begin{cases} R(\tau_2^{(3)}= 2) - z(\tau_2^{(3)} = 2) + F_4(\tau_3^{(4)} = 3); \\ \\ R(\tau_0^{(3)} = 0) - z(\tau_0^{(3)} = 0) - C_n + F_4(\tau_1^{(4)} = 1). \end{cases}$$

Используя таблицы табл. 4.3 и табл. 4.4 получим:

$$F_3(\tau_1^{(3)}) = \max \begin{Bmatrix} 75 - 25 + 70 \\ \\ 80 - 20 - 40 + 85 \end{Bmatrix} = 120 \qquad - U_1;$$

$$F_3(\tau_2^{(3)}) = \max \left\{ \begin{array}{l} 65 - 30 + 70 \\ \\ 80 - 20 - 40 + 85 \end{array} \right\} = 105 \begin{array}{l} - U_2; \\ \\ - U_1. \end{array}$$

Независимо от того, какое будет принято решение, прибыль будет одна и та же – 105 тыс. руб.

Допустимые состояния оборудования к началу 3-го года сведены в табл. 4.5.

Таблица 4.5

Условно-оптимальное решение для 3-го года

Возраст оборудования $\tau^{(3)}$, лет	Значение функции $F(\tau^{(3)})$, тыс. руб.	Условно-оптимальное решение, U^0
1	120	U_1
2	105	U_2

Наконец, рассмотрим допустимые состояния к началу 2-го года. Очевидно, на данный момент времени возраст оборудования может быть равен только 1 году.

$\tau_1^{(2)} = 1.$

$$F_2(\tau_1^{(2)}) = \max \left\{ \begin{array}{l} 75 - 25 + 105 \\ \\ 80 - 20 - 40 + 120 \end{array} \right\} = 155 \quad - U_1.$$

Таблица 4.6

Условно-оптимальное решение для 2-го года

Возраст оборудования $\tau^{(2)}$, лет	Значение функции $F(\tau^{(2)})$, тыс. руб.	Условно-оптимальное решение, U^0
1	155	U_1

Оборудование следует сохранить.

К началу 1-го года установлено новое оборудование.

$\tau_0^{(1)} = 0.$

$$F_1(\tau_0^{(1)}) = R(\tau_0^{(1)} = 0) - z(\tau_0^{(1)} = 0) + F_2(\tau_0^{(1)} = 1) = 80 - 20 + 155 = 215.$$

Для определения оптимального плана замены оборудования необходимо выполнить второй этап вычислительного процесса, т. е. рассмотреть прохождение шагов с начала 1-го и до начала 5-го года.

На основании данных табл. 4.3–4.6 получен оптимальный план замены оборудования. Результаты представлены в табл. 4.7.

Таблица 4.7

Оптимальный план замены оборудования

Годы	1	2	3	4	5
Оптимальное решение	Сохранить оборудование	С.О.	Произвести замену	С.О.	С.О.
Управление	U_1	U_1	U_2	U_1	U_1
Целевая функция	60	50	20	50	35
Максимальная прибыль	60	110	130	180	215

Для первого года решение единственное − U_1.

Значит, возраст оборудования к началу 2-го года равен 1-му году. При таком возрасте, согласно табл. 4.6, оптимальным является решение сохранить оборудование (U_1).

Реализация такого решения приводит к тому, что возраст оборудования к началу 3-го года становится равным 2-м годам. При таком возрасте (табл. 4.5) оборудование на 3-м году следует заменить (U_2).

После его замены к началу 4-го года возраст оборудования составит 1 год. Далее, как видно из табл. 4.4, оборудование менять не следует. Оптимальным является решение сохранить оборудование (U_1).

К началу 5-го года возраст оборудования составляет 2 года. Из табл. 4.3 следует, что его менять не надо (U_1).

Таким образом, максимальная прибыль предприятия в течение пятилетнего периода может быть равна 215 тыс. руб.

Контрольные вопросы

1. Сформулируйте математическую постановку задачи динамического программирования.
2. Приведите геометрическую интерпретацию задачи ДП.
3. Сформулируйте принцип оптимальности ДП.
4. В чем заключается многошаговый принцип решения задач ДП?
5. Почему решение задачи ДП начинается с последнего *n*-го шага?
6. Объясните понятие «оптимальная стратегия управления».
7. Объясните понятие «условно оптимальное управление».
8. Приведите математическую запись основного функционального уравнения Беллмана.
9. Рассмотрите пример решения задачи «о замене оборудования» методом ДП.
10. В чем заключаются отличительные особенности задач динамического и линейного программирования?

ГЛАВА 5. ОСНОВЫ ТЕОРИИ ГРАФОВ

Теорию графов начали разрабатывать для решения некоторых задач о геометрических конфигурациях. Для иллюстрации многих практических задач часто используют рисунки, состоящие из точек (вершин), представляющих основные элементы некоторой ситуации, и линий (ребер), соединяющих определенные пары этих вершин и представляющих связи между ними. Такие рисунки известны под общим названием графов.

Графы встречаются в различных областях науки и практической деятельности под разными названиями: «структуры», «сети», «социограммы» и т. д. В теории систем распространено графовое представление структуры системы: вершинами графа обозначаются отдельные блоки (подсистемы), а ребрами – связи между этими блоками. Удобны графы также при исследовании систем методом пространства состояний. В этом случае вершины отображают состояние системы, а ребра – траекторию движения системы в пространстве состояний или действия, которые приводят к переходу из одного состояния в другое.

В машиностроении теория графов используется при решении таких задач, как размерный анализ, сборка изделий, проектирование технологических процессов и др. Например, возможные варианты маршрута обработки детали можно представить в виде графа, где множество вершин характеризует состояние контуров операционных эскизов (сумма инструментальных участков), которые характеризуют межоперационные размеры, их точность, шероховатость поверхности и др. параметры. Множество дуг характеризует методы обработки, с помощью которых происходит переход от одного контура к другому, и представляет собой один из вариантов маршрута обработки.

5.1. Ориентированные графы

Наглядное представление об ориентированном графе можно получить, если представить себе некоторое множество точек плоскости X, называемых *вершинами*, и множество направленных отрезков U, соединяющих все или некоторые из вершин и называемых *дугами*. При этом несущественно, соединены ли точки отрезками прямых или криволинейными дугами, какова длина линии и другие геометрические характеристики конфигурации. Важно лишь то, что каждая линия соединяет какие-либо две из заданных точек.

Известны два основных способа задания графа. Граф может быть задан множеством точек или вершин x_1, x_2, ..., x_n, которое обозначается через X, и множеством дуг u_1, u_2, ..., u_n, которое обозначается символом U, соединяющих между собой все или часть этих точек.

Таким образом, математически граф G можно определить как пару множеств X и U:

$$G = (X, U).$$

На рис. 5.1 изображен граф, вершинами которого являются точки a, b, c, d, e, q, h, а дугами − отрезки (a, a), (c, b), (c, d), (c, e), (d, c), (d, d), (e,d), (q, h).

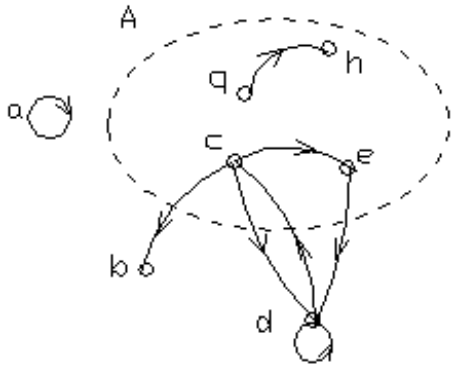


Рис. 5.1. Ориентированный граф

Другое, употребляемое чаще, задание графа состоит в следующем. Можно считать, что множество направленных дуг U, соединяющих элементы множества X, отображает это множество X само в себя. Поэтому можно считать граф заданным, если дано множество его вершин X и способ отображения Γ множества X в X.

Таким образом, граф G есть пара (X, Γ), состоящая из множества X и отображения Γ, заданного на этом множестве:

$$G = (X, \Gamma).$$

Так, для графа, изображенного на рис. 5.1, отображение определяется следующим образом:

$$\Gamma a = a;\ \Gamma b = \varnothing;\ \Gamma c = \{b, d, e\};\ \Gamma d = \{d, c\};$$

$$\Gamma e = d;\ \Gamma q = h;\ \Gamma h = 0.$$

Введем некоторые понятия и определения, служащие для описания различных видов графов.

Подграфом G_A графа $G = (X, \Gamma)$ называется граф, в который входит лишь часть вершин графа G, образующих множество A, вместе с дугами, соединяющими эти вершины, например очерченная пунктиром область на рис. 5.1.

Математический подграф $G_A = (A, \Gamma_A)$,

где $$A \subseteq X, \Gamma_A x = (\Gamma x) \bigcap A.$$

Частичным графом G_A по отношению к графу $G = (X, \Gamma)$ называется граф, содержащий только часть дуг графа G, т. е. определяемый условием

$$C_A = (X, \Delta),$$

где $$\Delta x \subseteq \Gamma x.$$

Так, на рис. 5.1 граф, образованный жирными дугами, является частичным графом.

