Ю. В. ЛИТОВКА, Н. В. МАЙСТРЕНКО, С. Я. ЕГОРОВ

МАТЕМАТИЧЕСКИЕ МЕТОДЫ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Тамбов
Издательский центр ФГБОУ ВО «ТГТУ»
2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Тамбовский государственный технический университет»

Ю. В. ЛИТОВКА, Н. В. МАЙСТРЕНКО, С. Я. ЕГОРОВ

МАТЕМАТИЧЕСКИЕ МЕТОДЫ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Утверждено Учёным советом университета в качестве учебного пособия
для магистрантов 1 курса, обучающихся по направлению подготовки
09.04.01 «Информатика и вычислительная техника»,
всех форм обучения

Учебное электронное издание

Тамбов
Издательский центр ФГБОУ ВО «ТГТУ»
2023

УДК 51(075.8)
ББК В193я73л646
Л64

Рецензенты:

Доктор физико-математических наук, профессор,
директор научно-исследовательского института
математики, физики и информатики,
профессор кафедры функционального анализа
ФГБОУ ВО «ТГУ им. Г. Р. Державина»
Е. С. Жуковский

Доктор технических наук, профессор, профессор кафедры
«Компьютерно-интегрированные системы в машиностроении»
ФГБОУ ВО «ТГТУ»
С. В. Карпушкин

Л64 **Литовка, Ю. В.**
Математические методы исследования операций [Электронный ресурс] : учебное пособие / Ю. В. Литовка, Н. В. Майстренко, С. Я. Егоров. – Тамбов : Издательский центр ФГБОУ ВО «ТГТУ», 2023. – 1 электрон. опт. диск (CD-ROM). – Системные требования : ПК не ниже класса Pentium II ; CD-ROM-дисковод ; 978 Kb ; RAM ; Windows 95/98/XP ; мышь. – Загл. с экрана.
ISBN 978-5-8265-2569-2

Рассмотрены математические методы исследования операций, а именно линейного, нелинейного, геометрического, целочисленного, динамического, стохастического программирования; теории графов, теории игр и др.

Предназначено для магистрантов 1 курса, обучающихся по направлению подготовки 09.04.01 «Информатика и вычислительная техника», всех форм обучения.

УДК 51(075.8)
ББК В193я73л646

ISBN 978-5-8265-2569-2

ВВЕДЕНИЕ

Под названием «Исследование операций» подразумевается применение математических, количественных методов для обоснования решений во всех областях целенаправленной человеческой деятельности.

Возникновение науки «Исследование операций» обычно относят к годам Второй мировой войны, когда в вооружённых силах США и Англии были сформированы специальные научные группы для подготовки решений по способам организации и обеспечения боевых действий. Справедливости ради надо отметить, что подобными исследованиями (правда, не под таким названием) занимались и до войны, в частности, в нашей стране, где были широко развиты математические методы оценки эффективности стрельбы, представляющие собой, в современном понимании, часть исследования операций.

Зародившись в области преимущественно военных задач, исследование операций с течением времени вышло из этой узкой сферы. В настоящее время исследование операций – одна из самых быстро развивающихся наук, завоевывающая всё более обширные области применения: промышленность, сельское хозяйство, торговля, транспорт, здравоохранение и т.д.

1. ОСНОВНЫЕ ПОНЯТИЯ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Под операцией понимается любое мероприятие (или система действий), объединённое единым замыслом и направленное к достижению определённой цели.

Операция всегда является управляемым мероприятием, т.е. от нас зависит выбрать тем или другим способом какие-то параметры, характеризующие способ её организации. «Организация» здесь понимается в широком смысле слова, включая и выбор технических средств, применяемых в операции.

Всякий определённый выбор зависящих от нас параметров мы будем называть решением. Решения могут быть удачными и неудачными, разумными и неразумными.

Оптимальными называются решения, которые по тем или иным соображениям предпочтительнее других.

Основная задача исследования операций – предварительное количественное обоснование оптимальных решений. Задача исследования операций – подготовить количественные данные и рекомендации, облегчающие человеку принятие решения.

Наряду с основной задачей – обоснованием оптимальных решений – к области исследования операций относятся и другие задачи:

– сравнительная оценка различных вариантов организации операции;

– оценка влияния на результат операции различных параметров (элементов решения и заданных условий);

– исследование так называемых «узких мест», т.е. элементов управляемой системы, нарушение работы которых особенно сильно сказывается на успехе операции и т.д.

Рассмотрим отдельную операцию *О*. Размышляя над организацией операции, мы стремимся сделать её наиболее эффективной. Под эффективностью операции подразумевается степень её приспособленности к выполнению стоящей перед ней задачи.

Чем лучше организована операция, тем она эффективнее. Чтобы судить об эффективности операции и сравнивать между собой по эффективности различно организованные операции, нужно иметь некоторый численный критерий оценки или показатель эффективности (в некоторых руководствах показатель эффективности называют «целевой функцией»).

Будем в дальнейшем обозначать показатель эффективности буквой *R*.

Конкретный вид показателя эффективности *R*, которым следует пользоваться при численной оценке эффективности, зависит от специфики рассматриваемой операции, её целевой направленности, а также от задачи исследования, которая может быть поставлена в той или другой форме.

Многие операции выполняются в условиях, содержащих элемент случайности (например, операции, связанные с колебаниями спроса и предложения, с движением народонаселения, заболеваемостью, смертностью, а также все военные операции). В этих случаях исход операции, даже организованной строго определённым образом, не может быть точно предсказан, остаётся случайным. Если это так, то в качестве показателя эффективности *R* выбирается не просто характеристика исхода операции, а её среднее значение (математическое ожидание). Например, если задача операции – получение максимальной прибыли, то в качестве показателя эффективности берётся средняя прибыль.