Пример

Пусть $G = (X, \Gamma)$ – карта шоссейных дорог России. Тогда карта шоссейных дорог какой-либо области представляет собой подграф, а карта главных дорог России – частичный граф.

Другими важными понятиями являются понятия *пути* и *контура*. Ранее было дано определение дуги как направленного отрезка, соединяющего две вершины. Дуга, соединяющая вершины a и b и направленная от a к b, обозначается $u = (a, b)$.

Путем в графе G называют такую последовательность дуг

$$\mu = (u_1, u_2, ..., u_k),$$

в которой конец каждой предыдущей дуги совпадает с началом следующей. Путь μ, последовательными вершинами которого являются вершины $a, b, ..., m$, обозначается через

$$\mu = (a, b, ..., m).$$

Длиной пути

$$\mu = (u1, u2, ..., uk)$$

называют число $l(\mu)$ k, равное числу дуг, составляющих путь μ.

Иногда каждой дуге u_i приписывают некоторое число $l(u_i)$, называемое длиной дуги. Тогда длина пути определяется как сумма длин дуг, составляющих путь

$$l(\mu) = \sum_{i=1}^{m} l(u_i). \qquad (5.1)$$

Путь, в котором никакая дуга не встречается дважды, называется *простым.*

Путь, в котором никакая вершина не встречается дважды, называется *элементарным.*

Контур – это конечный путь

$$\mu = (x_1, x_2, ..., x_k),$$

у которого начальная вершина x_1 совпадает с конечной x_k.

При этом контур называется *элементарным*, если все его вершины различны (за исключением начальной и конечной, которые совпадают).

Контур единичной длины, образованный дугой вида (a, a), называется петлей. Так, на рис. 5.1 (e, d, c, b) − путь, (c, e, d, c) − контур, (d, d) − петля.

Иногда удобно представлять графы в виде некоторых матриц, в частности, в виде матриц смежности и инциденций. Предварительно дадим два определения.

Вершины x и y являются смежными, если они различны и если существует дуга, идущая из x в y.

Дугу и называют инцидентной вершине x, если она заходит в эту вершину или исходит из нее.

Обозначим через x_1, ..., x_n вершины графа, а через u_1, ..., u_m его дуги. Введем числа:

$$r_{ij} = \begin{cases} 1, \text{ если имеется дуга, соединяющая вершину } i \text{ с вершиной } j; \\ 0, \text{ если такой дуги нет.} \end{cases}$$

Квадратная матрица $R = [r_{ij}]$ порядка $n \times n$ называется матрицей *смежности* графа. Введем числа:

$$s_{ij} = \begin{cases} +1, \text{ если } u_j \text{ исходит из } x_i; \\ -1, \text{ если } u_j \text{ заходит в } x_i; \\ 0, \text{ если } u_{ij} \text{ не инцидентна } x_i. \end{cases}$$

Матрица $S = [s_{ij}]$ порядка $n \times m$ называется *матрицей инциденций* для дуг графа. Матрицы инциденций в описанном виде применимы только к графам без петель. В случае наличия в графе петель эту матрицу следует разделить на две полуматрицы: положительную и отрицательную.

На рис. 5.2 изображен граф без петель, для которого приведены матрицы смежности и инциденций .

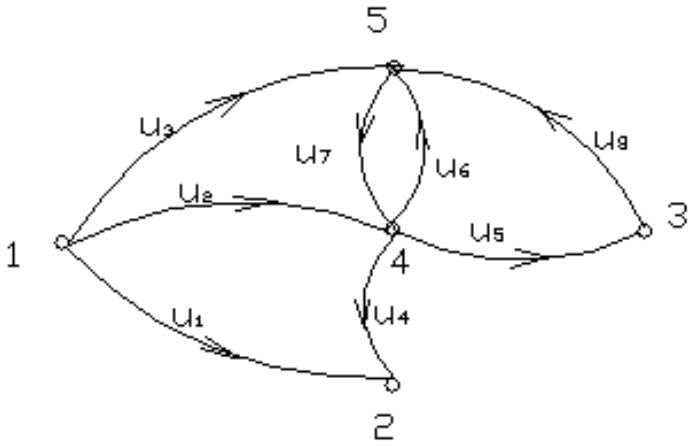


Рис. 5.2. Граф без петель

Матрица смежности

$$j = 1\ 2\ 3\ 4\ 5$$

$$\begin{bmatrix} 1\ 0\ 1\ 0\ 1\ 1 \\ 2\ 0\ 0\ 0\ 0\ 0 \\ 3\ 0\ 0\ 0\ 0\ 1 \\ 4\ 0\ 1\ 1\ 0\ 1 \\ 5\ 0\ 0\ 0\ 1\ 0 \end{bmatrix} \quad i$$

Матрица инциденций

$$j = 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 4 & 0 & -1 & 0 & 1 & 1 & 1 & -1 & 0 \\ 5 & 0 & 0 & -1 & 0 & 0 & -1 & 1 & -1 \end{bmatrix} \quad i$$

5.2. Неориентированные графы

Иногда граф рассматривают без учета ориентации дуг, тогда его называют *неориентированным графом* (рис. 5.3).

Для неориентированного графа понятие «дуга», «путь» и «контур» заменяются понятиями «ребро», «цепь», «цикл».

Ребро – это отрезок, соединяющий две вершины.

Цепь – последовательность ребер.

Цикл – цепь, у которой начальная и конечная вершины совпадают.

На рис. 5.3, *а* приведен пример неориентированного графа, у которого вершины обозначены цифрами, а ребра – буквами латинского алфавита.

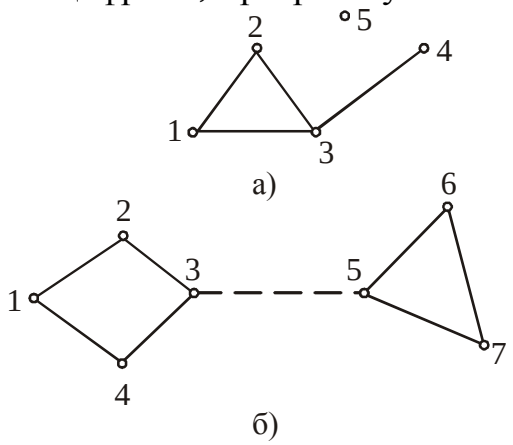


Рис. 5.3. Неориентированный граф

Описать неориентированный граф G можно путем задания пары множеств (X, U), где X – множество вершин; U – множество ребер. Однако более удобным является описание неориентированного графа с помощью матрицы смежности или инциденций, которые строятся аналогично соответствующим матрицам для ориентированных графов с той разницей, что не делается различия между заходящей в вершину и исходящей из нее дугами. При этом вершины x и y называют смежными, если существует соединяющее их ребро, а само это ребро называется инцидентным вершинам x и y.

Для графа рис. 5.3, *а* матрицы смежности и инциденций имеют вид:

Матрица смежности

j =	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	0	0
i = 3	1	1	0	1	0
4	0	0	1	0	0
5	0	0	0	0	0

Матрица инциденций

	a	b	c	d
1	1	1	0	1	0
2	1	1	0	0	0
x = 3	0	1	1	1	1
4	0	0	0	0	1
5	0	0	0	0	0

Приведем еще несколько определений, служащих для характеристики неориентированных графов.

Степенью вершины x, обозначаемой d_x, называют число ребер, инцидентных вершине x. Так, для графа на рис. 5.3, *а* имеем:

$$d_1 = 2,\ d_2 = 2,\ d_3 = 3,\ d_4 = 1,\ d_5 = 0.$$

Если $d_x = 1$, то вершину x называют *тупиковой* (вершина 4)*;* если $d_x = 0$, то вершину называют *изолированной* (вершина 5).

Между количеством вершин n и ребер m неориентированного графа существует следующее соотношение:

$$\sum_{i=1}^{n} d_j = 2m\,.$$

Для неориентированного графа понятия «подграф» и «частичный граф» аналогичны соответствующим понятиям для ориентированного графа.

С понятием неориентированного графа связана важная характеристика, называемая *связностью* графа. Говорят, что граф связен, если любые две его вершины можно соединить цепью. Если граф G не связен, то его можно разбить на такие подграфы G_i, что все вершины в каждом подграфе связны, а вершины из различных подграфов не связны.

Такие подграфы G_i называют *компонентами связности* графа G.

Если из графа на рис. 5.3, *а* исключить изолированную вершину 5, то полученный граф будет связным. Граф на рис. 5.3, *б* не связен и имеет две компоненты связности. Его можно превратить в связный, добавив ребро, соединяющее вершины 3 и 5 (штриховая линия).

Частный случай неориентированного графа − дерево. *Деревом* называют конечный связный неориентированный граф, не имеющий циклов (рис. 5.4). Ребра графа называются ветвями дерева.