В других случаях, когда задачей операции является осуществление вполне определённого события, в качестве показателя эффективности берут вероятность этого события (например, вероятность того, что в результате воздушного налёта данная группа целей будет поражена).

Правильный выбор показателя эффективности – необходимое условие полезности исследования, применяемого для обоснования решения.

2. МЕТОДИКА ПРОВЕДЕНИЯ ИССЛЕДОВАНИЙ ОПЕРАЦИЙ

Исследование операций включает следующие этапы [1].

Этап 1. *Определение целей*. Первоочередная цель любого исследования операции заключается в том, чтобы выяснять, что ожидает получить в результате его проведения руководитель или пользователь. Другими словами, требуется определить, каковы предполагаемые результаты завершения проекта.

Цели исследования следует формулировать, исходя из сущности решения или решений, на получение которых ориентирована данная работа. Специалисту по исследованию операций следует особо позаботиться о том, чтобы формулировка цели работы не была слишком узкой. Нельзя ставить и излишне широкие цели, что обычно приводит к безуспешной попытке сразу решить все проблемы в рамках одного всеобъемлющего исследования.

Этап 2. *Составление плана разработки проекта*. Под планированием подразумевается не выбор методов выполнения тех или иных работ, а установление реперных точек, т.е. временных координат, соответствующих необходимым срокам завершения определённых видов работ. Как документ, регламентирующий процесс исследования, план разработки проекта представляет собой просто календарный график выполнения его этапов. План разработки проекта может быть представлен в виде диаграммы Ганта (рис. 2.1).

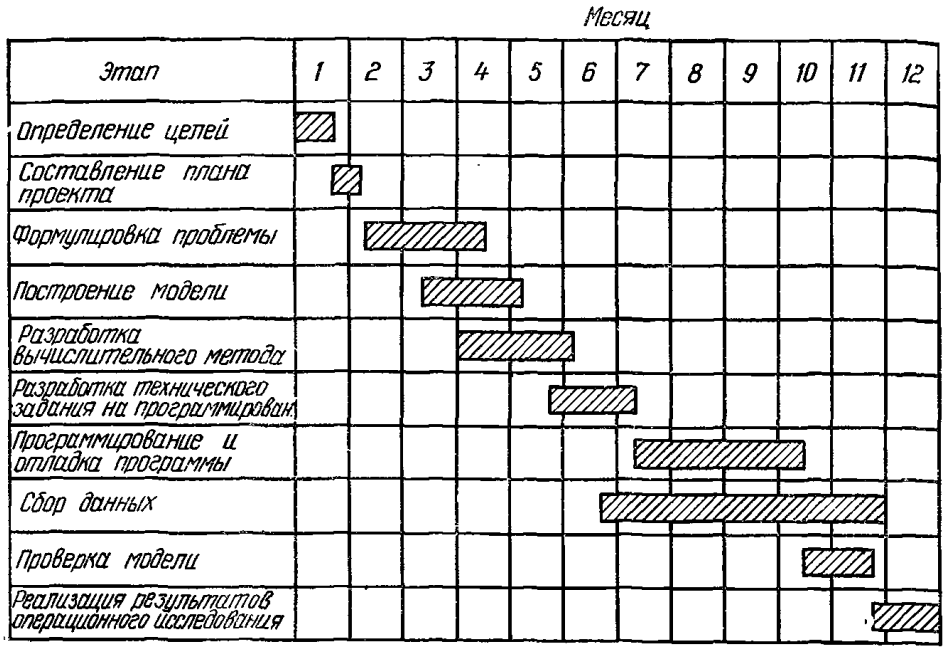

Рис. 2.1. План проекта в виде диаграммы Ганта

Этапы проекта во многих случаях детализируются до уровня отдельных заданий. Например, этап, связанный с разработкой вычислительного метода, может включать следующие задания:

1) разработать соответствующий метод решения для каждой подмодели (частной модели) рассматриваемой задачи;

2) описать и документально оформить методы решения. В итоговом отчёте для каждой модели следует привести описание входной и выходной информации, а также процедур расчёта, необходимых для получения решения. Процедуры расчёта следует представить в аналитической форме и привести их словесное описание и требуемые пояснения;

3) проверить предлагаемые методы решения на выбранных задачах небольшой размерности. Этот этап должен подтвердить возможность реализации предлагаемых методов решения и, кроме того, обеспечить точность документального оформления модели, прежде чем она будет переведена в машинную программу;

4) внести необходимые уточнения и изменения в методы решения на основании результатов пробных расчётов, выполненных вручную;

5) подготовить краткое изложение результатов испытаний, сопроводив его документально оформленными методами решения;

6) подготовить и представить в надлежащей форме отчёт о результатах данного этапа исследования.

Детальная разработка плана, рассмотренная выше, нередко оказывается очень важной для обоснованной оценки времени и ресурсов, требуемых для выполнения проекта. Разбиение всего исследования на отдельные последовательные этапы – лучший способ надёжной оценки необходимых финансовых, людских и временных ресурсов ещё до начала работы над проектом. Другими словами, составление плана разработки проекта увеличивает вероятность того, что исследование будет завершено в соответствии с графиком работ и в пределах выделенного финансирования.

При составлении плана во многих случаях следует уделить внимание распределению работ по отдельным исполнителям.