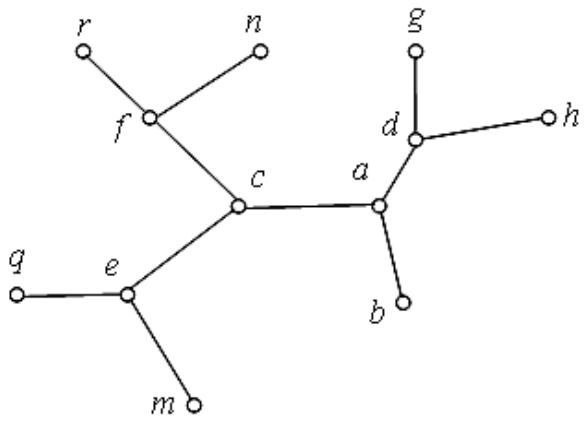


Рис. 5.4. Неориентированный граф − дерево

Если дано множество вершин *a*, *b*, *c*…, то дерево можно построить следующим образом. Одну из вершин, например *a*, примем за начальную и назовем ее корнем дерева. Из этой вершины проводим ребра в близлежащие вершины *b*, *c*, *d* ..., из них проводим ребра в соседние с

ними вершины e, f, g, h... и т. д. Таким образом, дерево можно построить, последовательно добавляя ребра в его вершинах. Это дает возможность установить связь между числом вершин и числом ребер дерева.

Простейшее дерево состоит из двух вершин, соединенных ребром. Каждый раз, когда мы добавляем еще одно ребро, в конце его прибавляется также и вершина. Следовательно, дерево с n вершинами имеет $(n - 1)$ ребер.

5.3. Изоморфизм графов

Один и тот же граф геометрически можно изобразить различными способами (рис. 5.5) Такие графы называют изоморфными.

Графы $G(X_1)$ и $G(X_2)$ называются изоморфными, если между множествами их вершин существует взаимно однозначное соответствие, такое, что вершины соединены ребрами в одном из графов в том и только том случае, когда соответствующие им вершины соединены в другом графе.

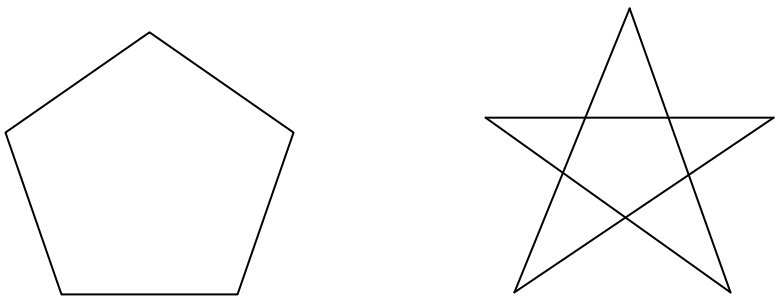


Рис. 5.5. Изоморфные графы

Если ребра графов ориентированы, то их направления в изоморфных графах также должны совпадать.

5.4. Задача о кратчайшем пути в графе

5.4.1. Постановка задачи

В практических приложениях имеет большое значение задача о нахождении кратчайшего пути между двумя вершинами связного неориентированного графа. К такой задаче сводятся многие задачи выбора наиболее экономичного (с точки зрения расстояния, времени или стоимости) маршрута на имеющейся карте дорог, многие задачи выбора наиболее экономичного способа перевода динамической системы из одного состояния в другое и т. п.

В математике разработан ряд методов для решения подобных задач. Однако весьма часто методы, основанные на использовании графов, оказываются наименее трудоемкими.

Задача о кратчайшем пути на графе в общем виде может быть сформулирована следующим образом. Дан неориентированный граф $G = (X, U)$. Каждому ребру этого графа приписано некоторое число $l(u) \geq 0$, называемое длиной ребра. В частных случаях $l(u)$ может быть расстоянием между вершинами, соединяемыми ребром u, временем или стоимостью проезда по этому ребру и т. п. При этом любая цепь μ будет характеризоваться длиной, определяемой соотношением (5.1).

Требуется для произвольных вершин a и b графа G найти путь μ_{ab}, причем такой, чтобы его полная длина была наименьшей.

Прежде чем найти общий метод решения этой задачи, рассмотрим правило для решения задачи частного вида, когда длина каждого ребра равна единице.

5.4.2. Нахождение кратчайшего пути в графе с ребрами единичной длины

Иногда приходится иметь дело с графами, ребра которых имеют одинаковую длину, принимаемую за единицу. Вершины такого графа обычно представляют собой состояния некоторой системы, в которой с некоторой точки зрения все переходы, делаемые за один шаг, эквивалентны.

Общее правило для нахождения кратчайшего пути в графе состоит в том, чтобы каждой вершине x приписать индекс λ_i, равный длине кратчайшего пути из данной вершины в конечную. Приписывание индексов вершинам в случае графа с ребрами единичной длины производится в следующем порядке:

1) конечной вершине x_0 приписывается индекс 0;

2) всем вершинам, из которых идет ребро в конечную вершину, приписывается индекс 1;

3) всем вершинам, еще не имеющим индексов, из которых идет ребро в вершину с индексом λ_x, приписывается индекс $\lambda_x + 1$. Этот процесс продолжается до тех пор, пока не будет помечена начальная вершина. По окончании разметки индекс у начальной вершины будет равен длине кратчайшего пути. Сам кратчайший путь найдем, если будем двигаться из начальной вершины в направлении убывания индексов.

5.4.3. Нахождение кратчайшего пути в графе с ребрами произвольной длины

Задача приписывания вершинам графа числовых индексов усложняется, если ребра графа имеют произвольную длину. Усложнение вызвано тем, что в сложном графе путь, проходящий через наименьшее число вершин, нередко имеет большую длину, чем некоторые обходные пути.

Процесс приписывания индексов для такого вида графов заключается в следующем:

1. Каждая вершина x_i помечается индексом λ_i. Первоначально конечной вершине x_0 приписывается индекс $\lambda_0 = 0$. Для остальных вершин предварительно полагаем

$$\lambda_i = \infty \ (i \neq 0).$$

2. Ищем такую дугу (x_i , x_j), для которой

$$\lambda_j - \lambda_i > l(x_i, x_j),$$

и заменяем индекс λ_j индексом

$$\lambda_{j'} = \lambda_j + l(x_i, x_j) < \lambda_j.$$

Продолжаем этот процесс замены индексов до тех пор, пока остается хотя бы одна дуга, для которой можно уменьшить λ_j.

Отметим два свойства, которыми будут обладать приписанные вершинам индексы.

1. Пусть (x_k, x_s) − произвольное ребро. Для него обязательно выполняется условие

$$\lambda_k \leq l(x_k, x_s),$$

так как если бы оно не выполнялось, индекс λ_s нужно было бы уменьшить.

2. Пусть x_p – произвольная вершина. При рассмотренном процессе приписывания индексов индекс λ_p монотонно уменьшается. Пусть x_q – последняя вершина, послужившая для его уменьшения. Тогда

$$\lambda_p = \lambda_q + (x_q, x_p).$$

Следовательно, для произвольной вершины x_p с индексом λ_p найдется вершина x_q, соединенная ребром с x_p, такая, что

$$\lambda_p - \lambda_q = l(x_q, x_p).$$

Эти свойства позволяют сформулировать следующее правило для нахождения кратчайшего пути.

Пусть $x_n = a$ – начальная вершина с индексом λ_n. Ищем вершину x_{p1}, такую, что

$$\lambda_n - \lambda_{p1} = l(x_{p1}, x_n).$$

Далее ищем вершину x_{p2}:

$$\lambda_{p1} - \lambda_{p2} = l(x_{p2}, x_{p1})$$

и т. д., до тех пор, пока не дойдем до конечной вершины

$$x_{p,\,k+1} = x_0 = b.$$

Путь

$$\mu_0 = (x_n, x_{p1}, ..., x_{pk}, x_0),$$

длина которого равна λ_n, является кратчайшим.

Метод нахождения кратчайшего пути проиллюстрирован на примере карты дорог, представленной в виде графа на рис. 5.6. Цифры у ребер указывают время проезда по каждой из дорог.