Этап 3. *Формулировка проблемы*. Этот этап является одним из самых важных этапов. Первый из вопросов, связанных с формулировкой проблемы, нередко заключается в том, чтобы определить, можно ли представить всю проблему в виде отдельных, более частных подпроблем, с тем, чтобы параллельно

или последовательно исследовать их независимо одну от другой. Это особенно часто необходимо в тех случаях, когда рассматриваемая проблема охватывает широкую н сложную сферу деятельности организации. Процесс логического разделения большой проблемы на отдельные моменты, конечно, приводит в итоге к получению субоптимального решения. Однако отклонение от оптимума может быть минимальным, если проблему разделить таким образом, чтобы выделенные подпроблемы по возможности не пересекались.

Второй вопрос, подлежащий решению на стадии формулировки проблемы, связан с определением степени детализации разрабатываемой модели. Основные факторы, влияющие на его решение, – это объём выделенных средств, календарный план разработки проекта и цели исследования.

После разделения проблемы на соответствующие части и определения желаемой степени детализации модели процесс формулировки проблемы может быть продолжен. Следующая фаза этого процесса связана с определением области применения и размерности разрабатываемой модели. При рассмотрении этой фазы мы выделим пять вопросов.

3.1. *Определение размерности задачи*. Первый этап процесса построения модели связан с определением переменных исследуемой задачи, которые необходимо учитывать. Размерность задачи часто принимают в качестве заранее заданного условия без какой-либо аргументации.

3.2. *Определение управляющих переменных.* Этот этап, являющийся вторым этапом построения модели, заключается в определении переменных, которые могут изменяться управляющим органом. В процессе формулировки задачи нужно точно определить эти переменные, чтобы разделить все учитываемые параметры на заданные и такие, которые можно рассматривать в качестве управляющих.

3.3. *Определение неуправляемых переменных*. Третий вопрос, подлежащий решению на стадии формулировки проблемы, связан с определением неуправляемых переменных, т.е. таких параметров, которые не могут быть изменены управляющим органом, но оказывают влияние на моделируемую деятельность.

3.4. *Определение технологических параметров системы*. Определение технологических параметров является четвёртым вопросом, который нужно решать па стадии формулировки проблемы. Технология описывается совокупностью констант и параметров, которые определяют предельные значения

переменных и соотношения между ними. Часто приходится моделировать сложные технологические процессы.

3.5. *Определение показателей эффективности*. Показатели эффективности служат основой для оценки конкретных решений рассматриваемой проблемы. В большинстве случаев при исследовании операций используется несколько показателей эффективности. Наиболее важные из них следует выявить уже на стадии определения целей исследования.

Мы можем записать общую задачу исследования операций следующим образом:

$$R(x, y, z) \to \min; \quad g_i(x, y, z) \le 0, \quad i = 1, \dots, m; \quad x \in X, \quad y \in Y, \quad z \in Z,$$

где x – вектор контролируемых факторов; y – вектор случайных факторов; z – вектор неопределённых факторов; X, Y, Z – подмножества некоторых векторных пространств.

Значения контролируемых факторов выбираются теми, кто принимает решение (оперирующей стороной). Случайные и неопределённые факторы – это неконтролируемые факторы для оперирующей стороны. Разница между случайными и неопределёнными факторами состоит в следующем. Вектор y – это случайный вектор с известным законом распределения. Например, y_5 есть нормальная случайная величина с математическим ожиданием $m \in [m_1, m_2]$ и стандартным отклонением $\sigma \in [\sigma_1, \sigma_2]$. В противоположность оперирующей стороне известны только области значений неопределённых факторов.

Этап 4. *Построение модели*. Данный этап исследования связан с разработкой модели. Модель выражает взаимосвязь между управляющими переменными, неуправляемыми переменными, технологическими параметрами и показателями эффективности. Правильное построение модели – основное условие успешной разработки проекта.

Математические модели могут быть построены аналитическим, экспериментальным и экспериментально-аналитическим методами.

Результатом данного этапа будет математическая постановка оптимизационной задачи.

Этап 5. *Разработка вычислительного метода*. Одновременно с проведением работ по построению модели, необходимо выбрать или разработать численный метод решения оптимизационной задачи. Здесь можно применить известные методы: линейного, нелинейного, целочисленного, динамического программирования или разработать новые (как правило, эвристические) методы.

Этап 6. *Разработка технического задания па программирование*; *программирование и отладка*. Составление программ для ЭВМ является составной частью проводимого исследования. Разработка технического задания на программирование для многих специалистов является неприятной задачей. Однако, только тщательное и аккуратное составление этих заданий обеспечивает должный контроль за качеством последующих работ по программированию и затратами на их выполнение. Тщательная разработка указанных технических заданий обеспечивает также более качественное документальное оформление программ, в результате чего исследование становится в большей степени ориентированным на пользователя и удовлетворение его нужд.

Одним из важнейших требований, предъявляемых к программному обеспечению, является требование к интерфейсу пользователя.

На этом этапе решаются задачи собственно программирования и отладки разработанных программ.

Этап 7. *Сбор данных*. На этом этапе осуществляется сбор и анализ данных, необходимых для проверки правильности модели и практического использования результатов исследования операций.

Этап 8. *Проверка моделей*. Важность этого этапа трудно переоценить. Он включает две фазы: определение способов проверки модели и осуществление этой проверки. На первой фазе выбираются аналитические и экспериментальные методы для проверки непротиворечивости, чувствительности, реалистичности и работоспособности модели. Для осуществления проверки модели требуются данные, собранные на предыдущем этапе. Результаты такой работы нередко приводят к необходимости перестройки модели и соответственно к составлению новых программ.

8.1. *Непротиворечивость*. Даёт ли модель непротиворечащие логике результаты при вариации величин важнейших параметров, особенно в тех случаях, когда их значения близки к экстремальным? Чтобы ответить на этот вопрос, необходимо проанализировать характер реакции модели на изменения соответствующих входных параметров.

8.2. *Чувствительность*. Соответствуют ли относительные изменения выходных переменных модели небольшим изменениям её параметров? Анализ чувствительности модели обычно требует проведения численных расчётов.

8.3. *Реалистичность*. Соответствует ли модель тем частным случаям, для которых уже имеются фактические данные?

8.4. *Работоспособность*. Легко ли получить решение с помощью предлагаемой модели? Если она должна использоваться в процессе выработки решений, принимаемых руководителями линейных подразделений, то необходимо, чтобы расчёты можно было выполнить в пределах сроков, установленных для подготовки соответствующих решений. Кроме того, трудозатраты и ресурсы, требуемые для эксплуатации модели, должны укладываться в установленные лимиты машинного времени и фонда зарплаты. Нередко это вынуждает исследователя использовать приближённые методы и эвристические процедуры даже тогда, когда это не обусловлено технической необходимостью.

Этап 9. *Реализация результатов исследования операций*. Слишком часто после разработки и проверки модели специалист по исследованию операций считает свою работу законченной. Однако, это неверно. В действительности же многие трудности возникают только на конечном этапе разработки проекта, связанном с реализацией полученных результатов. Эффективность операционного проекта в целом существенно зависит от того, в какой мере удалось обеспечить при его разработке сотрудничество руководителей различных уровней управления, ответственных за решение исследуемой проблемы или выполнение функций, имеющих к ней определённое отношение.

Для успешной реализации проекта необходимы: составление материалов для ознакомления управленческого персонала с принципами функционирования внедряемой системы; подготовка и проведение соответствующих семинаров и наглядная демонстрация результатов исследования на практических примерах. Хотя к моменту завершения проекта многие руководители могут быть уже ознакомлены со всеми моделями и разработанными системами, все же весьма важно ещё раз убедиться в том, что их представления достаточно основательны и имеются нужные материалы для обучения других сотрудников. Такие меры увеличивают полезный срок службы моделей. Только при должном стремлении исследователя к взаимодействию и взаимопониманию в работе со всеми, чью деятельность он изучает и анализирует, можно обеспечить действенное сотрудничество и успешное завершение исследования операций.

3. МЕТОДЫ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

3.1. ДЕТЕРМИНИРОВАННЫЕ МЕТОДЫ

Детерминированные задачи – те, при которых считается, что каждая выбираемая стратегия приводит к единственному, заранее известному результату. В зависимости от конкретной ситуации, возможны различные постановки задач, для каждого класса которых разработаны специальные методы их решения.

3.1.1. ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Задача линейного программирования есть задача максимизации (или минимизации) линейной функции при линейных ограничениях [4 – 7].

Математическая постановка задачи: найти значения x_1^*, x_2^*, …, x_n^*, при которых

$$R = \sum_{i=1}^{n} C_i x_i^* \rightarrow \max(\min), \tag{3.1}$$

где C_i, $i = 1, 2, \ldots, n$ – заданные постоянные коэффициенты, при ограничениях:

$$\sum_{i=1}^{n} a_{ij} x_i \le b_j \quad j = 1, 2, \ldots, m_1; \tag{3.2}$$

$$\sum_{i=1}^{n} a_{ij} x_i \ge b_j, \quad j = m_1 + 1, \ldots, m_2; \tag{3.3}$$

$$\sum_{i=1}^{n} a_{ij} x_i = b_j, \quad j = m_2 + 1, \ldots, m; \tag{3.4}$$

$$x_i \ge 0, \quad i = 1, 2, \ldots, n. \tag{3.5}$$

Последнее условие – не обязательное, однако в реальных задачах x_i имеют чёткий физический смысл – количество продукции, цена, и т.д., которые не могут быть отрицательными.

Количество ограничений типа равенств (3.4) не должно превышать число независимых переменных n. Количество ограничений типа неравенств (3.2) и (3.3) может быть любым.

Геометрическая иллюстрация для функции двух переменных показана на рис. 3.1, где $R = c_1x_1 + c_2x_2$; 1 – 4 – ограничения.