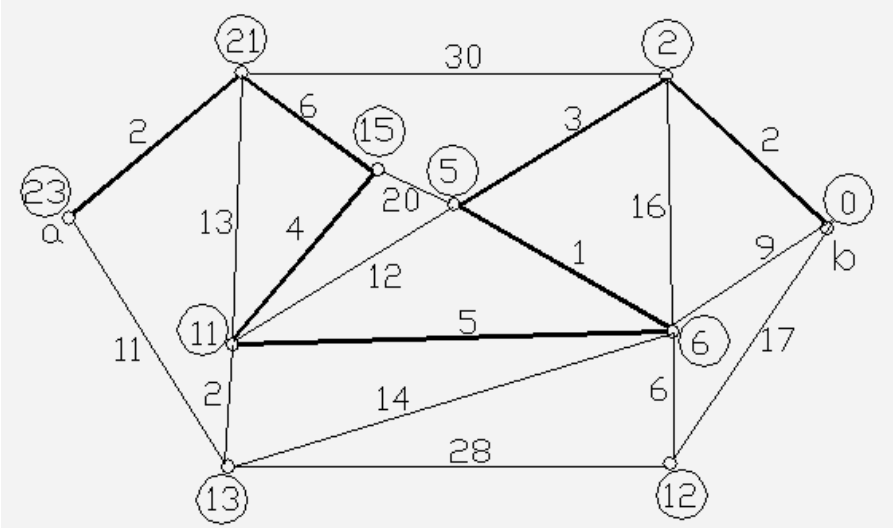


Рис. 5.6

Индексы вершин дают время проезда от данной вершины до конечной. Кратчайший путь из вершины b в a выделен на графе более яркими линиями.

5.5. Построение графа наименьшей длины

Большое практическое значение имеет следующая задача, которую можно сформулировать в виде задачи о проведении дорог. Имеется несколько городов a, b, c ... (рис. 5.7), которые нужно соединить между собой сетью дорог. Для каждой пары городов (x, y) известна стоимость $l(x, y)$ строительства соединяющей их дороги. Задача состоит в том, чтобы построить самую дешевую из возможных сетей дорог.

Вместо сети дорог можно рассматривать сеть линий электропередачи, сеть нефтепроводов и т. п. Называя в графе, изображающем сеть дорог, величину $l(x, y)$ длиной ребра (x, y), приходим к задаче о построении графа наименьшей длины. Поэтому далее в качестве стоимости дорог примем длину ребер графа.

Если имеются всего три вершины (*a*, *b*, *c*), то достаточно построить одну из соединяющих цепей *abc*, *acb*, *bac*, причем если *bc* − самое длинное ребро, то именно его и надо исключить, построив цепь *bac.*

Граф наименьшей длины всегда является деревом, так как если бы он содержал цикл, можно было бы удалить одно из ребер этого цикла и вершины все еще остались бы соединенными. Следовательно, для соединения *n* вершин нужно построить (*n* – 1) ребро.

Покажем, что граф наименьшей длины можно построить, пользуясь следующим правилом. Прежде всего соединяем две вершины с наиболее коротким ребром u_1. На каждом из следующих шагов добавляем самое короткое из ребер u_i, при присоединении которого к уже имеющимся ребрам не образуется никакого цикла. Если имеется несколько ребер одинаковой длины, то выбираем любое из них.

Каждое дерево *Q*, построенное таким образом, будем называть *экономическим деревом.* Его длина равна сумме длин отдельных ребер:

$$l(Q) = l(u_1) + ... + l(u_{n-1}).$$

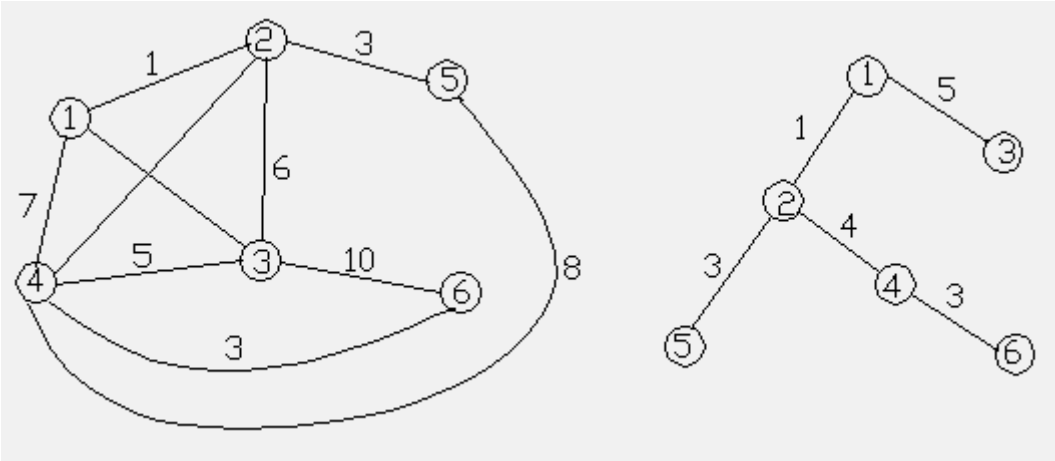
Пример построения экономического дерева приведен на рис. 5.7 (*а*, *б*).

Ниже приведена программа нахождения кратчайшего пути в графе, составленная на языке Си.

Кратчайший путь в графе:

```
#include<stdio.h>
main()
{
int m[10][10],i,j,k,n,s=0;
clrscr();
puts(«Введите количество точек n»);
scanf(«%d»,&n);
printf(«Введите расстояние между точками \n»);
printf(«если пути нет, введите 1000\n»);
for(i=1; i<=n; i++)
for(j=1; j<=n; j++)
scanf(«%d»,&m[i][j]);
for(i=1; i<=n; i++)
for(j=1; j<=n; j++)
{if(m[i][j] >= 1000)continue;
for(k=1; k<=n; k++)
{if(m[i][k] >=1000)continue;
s=m[j][i]+m[i][k];
if(s<m[j][k])m[j][k] = s;
```

```
}}
puts(«матрица кратчайших расстояний»);
for(j=1; j <= n; j++)
{for(k=1;k <= n;k++)
printf(«%d «, m[j][k]);
printf(«\n»);
}}
```



а *б*

Рис. 5.7. Построение экономического дерева

Пример

Введите количество точек *n*.

6

Введите расстояние между точками, если пути нет, введите 1000.

Исходная матрица расстояний между вершинами графа x_i и x_j

	x_j					
	0	10	22	7	1000	48
x_i	10	0	1000	5	44	30
	22	1000	0	35	6	12
	7	5	35	0	65	1000
	1000	44	6	65	0	1000
	48	30	12	1000	1000	0

Решение

Матрица кратчайших расстояний между вершинами графа x_i и x_j.

x_j

	0	10	22	7	28	34
x_i	10	0	32	5	38	30
	22	32	0	29	6	12
	7	5	29	0	35	35
	28	38	6	35	0	18
	34	30	12	35	18	0

Контрольные вопросы

1. Дайте определение следующим элементам ориентированного графа: вершина, дуга, путь, цикл, контур и др.
2. Что такое подграф?
3. Что такое частичный граф?
4. Построить матрицы смежности и инциденций для графа, изображенного на рис. 5.8.
5. Дайте определение элементам неориентированного графа.
6. Каково число ребер в полном неориентированном графе с *n* вершинами?
7. Построить матрицы смежности и инциденций для графа, изображенного на рис. 5.8, без учета ориентации дуг.

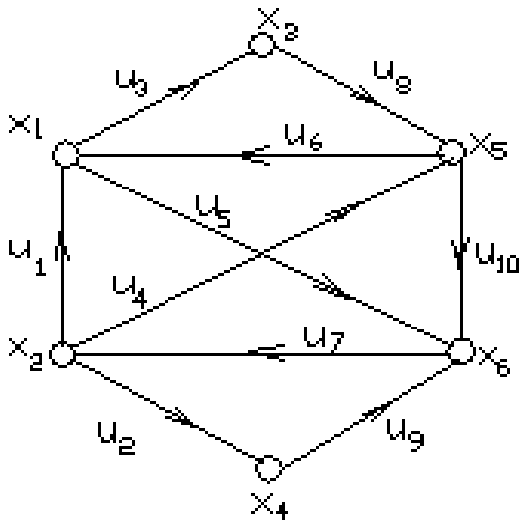


Рис. 5.8

8. Что такое экономическое дерево?
9. Сформулируйте правила построения экономического дерева.
10. Как осуществляется разметка вершин графа?

11. Сформулируйте правила нахождения кратчайшего пути в графе с ребрами единичной длины.
12. Сформулируйте правила нахождения кратчайшего пути в графе с ребрами произвольной длины.
13. Какие графы являются изоморфными?
14. Приведите примеры практических задач, которые могут быть представлены в виде графа.

ГЛАВА 6. КОМБИНАТОРНЫЕ ЗАДАЧИ НА СОСТАВЛЕНИЕ РАСПИСАНИЯ

6.1. Понятие о комбинаторных задачах

Существо комбинаторных задач проиллюстрируем на примере определения оптимальной последовательности обработки деталей.

Пусть имеется два станка, например токарный и фрезерный, и n различных деталей, каждая из которых должна быть последовательно обработана на каждом из этих станков. Требуется определить, в какой последовательности нужно обрабатывать эти детали.