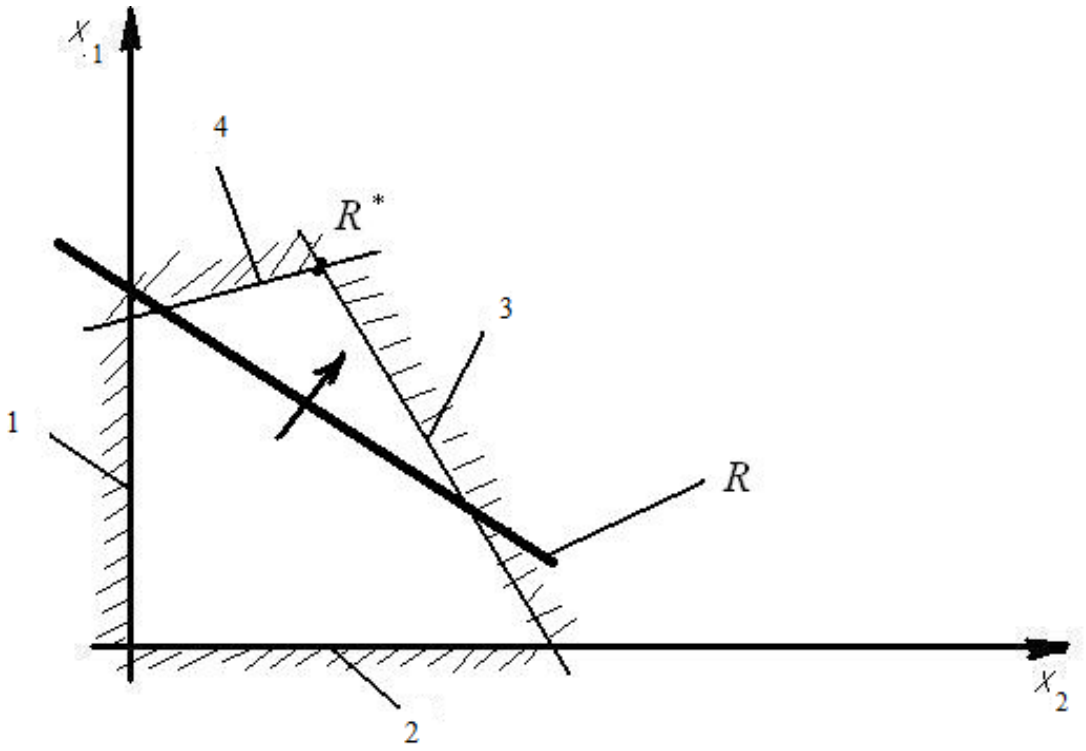

Рис. 3.1. Геометрическая иллюстрация задачи линейного программирования

В данном примере видно, что максимальное значение критерия R лежит в точке пересечения ограничений *3* и *4*.

В многомерном случае решение будет лежать либо в одной из вершин, либо на грани (гиперплоскости).

В качестве примера задачи линейного программирования приведём задачу использования ресурсов.

Для изготовления нескольких видов продукции P_1, P_2, …, P_n используют m видов ресурсов S_1, S_2, …, S_m – это могут быть различные материалы, сырьё, энергоресурсы, полуфабрикаты. Объём каждого из видов ресурсов ограничен и известен b_1, b_2, …, b_m; a_{ji} – количество единиц i-го сырья, идущего на изготовление единицы j-й продукции. Предприятие может обеспечить выпуск каждого вида продукта в количестве не более d_j единиц, где j = 1, 2, …, n. При реализации единиц j продукции прибыль составляет c_j единиц.

Необходимо составить план выпуска продукции, который обеспечивал бы получение максимальной прибыли при реализации выпускаемой продукции.

Составим математическую модель задачи линейного программирования.

Пусть x_j, j = 1, 2, …, n – количество единиц j-й продукции, которое необходимо выпустить. Тогда целевая функция имеет вид:

$$R = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow \max$$

при ограничениях:

$$\sum_{i=1}^{n} a_{ij} x_i \le b_j , \quad j = 1, 2, \ldots, m;$$

$$0 \le x_j \le d_j, \qquad j = 1, 2, \ldots, n.$$

Для решения задачи линейного программирования используют методы полного перебора и симплексный [4, 5].

а) *Метод полного перебора*. Согласно данному методу требуется найти все вершины многогранника, для чего решить все возможные комбинации по n уравнений из общего числа $m + n$ уравнений системы (3.2) − (3.5). При больших значениях n и m число таких комбинаций может быть настолько велико, что поиск решения потребует значительного времени расчёта.

б) *Симплексный метод*. Идея метода заключается в поиске по одному известному решению (любому) другого, при котором значение критерия оптимальности станет больше.

Предварительно все ограничения типа неравенств приводятся к равенствам введением m новых, дополнительных переменных. Для этого в каждом соотношении (3.2) прибавим к левой части дополнительную положительную переменную x_{n+j}, которая превращает неравенство в равенство:

$$\sum_{i=1}^{n} a_{ij} x_i + x_{n+j} = b_j , \quad j = 1, 2, \ldots, m_1.$$

В каждом соотношении (3.3) отнимем от левой части дополнительную положительную переменную:

$$\sum_{i=1}^{n} a_{ij} x_i - x_{n+j} = b_j , \quad j = m_1 + 1, \ldots, m_2.$$

Тогда система ограничений (3.2) − (3.4) получит вид

$$\sum_{i=1}^{n+m} a_{ij} x_i = b_j \quad j = 1, 2, \ldots, m. \tag{3.6}$$

Любое решение системы уравнений (3.6), состоящее из набора $n + m$ неотрицательных значений переменных x_j ($j = 1, 2, \ldots, n + m$) называется допустимым.

Допустимое решение, в котором ровно n составляющих отличны от 0, называется базисным или планом.

Базисное решение определяет координаты вершины многогранника условий оптимальной задачи, в то время как допустимое решение может определять координаты любой другой точки этого многогранника, включая и его внутренние точки.

Оптимальное решение всегда должно быть базисным, поэтому его поиск заключается в переборе только базисных решений.

Рассмотрим алгоритм решения задачи линейного программирования симплекс-методом.

Необходимо найти значения x_1^*, x_2^*, …, x_n^*, при которых

$$f_0 = C_1x_1 + C_2x_2 + \ldots + C_nx_n \rightarrow \min$$

при ограничениях:

$$n_1 \text{ штук} = \left\{\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1 \\ ---------- \\ a_{n1,1}x_1 + a_{n1,2}x_2 + \ldots + a_{n1.n}x_n \le b_{n1} \end{array}\right\}; \quad (3.7)$$

$$n_2 \text{ штук} = \left\{\begin{array}{l} h_{11}x_1 + h_{12}x_2 + \ldots + h_{1n}x_n = b_{n1+1} \\ ---------- \\ h_{n2,1}x_1 + h_{n2,2}x_2 + \ldots + h_{n2,n}x_n = b_{n1+n2} \end{array}\right\}; \quad (3.8)$$

$$n_3 \text{ штук} = \left\{\begin{array}{l} h_{n2+1,1}x_1 + h_{n2+1,2}x_2 + \ldots + h_{n2+1,n}x_n \ge b_{n1+n2+1} \\ ---------- \\ h_{n2+n3,1}x_1 + h_{n2+n3,2}x_2 + \ldots + h_{n2+n3,n}x_n \ge b_{n1+n2+n3} \end{array}\right\}. \quad (3.9)$$

Алгоритм симплекс-метода состоит из следующих этапов.