Данная задача имеет характерные признаки задач на составление расписания с ограничениями на последовательность операций (сначала токарная обработка, а затем фрезерование) и с ограничениями на используемые ресурсы (имеются только один токарный и один фрезерный станки). Однако данные ограничения не являются чрезвычайно жесткими и допускают большую свободу выбора вариантов обработки деталей. Начать обработку можно с любой из n деталей, потом можно обрабатывать любую из оставшихся $(n - 1)$ деталей и т. д. Общее число вариантов

$$n\,(n - 1),\ ...,\ 1 = n!$$

Может быть, все эти варианты равноценны, так что нет никакой проблемы? Оказывается, что это не так. От выбора порядка обработки деталей зависят простои оборудования (фрезерный станок уже освободился, а токарная обработка очередной детали не закончена) и перерывы в обработке деталей (закончена токарная обработка очередной детали, но фрезерный станок еще занят).

Очевидно, из всех возможных вариантов обработки следует выбрать такой, который позволит закончить обработку всей партии деталей за кратчайшее время. Поэтому данная задача является задачей оптимизации.

Но ее решение чрезвычайно затруднено наличием огромного числа вариантов обработки, что и заставляет отнести данную задачу к задачам комбинаторного типа. Ее решение приведем позднее. Рассмотрим сначала задачу с более простыми видами ограничений, на примере которой и познакомимся с некоторыми методами решения задач подобного типа.

6.2. Задача коммивояжера

Рассмотрим более простой вариант задачи об определении оптимальной последовательности обработки деталей.

Предположим, что n различных деталей, занумерованных от 1 до n, надо обработать на одном станке. При переходе от обработки детали i к детали k требуется время d_{ik} на переналадку станка, различное при различных i и k.

Требуется определить такой порядок обработки деталей, при котором общее время на обработку будет минимальным.

Поскольку возможно n вариантов обработки первой детали, $(n-1)$ − второй и т. д., то общее число вариантов обработки будет равно $n!$ Следовательно, эта задача относится к задачам комбинаторного типа.

Поскольку время, затрачиваемое непосредственно на обработку всех деталей, будет одним и тем же при любом порядке обработки, то минимизация общего времени обработки сводится к минимизации времени на переналадку станков.

В математике задача подобного вида получила название задачи коммивояжера. Коммивояжер (агент, рекламирующий товар своей фирмы) должен посетить $(n-1)$ других городов и вернуться к исходному пункту. Известны расстояния между городами d_{ik}. Требуется установить, в каком порядке коммивояжер должен посещать города, чтобы общее пройденное расстояние было минимальным.

Если несколько изменить формулировку задачи и не требовать возвращения к исходному пункту, то получим незамкнутую задачу коммивояжера. Если же не задан и начальный пункт, то получаем полную аналогию с задачей о выборе оптимальной последовательности обработки деталей. Впрочем, эти последние задачи без труда сводятся к первоначальной, как будет показано далее.

6.3. Представление задачи коммивояжера в виде графа

Очень наглядно задачу коммивояжера можно представить в виде графа $G = (X, U)$, вершины которого соответствуют городам, а дуги − дорогам, соединяющим города между собой. Каждой дуге, соединяющей города i и k, приписывается число d_{ik}, называемое длиной дуги и равное расстоянию между городами i и k . Если из каждого города имеется непосредственный путь в любой другой город без захода в промежуточные города, то каждые две вершины графа будут соединены дугами в обоих направлениях. Получающийся при этом граф оказывается полным.

На практике полные графы встречаются редко. Если граф неполный, то его можно считать полным, мысленно восстановив недостающие дуги и приписав им длину $d = \infty$. В таблице расстояний соответствующие клетки удобно оставлять незаполненными.

Элементарный путь в графе, проходящий через все вершины, называют гамильтоновым путем. Замкнутый элементарный путь, т. е. элементарный контур, проходящий через все вершины графа, называют гамильтоновым контуром.

Как видим, задача коммивояжера сводится к нахождению кратчайшего гамильтонова контура или кратчайшего гамильтонова пути.

В дальнейшем основным видом задачи коммивояжера будем считать замкнутый вариант, когда задана вершина, в которой начинается и заканчивается маршрут коммивояжера. Однако незамкнутые варианты легко приводятся к замкнутым.

Пусть дан граф $G = (X, U)$ и дана начальная точка $x_0 \in X$, но не указана конечная точка. Заменим длины всех дуг, заканчивающихся в x_0, нулями. Тогда любой гамильтонов контур будет содержать дугу нулевой длины, заканчивающуюся в x_0. Если найти кратчайший контур и исключить из него дугу нулевой длины, оканчивающуюся в x_0, то получим кратчайший гамильтонов путь.

Если не указаны ни начальная, ни конечная вершины графа, то вводим фиктивную вершину z, которая соединяется дугами нулевой длины со всеми вершинами графа в обоих направлениях. Эту вершину принимаем за начальную и находим кратчайший гамильтонов контур. Исключив из него дугу нулевой длины, выходящую из z, и дугу нулевой длины, заходящую в z, получим кратчайший гамильтонов путь.

6.4. Примеры задач, сводящихся к задаче коммивояжера. Задача Гамильтона

На рис. 6.1 изображен додекаэдр, т. е. правильный многоугольник с 6 пятиугольными гранями и 16 вершинами.

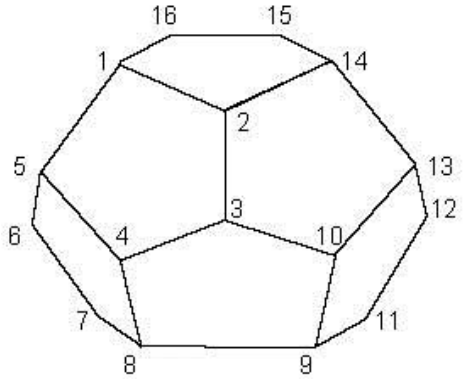


Рис. 6.1. Додекаэдр

Требуется, двигаясь по ребрам додекаэдра, обойти все вершины, заходя в каждую из них только один раз, и вернуться в исходную вершину.

Каждый такой путь, если он существует, называют гамильтоновым циклом или гамильтоновым контуром.

Для того чтобы свести эту задачу к задаче коммивояжера, будем считать, что расстояние d_{ik} между двумя вершинами равно единице, если вершины соединены ребром, и равно бесконечности в противоположном случае.

Имеется огромное число задач, связанных с объездом ряда пунктов и возвращением в исходный пункт: задачи соединения отдельных пунктов линиями электроснабжения, газоснабжения и т. п.

6.5. Определение оптимальной последовательности обработки деталей на двух станках

Вернемся к задаче об обработке деталей на двух станках. Обозначим через a_k и b_k время обработки k-й детали на первом и втором станках, соответственно. Требуется найти такую последовательность обработки деталей, при которой общее время обработки всех деталей будет наименьшим.

Отметим, прежде всего, особенности работы каждого из станков. Первый станок может начать обрабатывать следующую деталь сразу же после окончания обработки предыдущей, и простоев у него не будет.

Второй станок может начать обработку очередной детали только в том случае, когда: 1) закончилась обработка этой детали на первом станке; 2) закончилась обработка предыдущей детали на втором станке.

Рассмотрим простейший случай обработки двух деталей. Если сначала обрабатывается первая деталь, то момент окончания ее обработки на первом станке будет a_1. Моменты $a_1 + b_1$ и $a_1 + b_2$ соответствуют окончанию обработки первой детали на втором станке и второй детали на первом станке. Вторую деталь на втором станке можно начать обрабатывать тогда, когда обе эти операции закончены, т. е. в момент

$$\max(a_1 + b_1, a_1 + a_2) = a_1 + \max(a_2, b_1).$$

Обработка обеих деталей закончится в момент

$$t' = a_1 + \max(a_2, b_1) + b_2.$$

Аналогично, если сначала обрабатывается вторая деталь, то момент окончания обработки обеих деталей

$$t'' = a_2 + \max(a_1, b_2) + b_1.$$

Обработку выгодней начинать с первой детали при $t' < t''$, т. е. если

$$a_1 + \max(a_2, b_1) + b_2 <= a_2 + \max(a_1, b_2) + b_1.$$

Легко можно получить следующие соотношения:

$$a_1 + b_2 = \max(a_1, b_2) + \min(a_1, b_2);$$

$$a_2 + b_1 = \max(a_2,, b_1) + \min(a_2, b_1),$$

с учетом которых запишем:

$$\min(a_1, b_2) <= \min(a_2, b_1).$$

Это и есть условие, когда обработку выгодней начинать с первой детали.

Рассмотрение случая n деталей более сложно, но можно показать, что из деталей i и j раньше нужно начинать обработку детали i, если выполняется условие

$$\min(a_i, b_j) <= \min(a_j, b_i).$$

Если теперь поставить в соответствие каждой детали вершину графа и считать, что расстояние d_{ij} между вершинами i и j равно единице, когда выполняется условие

$$\min(a_i, b_i) <= \min(a_j, b_i),$$

и равно бесконечности в противном случае, то приходим к матрице расстояний, определяющей соответствующую задачу коммивояжера. Кратчайший гамильтонов путь, определяемый этой матрицей, и дает оптимальную последовательность обработки деталей.

6.6. Методы решения задачи коммивояжера

6.6.1. Применение метода Монте-Карло

Методом Монте-Карло называют любую статистическую процедуру, включающую в себя приемы статистической выборки. Не останавливаясь на общих идеях метода Монте-Карло, рассмотрим его применение к решению задачи коммивояжера.

Вершину 1 принимают за начальную и закладывают в урну жетоны с номерами от 2 до n. Тщательно перемешав жетоны, вытаскивают их по одному и записывают номера, например i_2, ..., i_n. При этом получают гамильтонов контур 1, i_2, ..., i_n, 1. Подсчитывают длину этого контура и запоминают ее. После этого процедуру повторяют, и если новый маршрут окажется хуже, его тотчас забывают, а если лучше, то забывают предыдущий, а новый запоминают. Проводя такую процедуру многократно, можно с высокой степенью вероятности рассчитывать на то, что удастся найти, если и не наилучший, то достаточно хороший маршрут.

6.6.2. Сведение к задаче целочисленного линейного программирования

В графе $G = (X, U)$, соответствующем задаче коммивояжера, рассмотрим некоторый гамильтонов контур $1, i_2, ..., i_n, 1$. Совокупность дуг, входящих в гамильтонов контур, можно описать в виде матрицы

$$Y = [y_{ij}] = \begin{bmatrix} y_{11} \cdots y_{1n} \\ \cdots\cdots \\ y_{n1} \cdots y_{nn} \end{bmatrix}, \tag{6.1}$$

в которой

$$y_{ij} = \begin{cases} 1, \text{ если дуга } (i, j) \text{ входит в рассматриваемый контур;} \\ \\ 0 - \text{в противоположном случае.} \end{cases} \tag{6.2}$$

Так как каждая вершина встречается в гамильтоновом контуре один и только один раз, то в каждую вершину обязательно входит и из каждой вершины обязательно выходит одна и только одна дуга. Это означает, что в матрице (6.1) в каждой строке и в каждом столбце должна стоять одна и только одна единица, что математически запишется в виде

$$\sum_{i=1}^{n} y_{ij} = 1; \qquad \sum_{j=1}^{n} y_{ij} = 1. \tag{6.3}$$

С учетом (6.3) условие, чтобы все y_{ij} были равны 0 или 1, можно заменить требованием, чтобы все y_{ij} были неотрицательными целыми числами:

$$\begin{aligned} & y_{ij} \geq 0, \ i, j = \overline{1,n}; \\ & ent[y_{ij}] = y_{ij}, \end{aligned} \tag{6.4}$$

где $ent\,[y_{ij}]$ означает операцию взятия целой части числа y_{ij}.

Как видим, любой гамильтонов контур может быть описан матрицей (6.1), элементы которой y_{ij} удовлетворяют соотношениям (6.3), (6.4). Длина гамильтонова контура равна сумме длин составляющих его дуг:

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} y_{ij}. \tag{6.5}$$

Нахождение гамильтонова контура минимальной длины сводится, таким образом, к нахождению значений переменных y_{ij}, $j = \overline{1,n}$, удовлетворяющих линейным уравнениям и неравенствам (6.3) и (6.4), условию целочисленности и обращающих в минимум линейную форму (6.5), т. е. к решению *задачи целочисленного линейного программирования.* Одним

из наиболее распространенных методов решения этой задачи является метод ветвей и границ.

6.6.3. Метод ветвей и границ

Пусть U − множество вариантов решения некоторой задачи. Если эти варианты перенумеровать от 1 до m, то под U можно понимать множество номеров вариантов решения, т. е. положить $U = \{1, ...,m\}$. Обозначим через q_i значение некоторого параметра, характеризующего *качество* решения в i-м варианте.

Оптимальным решением будет такое, *которое дает минимальное значение критерию качества*, являющегося точной нижней границей для множества $\{q_1, ..., q_m\}$.

$$q^* = \min_{i \in U} q_i. \qquad (6.6)$$

Введем понятие *оценки снизу* для множества U, под которым будем понимать такое значение $q = \hat{q}$, которое удовлетворяет соотношению

$$\hat{q} < q^* \text{ или } \hat{q} \le q^*. \qquad (6.7)$$

Если в соотношении (6.7) стоит знак нестрогого неравенства $\le$, то оценку $\hat{q}$ будем называть *достижимой*.

Метод ветвей и границ состоит в последовательном улучшении оценки $\hat{q}$ и приближении ее к оптимальному значению q^*. Предположим, что имеем возможность получить оценку снизу как для всего множества U, так и для его различных подмножеств.

Разобьем множество U на два подмножества A и B с точными нижними границами критерия качества q_A^* è q_B^*, связанными с q^* очевидным соотношением

$$q^* = \min(q_A^*, q_B^*).$$

Поскольку в множествах A и B присутствует меньшее число элементов, чем в множестве U, то имеется возможность получить оценки $\hat{q}_A$ и $\hat{q}_B$, более близкие к значениям q_A^* è q_B^*, чем в первоначальном множестве U. Это означает, что оценка $\min(\hat{q}_A, \hat{q}_B)$, будет более близка к q^*, чем оценка $\hat{q}$.

Всю процедуру разбиения удобно представить в виде дерева разбиений, приведенного на рис. 6.2.

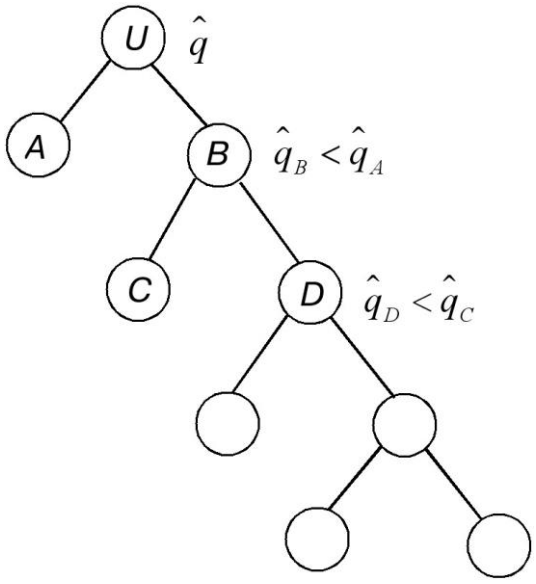


Рис. 6.2. Дерево разбиений по методу ветвей и границ

Имеется также большая вероятность того, что оптимальный вариант, т. е. вариант с наименьшим значением $q = q^*$, окажется в том из подмножеств A и B, которое имеет меньшую оценку снизу. Эта вероятность будет тем больше, чем больше отличаются друг от друга оценки $\hat{q}_A$ и $\hat{q}_{\text{в}}$, поэтому разбивать множества U на подмножества A и B желательно так, чтобы оценки снизу для этих подмножеств отличались возможно больше.

Если оказалось, что $\hat{q}_{\text{в}} < \hat{q}_A$, то есть основания предполагать, что оптимальный вариант входит в множество B и это множество надо исследовать детальней. Для этого разбиваем его, в свою очередь, на два подмножества C и D так, чтобы оценки $\hat{q}_C$ и $\hat{q}_D$ различались возможно больше. Если оказалось, что $\hat{q}_D < \hat{q}_C$, то подобному же разбиению подвергаем множество D и т. д. до тех пор, пока не придем к подмножеству, состоящему всего из одного элемента со значением параметра $q = q_0$.

Однако найденный элемент со значением $q = q_0$ может лишь подозреваться как оптимальный. Необходимо проверить, нет ли среди нерассмотренных множеств A, C и т. д. элемента со значением $q < q_0$.

При этом множества, у которых $\hat{q} > q_0$, могут не рассматриваться, поскольку элемента с $q < q_0$ в них быть не может.

Разбиение вышеуказанных множеств или приведет к элементу с $q < q_0$, тогда этот элемент и должен быть принят за оптимальный, или приведет к оценкам снизу $\hat{q} > q_0$. В этом случае элемент со значением параметра q_0 и будет искомым вариантом решения задачи.

6.7. Применение метода ветвей и границ к решению задачи коммивояжера

Считаем, что задача коммивояжера задана в виде матрицы расстояний в виде табл. 6.1.

Таблица 6.1

а

	j					
	1	2	3	4	5	
1	X	7/5	7/4	2/0	4/2	2/–
2	6/5	X	2/0	4/3	1/0	1/–
3	6/5	1/0	X	9/8	2/1	1/–
4	2/0	3/1	8/5	X	5/3	2/–
5	6/5	1/0	2/0	5/4	X	1/–

б

	j			
	2	3	4	5
1	5/3	4/2	X	2/0
2	X	0/0	3/0	0/0
3	0/0	X	8/5	1/1
5	0/0	0/0	4/1	X

в

	j		
	2	3	4
2	X	0	0
3	0	X	5
5	0	0	X

Примечание. В знаменателе даны величины для *приведенных* таблиц.

Для конкретности зададимся численными значениями расстояний, приводимыми в числителе табл. 6.1, *а*.