Предварительно преобразуем ограничения.

1. Добавляем к ограничениям (3.9) в левую часть дополнительные переменные x_{n+1}, …, x_{n+n_3} с коэффициентом «–1».

2. Добавляем к ограничениям (3.7) в левую часть дополнительные переменные x_{n+n_3+1}, …, $x_{n+n_3+n_1}$ с коэффициентом «+1».

Первые два этапа нужны для преобразования неравенств в равенства.

3. Добавляем к ограничениям (3.8) в левую часть вспомогательные переменные $x_{n+n_3+n_1+1}$, …, $x_{n+n_3+n_1+n_2}$ с коэффициентом «+1».

4. Добавляем к ограничениям (3.9) в левую часть вспомогательные переменные $x_{n+n_3+n_1+n_2+1}$,…, $x_{n+n_3+n_1+n_2+n_3}$ с коэффициентом «+1».

Последние два этапа нужны для формирования единичной диагональной матрицы.

Начальное допустимое базисное решение имеет вид:

$x_i = 0,\ i = 1, 2, \ldots, n + n_3$ − небазисные переменные;

$x_i = b_{i-n-n_3},\ i = n + n_3 + 1, \ldots, n + n_3 + n_2 + n_1$ − базисные переменные.

Введём искусственную целевую функцию W, которая является суммой небазисных переменных:

$$-\sum_{i=1}^{n_2+n_3} h_{i,1}x_1 - \sum_{i=1}^{n_2+n_3} h_{i,2}x_2 - \ldots - \sum_{i=1}^{n_2+n_3} h_{i,n+n_3}x_{n+n_3} = W - \sum_{i=n_1+1}^{n_1+n_2+n_3} b_i \qquad (3.10)$$

или

$$-d_1x_1 - d_2x_2 - \ldots - d_{n+n_3}x_{n+n_3} = W - W_0\,.$$

Далее необходимо найти минимум вспомогательной функции W, для чего строятся симплекс-таблицы и делается необходимое число итераций.

Полученное решение является исходным базовым для исходной задачи.

Приведём пример работы алгоритма с использованием симплекс-таблицы для $n = 3$, $n_1 = 1$, $n_2 = 1$, $n_3 = 2$.

Необходимо найти значения $x_1^*,\ x_2^*,\ x_3^*$, при которых

$$f_0 = C_1x_1 + C_2x_2 + C_3x_3 \to \min$$

при ограничениях:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \le b_1;$$

$$h_{11}x_1 + h_{12}x_2 + h_{13}x_3 = b_2;$$

$$h_{21}x_1 + h_{22}x_2 + h_{23}x_3 \ge b_3;$$

$$h_{31}x_1 + h_{32}x_2 + h_{33}x_3 \ge b_4.$$

После преобразований (3.1) – (3.4) получаем:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + x_6 = b_1;$$

$$h_{11}x_1 + h_{12}x_2 + h_{13}x_3 + x_7 = b_2;$$

$$h_{21}x_1 + h_{22}x_2 + h_{23}x_3 - x_4 + x_8 = b_3;$$

$$h_{31}x_1 + h_{32}x_2 + h_{33}x_3 - x_5 + x_9 = b_4.$$

Построим симплекс-таблицу первой итерации.

Если все коэффициенты вспомогательной функции W положительные, то функция W не может быть уменьшена при увеличении x_i от 0 в положительную сторону. Таким образом, минимум найден.

Если все коэффициенты отрицательные, то ищется наименьший среди них и начинаем с ним работать, так как если это d_i, то увеличение x_i от 0 в положительную сторону приведёт к самому сильному уменьшению W. Поэтому x_i включается в базисные переменные.

Итерация 1

Базис	Значения	Коэффициенты при:								
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_6	b_1	a_{11}	a_{12}	a_{13}	0	0	1	0	0	0
x_7	b_2	h_{11}	h_{12}	h_{13}	0	0	0	1	0	0
x_8	b_3	h_{21}	h_{22}	h_{23}	–1	0	0	0	1	0
x_9	b_4	h_{31}	h_{32}	h_{33}	0	–1	0	0	0	1
$-f_0$	0	C_1	C_2	C_3	0	0	0	0	0	0
$-W$	$\sum_{i=2}^{4} b_i$	$-\sum_{i=1}^{3} h_{i1}$	$-\sum_{i=1}^{3} h_{i2}$	$-\sum_{i=1}^{3} h_{i3}$	1	1	0	0	0	0
	↓	↓	↓	↓		↓	↓			
	W_0	$-d_1$	$-d_2$	$-d_3$		$-d_4$	$-d_5$			

Пусть в рассматриваемом примере самым маленьким отрицательным коэффициентом в функции W будет d_2, тогда в базисные будем включать x_2.

Найдём, до какого максимального значения может измениться x_2 из выражения

$$x_2^{\max} = \min\{b_i/h_{i-1,2}\}; \quad i = n_1 + 1, 2, ..., n_1 + n_2 + n_3,$$

где $h > 0$.