Для получения оценки снизу длин множества всех гамильтоновых контуров воспользуемся тем, что в каждый гамильтонов контур входит только по одному элементу из каждой строки и из каждого столбца матрицы расстояний. Поэтому, если элементы любой строки или любого столбца уменьшить на какое-либо число, то на это же число уменьшатся длины всех гамильтоновых контуров. На этом свойстве основан метод приведения матрицы расстояний.

Приведение матрицы может производиться по строкам и столбцам. Приведение матрицы расстояний по строкам заключаемся в том, что из элементов каждой строки i вычитают наименьший элемент этой строки, обозначаемый h_i. При этом длины всех гамильтоновых контуров уменьшатся на сумму вычтенных из каждой строки чисел, т. е. на величину $\sum h_i$, оставаясь в то же время неотрицательными. Поэтому величина $\sum h_i$, равная в рассматриваемом примере 7, дает некоторую оценку снизу длин всех гамильтоновых контуров.

Полученная оценка может быть улучшена путем повторного приведения матрицы расстояний по столбцам. При этом из значений элементов каждого столбца j приведенных по строкам матрицы расстояний

вычитается наименьший элемент этого столбца, обозначаемый g_j. Сумма вычтенных при этом чисел $\sum g_j = 1$. Уточненная оценка снизу длин гамильтоновых контуров

$$\hat{q} = \sum_i h_i + \sum_j g_j = 7 + 1 = 8.$$

Получаемая после приведения матрица расстояний (значения в знаменателе) дана в виде табл. 6.1, *а*.

Метод приведения матрицы расстояний будем применять и далее для получения оценок снизу различных подмножеств множества гамильтоновых контуров.

Рассмотрим теперь способ разбиения множества гамильтоновых контуров на подмножества. Возьмем некоторую дугу (i, j). К первому подмножеству отнесем все гамильтоновы контуры, в которые эта дуга входит. Обозначим это множество $[(i, j)]$. Ко второму множеству отнесем все гамильтоновы контуры, в которые дуга (i, j) не входит. Это множество обозначим $\left[\overline{(i, j)}\right]$.

Таблицу расстояний для множества $\left[(i, j)\right]$ легко получить, если учесть, что включение дуги (i, j) в гамильтонов контур исключает возможность включения других дуг, стоящих в i-й строке или j-м столбце. Следовательно, таблица расстояний для множества $[(i, j)]$ получается из первоначальной таблицы вычеркиванием i-й строки и j-го столбца.

Для того чтобы получить таблицу расстояний для множества $\left[\overline{(i, j)}\right]$, следует в первоначальной таблице запретить движение по дуге (i, j), положив ее длину, равной бесконечности. Этот запрет движения по дуге (i, j) будем отмечать знаком «х» в соответствующей клетке первоначальной таблицы.

Теперь возникает вопрос: какую дугу положить в основу разбиения множества маршрутов на подмножества?

Заметим, прежде всего, что по количеству элементов множества $\left[\overline{(i, j)}\right]$ и $\left[(i, j)\right]$ неодинаковы. Общее число гамильтоновых контуров в задаче с n городами равно $(n - 1)!$ Если зафиксировали дугу (i, j), т. е. выбрали две вершины, то имеется $(n - 2)$ способов выбрать третью, $(n - 3)$ способов – четвертую и т. д.

Следовательно, имеется $(n - 2)!$ гамильтоновых контуров, включающих дугу (i, j), а значит, $(n - 1)! - (n - 2)! = (n - 2)\,(n - 2)!$ гамильтоновых контуров, не включающих дугу (i, j). Поэтому при $n > 2$ множество $\left[\overline{(i, j)}\right]$ будет содержать большее число элементов, чем множество

$[(i, j)]$. А поскольку множество с меньшим числом элементов исследовать проще, то в качестве дуги (i, j) следует брать такую, при которой множество $[(i, j)]$ имеет меньшую оценку, чем множество $\left[\overline{(i, j)}\right]$. Из множества дуг, удовлетворяющих этому условию, следует отдать предпочтение той, которая дает наибольшую разницу в оценках для множеств $[(i, j)]$ и $\left[\overline{(i, j)}\right]$.

Если в качестве дуги (i, j) взять такую, у которой в приведенной таблице $d_{ij} \neq 0$, то для группы $[(i, j)]$ оценка снизу возрастает на d_{ij}, поскольку заранее известно, что эта дуга войдет во все гамильтоновы контуры. Она может еще возрасти, если таблица при вычеркивании i-й строки и j-го столбца будет допускать дальнейшее приведение. В то же время при замене элемента d_{ij} на бесконечность таблица расстояний останется приведенной, т. е. оценка снизу для $\left[\overline{(i, j)}\right]$ не возрастет.

Следовательно, меньшее значение оценки снизу будет для множества $\left[\overline{(i, j)}\right]$, что нежелательно.

Поэтому в качестве дуги (i, j) надо брать такую, у которой в приведенной таблице расстояний величина $d_{ij} = 0$.

Но таких дуг несколько. Какую же из них выбрать? Очевидно ту, для которой увеличение оценки для множества $\left[\overline{(i, j)}\right]$ будет наибольшим, так как при этом получится наибольшая разница в оценках для множеств $[(i, j)]$ и $\left[\overline{(i, j)}\right]$.

Обозначим увеличение оценки множества $\left[\overline{(i, j)}\right]$ через $\Delta(i, j)$.

Значение этой величины получаем путем сложения наименьших чисел в i-строке и j-м столбце.

Обратимся вновь к рассматриваемому примеру. Для табл. 6.1, *а* (знаменатель) значения $\Delta(i, j)$ следующие:

$\Delta(1, 4) = 2 + 3 = 5$; $\Delta(2, 3) = 0$;
$\Delta(2, 5) = 1$; $\Delta(3, 2) = 1$;
$\Delta(4, 1) = 6$; $\Delta(5, 2) = 0$.
$\Delta(5, 3) = 0$;

Наибольшее увеличение оценки для группы $\left[\overline{(i, j)}\right]$ получается для дуги (4, 1). Исключая четвертую строку и первый столбец из приведен-

ной матрицы, приходим к табл. 6.1, *б* (числитель), представляющий собой таблицу расстояний для множества [(4, 1)].

При получении подобных матриц нужно строго следить за тем, чтобы не могло получиться контуров, не являющихся гамильтоновыми. Поскольку дуга [(4, 1)] уже введена в намечаемый гамильтонов контур, то нужно наложить запрет на дугу [(1, 4)], отметив клеточку [(1, 4)] крестом, так как дуга [(1, 4)] совместно с дугой [(4, 1)] образует контур. В табл. 6.1, *б* (знаменатель) дана приведенная таблица расстояний с увеличением оценки $\Delta q = 2 + 3 = 5$.

Для табл. 6.1, *б* (знаменатель) значения $\Delta [(i, j)]$ следующие:

$\Delta(1, 5) = 2;$ $\Delta(2, 3) = 0;$
$\Delta(2, 4) = 1;$ $\Delta(2, 5) = 0;$
$\Delta(3, 2) = 1;$ $\Delta(5, 2) = 0;$
$\Delta(5, 3) = 0;$

Наибольшее увеличение оценки для группы $\left[\overline{(i, j)}\right]$ получается для дуги [(1, 5)], которую также вводим в намечаемый гамильтонов контур, получая цепочку [(4, 1, 5)]. Исключая первую строку и пятый столбец из матрицы табл. 6.1, *б* (знаменатель), приходим к табл. 6.1, *в* для множества [(4, 1) (1, 5)], в которой необходимо наложить запрет на дугу [(5, 4)], образующую контур с участком [(4, 1, 5)]. Полученная таблица является приведенной. Поэтому для нее сразу же находим значения $\Delta[(i, j)]$:

$\Delta(2, 3) = 0;$ $\Delta(2, 4) = 5;$
$\Delta(3, 2) = 5;$ $\Delta(5, 2) = 0;$
$\Delta(5, 3) = 0;$

Как видим, наибольшие значения $\Delta [(i, j)]$ получились равными для дуг (2, 4) и (3, 2). Поэтому обе эти дуги можно отнести к намечаемому гамильтонову контуру, получая, таким образом, последовательность дуг [(3, 2)], [(2, 4)], [(4, 1)], [(1, 5)], для замыкания которых не хватает лишь дуги [(5, 3)]. Добавляя эту дугу, получаем предполагаемый оптимальный гамильтонов контур [(3, 2, 4, 1, 5, 3)] длиной 13.

Анализируя не рассмотренные ранее подмножества $\left[\overline{(4, 1)}\right]$, $\left[(4, 1), \overline{(1, 5)}\right]$, $\left[(4, 1), (1, 5), \overline{(2, 4)}\right]$, убеждаемся, что их оценки снизу боль-

ше, чем 13, так что эти подмножества не могут содержать гамильтоновы контуры короче найденного. Следовательно, дальнейшая проверка не нужна и полученный гамильтонов контур является оптимальным.

На рис. 6.3 приведено дерево разбиений, строящееся по ходу решения задачи.