В нашем примере

$$x_2^{\max} = \min\{b_2/h_{12}, b_3/h_{22}, b_4/h_{32}\}.$$

Пусть минимум достигается при b_4/h_{32}. Тогда базисная переменная этой строки (в примере это x_9) исключается из базисных, на её место ставится x_2.

Для построения новой симплекс-таблицы (итерация 2) проделаем следующие преобразования.

В строке новой базисной переменной (x_2) поделим все коэффициенты на h_{32}. Во всех остальных строках из каждого коэффициента вычитается значение коэффициента на пересечении данной строки и выбранного базового столбца (в нашем примере столбец при x_2), умноженного на коэффициент новой базовой строки (опять x_2), стоящий в том же столбце, что и исходный коэффициент.

После решения вспомогательной задачи последняя симплекс-таблица используется как начальная для решения исходной задачи, т.е. работаем уже с функцией f_0, а не W.

3.1.2. ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ И КОМБИНАТОРИКА

Постановка задачи целочисленного программирования: найти x_1^*, x_2^*, …, x_n^*, при которых

$$f_0(x_1^*, x_2^*, ..., x_n^*) \to \min$$

при условиях:

$$f_i(X) \geq 0, \ \ i = 1, \ 2, \ ..., \ m.$$

Дополнительные условия:

а) x_i – целые, $I = 1, 2, \ldots, q$, $q < n$ – задача частично целочисленного программирования;

б) x_i – целые, $I = 1, 2, \ldots, n$ – задача полностью целочисленного программирования;

в) x_i – целые, $I = 1, 2, \ldots, n$ и могут принимать только значения 0 или 1 – задача бивалентного программирования.

Заметим, что целевая функция $f_0(X)$ может принимать любые (не целые) значения.

В общем случае поставленную задачу нельзя решить обычными методами, отменив условия (*а*) – (*в*), с последующим округлением компонент найденного решения до ближайших целых чисел.

Пример: задача о рюкзаке. Необходимо выбрать, какие предметы взять с собой (положить в рюкзак). Каждый предмет имеет свой вес и свою ценность.

Требуется, чтобы общий вес рюкзака с выбранными предметами не превышал некоторой заданной величины, а их общая ценность была максимальной.

Введём обозначения: a_j – вес одного предмета j-го типа; c_j – его ценность; x_j – число выбранных предметов j-го типа (которые попали в рюкзак); b – заданная величина, ограничивающая вес. Тогда математическая постановка задачи примет вид: найти x_j^* , $j = 1, 2, \ldots, n$, при которых

$$R = \sum_{j=1}^{n} c_j x_j^* \to \max$$

при ограничениях

$$\sum_{j=1}^{n} a_j x_j \le b,$$

где $x_j \ge 0$, x_j – целые, $j = 1, 2, \ldots, n$.

Для решения задачи целочисленного программирования используют методы полного перебора, ветвей и границ, случайного поиска, Гомори [4, 6].

Полный перебор всех возможных целочисленных вариантов варьируемых переменных x_i даёт гарантированное решение задачи. В то же время, при большой размерности задачи время расчёта может быть большим вследствие перебора огромного количества вариантов. Однако, поскольку расчёты критерия для каждого из возможных вариантов никак не связаны друг с другом, возможно применение параллельных вычислений с использованием кластера вычислительных устройств, видеокарт и облачных вычислений. Например, современные видеокарты оснащены более, чем 10 000 вычислительными блоками, что позволяет ускорять вычисления в сотни раз.

Основная идея метода *ветвей и границ* заключается в замене полного перебора множества вариантов решения сокращённым. Полного перебора удаётся избежать за счёт отбрасывания неперспективных множеств вариантов, т.е. таких, которые заведомо не могут содержать искомого оптимального решения задачи. Эта идея реализуется путём последовательного разбиения всего множества допустимых решений задачи на подмножества и построения оценок, позволяющих сделать обоснованный вывод о том, какие из полученных подмножеств не содержат допустимых решений, а какие – оптимальных. Исключение из дальнейшего рассмотрения таких бесперспективных подмножеств даёт возможность сокращать перебор вариантов решения задачи. Однако не исключена

ситуация, когда отсекается ветвь, в конечном итоге приводящая к оптимальному решению. В этом случае будет найдено наилучшее решение среди оставшихся ветвей (такие решения называют «оптимистичные»).

В алгоритме Гомори исходная задача решается обычными методами без учёта целочисленности. Если решение, полученное таким образом, удовлетворяет условию целочисленности, то оно и является оптимальным. Однако если оптимальное решение не является допустимым (условие целочисленности нарушено), то формулируется новая задача путём добавления новых ограничений. Новое ограничение выбирается так, что множество допустимых решений новой задачи не включает оптимальное решение, полученное на предыдущем этапе, но включает все допустимые решения задачи. Затем решается новая задача и т.д. Дополнительные ограничения называются отсечениями. Алгоритм Гомори [14] представлен на рис. 3.2.

Рис. 3.2. Алгоритм Гомори

3.1.3. ТЕОРИЯ ГРАФОВ

Граф характеризует отношения между множествами объектов, и теория графов направлена на исследование некоторых из многих возможных в заданном представлении свойств этих объектов [1, 8]. Более строго, граф есть совокупность точек и линий, соединяющих эти точки. Эти соединения могут обладать многими характеристиками, и теория графов, по существу, занимается изучением этих характеристик. Основное внимание уделяется тому, связаны ли точки между собой. В некоторых случаях важен также и порядок их соединения.