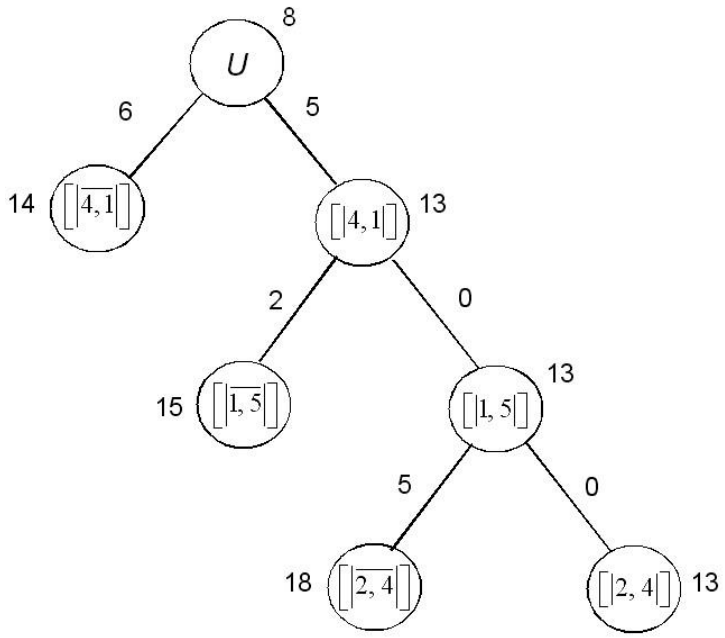


Рис. 6.3. Дерево разбиений к решению задачи коммивояжера

Более подробные сведения о методах решения комбинаторных задач содержатся в [7].

Индивидуальные задания

Тема: Определение оптимальной последовательности запуска деталей в производство.

Решить контрольный вариант в соответствии с номером индивидуального задания.

Требуется определить такой порядок обработки *n* деталей, при котором общее время обработки всех деталей будет минимальным. Значения времени на переналадку станка *tij* при переходе от *i* к *j* детали заданы в табл. 6.2.

Решение выполнить по методу ветвей и границ в соответствии с методикой, изложенной в параграфе 6.7.

Таблица 6.2

Значения времени на переналадку станка t_{ij}

№ 1	*j*	№ 2	*j*
i	• 3 10 5 9 16 8 17	*i*	• 13 6 10 13 8 9 5

	• 6 8 11 8 18 19 20 • 7 13 10 3 4 14 18 • 5 9 6 21 12 17 22 • 5 4 11 6 13 14 11 • 17 7 12 13 16 7 9 • 13 0 8 8 10 12 17		• 9 7 11 8 12 11 10 • 5 8 12 4 18 4 11 • 6 9 8 5 8 5 12 • 4 4 4 5 6 6 9 • 8 7 4 7 3 8 7 • 5 5 8 12 13 9 14

Окончание таблицы 6.2

№ 3	*j*	№ 4	*j*
i	• 10 4 5 12 3 6 14 • 7 1 9 4 11 2 10 • 4 12 10 6 8 14 18 • 5 4 6 1 7 8 6 • 8 9 9 7 10 13 15 • 7 4 10 13 16 7 9 • 13 0 8 8 10 12 17	*i*	• 1 17 1 4 5 18 2 • 21 5 6 8 7 6 16 • 9 10 9 14 10 15 8 • 14 2 3 13 6 7 15 • 10 8 12 1 11 2 4 • 6 7 1 10 3 8 6 • 3 19 4 12 13 20 4
№ 5	*j*	№ 6	*j*
i	• 9 12 8 5 13 16 17 • 1 2 6 8 7 3 4 • 6 13 16 3 4 14 10 • 4 5 9 20 11 19 15 • 10 1 18 2 13 17 11 • 17 7 12 13 16 7 9 • 18 5 6 4 10 14 15	*i*	• 2 6 12 10 8 4 9 • 4 7 10 8 17 14 10 • 6 3 12 7 19 5 13 • 5 4 3 6 10 4 5 • 9 10 8 9 5 13 6 • 6 7 11 12 7 8 9 • 12 4 5 6 2 5 4
№ 7	*j*	№ 8	*j*
i	• 1 5 7 10 2 3 4 • 8 2 5 4 7 10 1 • 7 3 10 17 8 2 3 • 5 9 6 10 1 3 7 • 4 8 12 5 4 5 6 • 10 15 1 2 5 6 7 • 8 7 12 6 18 5 4	*i*	• 5 1 4 2 10 6 7 • 4 5 10 4 5 8 10 • 15 12 14 18 4 5 7 • 4 8 9 10 12 13 14 • 5 4 7 8 9 10 5 • 7 8 4 3 5 6 7 • 9 10 5 8 11 4 3
№ 9	*j*	№ 10	*j*
i	• 3 5 10 7 10 8 12 • 4 6 12 8 14 9 7 • 12 13 11 6 7 8 9 • 10 4 5 8 12 10 16 • 8 7 9 5 6 4 11 • 1 3 12 1 4 5 6 • 4 10 11 13 15 16 8	*i*	• 20 5 12 13 4 3 8 • 9 10 11 12 13 14 15 • 8 4 5 4 6 7 8 • 10 5 7 3 4 5 4 • 3 12 13 4 6 7 8 • 9 4 8 9 6 5 4 • 11 4 3 15 4 3 14
№ 11	*j*	№ 12	*j*
i	• 8 15 4 3 1 12 13	*i*	• 1 5 2 10 3 12 4

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

Контрольные вопросы

1. Сформулируйте содержательную постановку задачи комбинаторного типа на составление расписания.
2. Какие ограничения накладываются на условия решения задач данного типа?
3. Сформулируйте целевую функцию оптимизации решения задачи комбинаторного типа.
4. Рассмотрите пример определения оптимальной последовательности обработки деталей.
5. Приведите содержательную постановку задачи коммивояжера.
6. Представьте формализованное описание задачи коммивояжера в виде графа.
7. Выполните математическую постановку задачи коммивояжера. Рассмотрите сведение ее к задаче целочисленного линейного программирования.
8. Назовите методы решения задачи коммивояжера.
9. Каковы особенности метода Монте-Карло?
10. Рассмотрите принципы ветвления и построения дерева разбиений в методе ветвей и границ.
11. Приведите примеры задач, сводящихся к задаче коммивояжера.

СПИСОК ЛИТЕРАТУРЫ

1. Перегудов Ф.И., Тарасенко Ф.П. Основы системного анализа: учебник. – 2-е изд., доп. – Томск: Изд-во НТЛ, 2001. – 396 с.
2. Кориков А.М., Сафьянова Е.Н. Основы системного анализа и теории систем: учебное пособие. – Томск: Изд-во Том. ун-та, 1989. – 207 с.
3. Советов Б.Я., Яковлев С.А. Моделирование систем: учебник для вузов. – 5-е изд., перераб. и доп. – М.: Высшая школа, 2009. – 343 с.
4. Коршунов Ю.М. Математические основы кибернетики: учебное пособие. – 3-е изд. перераб. и доп. – М: Энергоатомиздат, 1987. – 494 с.
5. Заварыкин В.М., Житомирский В.Г., Лапчик М.П. Численные методы: учеб. пособие. – М.: Просвещение, 1991. – 174 с.
6. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран, Паскаль. – Томск: МП «РАСКО», 1991. – 272 с.
7. Танаев В.С., Шкурба В.В. Введение в теорию расписаний. – М.: Наука, 1975. – 256 с.
8. Карманов В.Г. Математическое программирование / В.Г. Карманов. – 6-е изд. испр. – М.: Физматлит, 2008. – 264 с.
9. Моисеев Н.Н. Методы оптимизации: учебное пособие / Н.Н. Моисеев, Ю.П. Иванилов, Е.М. Столярова. – М.: Наука, 1978. – 351 с.
10. Вентцель Е.С. Исследование операций: задачи, принципы, методология / Е.С. Вентцель. – М.: Высшая школа, 2001. – 208 с.
11. Системы автоматизированного проектирования технологических процессов, приспособлений и режущих инструментов: учебное пособие / под ред. С.Н. Корчака. – М.: Машиностроение, 1988. – 351 с.

Учебное издание

БОГОЛЮБОВА Мария Никитична

СИСТЕМНЫЙ АНАЛИЗ И МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ В МАШИНОСТРОЕНИИ

Учебное пособие

Научный редактор *кандидат технических наук*,
доцент В.Ф. Скворцов

Выпускающий редактор *Т.С. Савенкова*
Редактор *Д.В. Заремба*
Компьютерная верстка *В.П. Аршинова*
Дизайн обложки *Т.А. Фатеева*

Подписано к печати 00.11.2010. Формат 60x84/16. Бумага «Снегурочка».
Печать XEROX. Усл. печ. л. 7.15. Уч.-изд. л. 6.47.
Заказ 0000-10. Тираж 100 экз.

Национальный исследовательский Томский политехнический университет
Система менеджмента качества
Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2008

ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30
Тел./факс: 8(3822)56-35-35, www.tpu.ru