Вследствие общего характера представлений теории графов её основные концепции нашли широкую сферу применения. В настоящее время графы используются для формализованной постановки множества задач, связанных с дискретным размещением объектов. К ним, в частности, относятся: проектирование и исследование сетей связи, анализ электрических сетей, анализ печатных схем, графы потока сигналов и теория обратной связи, задачи проектирования электрических и монтажных схем, блок-схемы программ, исследование автоматов, анализ и синтез логических цепей, задачи календарного планирования, максимизация производительности поточной линии, планирование и обеспечение материально-технического снабжения, поиск информации, теория информации, стратегия инвестиций, анализ качества, исследование движения транспорта, размещение предприятий коммунального обслуживания, моделирование, служебная переписка, экономические задачи, чувствительность структур, теория игр, генетика, биология, исследование поведения индивидуумов, головоломки, генеалогия, архитектура, влияние ветра на траекторию полета и др.

Граф есть упорядоченная пара (V, E), где V – непустое множество, называемое множеством вершин; E – неупорядоченное бинарное отношение на V [т.е. если (u&v) в V&V, то (u&v) = (v&u)]. E называется множеством рёбер. Говорят, что ребро, принадлежащее множеству E, инцидентно вершинам, которые оно соединяет. Выше дано определение неориентированного графа, который полностью определяется списком вершин и указанием, какие пары вершин имеют соединяющие их ребра. Ориентированный граф есть упорядоченная пара (V, E), где V – множество вершин, а E – упорядоченное отношение на V (т.е. V×V). В этом случае E называется множеством дуг. Первая и вторая вершины дуги называются соответственно начальной и конечной вершинами.

Граф (V', E') называется подграфом графа (V, E), если V', E' содержатся соответственно в V, E. Графы с конечным числом рёбер называются конечными.

Петлёй называется ребро, концевые точки которого совпадают. Степенью (валентностью) вершины называется число инцидентных ей ребер. Кратностью пары вершин называется число соединяющих их рёбер.

В неориентированном (ориентированном) графе последовательность из n рёбер (дуг) e_1, ..., e_n называется маршрутом (ориентированным маршрутом) длины n, если существует соответствующая последовательность из $n + 1$ (не обязательно различных) вершин v_0, v_1, ..., v_n, таких, что e_i инцидентно (v_{i-1}& v_i), $i = 1, ..., n$. Маршрут (ориентированный маршрут) называется замкнутым (незамкнутым), если $v_0 = v_n$ ($v_0 \neq v_n$). Если $e_i \neq e_j$ для всех i и j, $i \neq j$, то маршрут (ориентированный маршрут) называется цепью (путём). Говорят, что множество рёбер (дуг) образуют цепь. Если $v_0 = v_n$, то цепь (путь) называется циклом (контуром). Если, кроме того, вершины различны, мы имеем простую цепь (простой контур). Протяжённость (число ребер) наиболее длинного (простого) цикла называется окружением графа. Часто для краткости вместо «простой цикл» («простой контур») говорят «цикл» («контур»). Во всех этих случаях определения относятся к такой последовательности рёбер графа, когда ни одно ребро не встречается повторно. Однако вершины, например, в маршруте, могут встречаться более одного раза.

Граф называется связным, если каждая пара вершин может быть соединена цепью. Граф, который не является связным, может быть разбит на конечное число связных подграфов, называемых компонентами или частями. Связностью графа называется наименьшее число вершин, удаление которых делает исходный граф несвязным. Разрезом графа называется минимальное множество ребер, удаление которых увеличивает число компонент. Граф является полным, если каждая вершина соединена ребром с каждой из всех других вершин. Граф называется двудольным, если его вершины могут быть разбиты на два таких множества, что каждое ребро соединяет вершину одного множества с вершиной другого. (При этом не обязательно, чтобы каждая вершина одного множества была инцидентна каждой вершине другого множества.) Связный граф называется регулярным, если все его вершины имеют одинаковую степень.

Деревом в неориентированном графе называется связный подграф, не имеющий циклов. Если все вершины графа принадлежат дереву, то оно называется покрывающим. Ориентированное дерево имеет корень в вершине v, если существует путь от v к каждой из других вершин. Рёбра графа, принадлежащие покрывающему дереву, называются ветвями. Все остальные рёбра называются хордами.

Ориентированный граф является сильно связным, если для каждых двух вершин u и v существует путь от u к v и путь от v к u. Вершина связного графа, после удаления которой граф становится несвязным, называется точкой сочленения.

Граф называется планарным, если он может быть уложен (изображён) на плоскости таким образом, что его рёбра будут пересекаться только в вершинах.

Плоская карта – это планарный граф вместе с областями или зонами (часто называемыми странами), простые циклы которого разделяют плоскость таким образом, что каждое ребро является границей двух областей. (Два ребра, являющихся границами смежных областей, могут быть заменены одним ребром, разделяющим эти две области). В более общей формулировке карта M – это граф G вместе с поверхностью S, на которой G нанесён таким образом, что его рёбра пересекаются только в их граничных точках. Говорят, что карта M является укладкой графа G на поверхности S.

Правильной раскраской карты в n цветов (n-раскраской) называют такую, когда каждая из областей раскрашивается в один из n цветов и ни одна пара смежных областей не окрашивается одним цветом (в дальнейшем будем иметь дело только с правильной раскраской).

Поверхность является неориентируемой, если можно выбрать направление вращения вокруг некоторой точки и двигать эту точку по поверхности, соблюдая одну и ту же ориентацию относительно точки задания вращения так, что, когда точка вернётся в исходное положение, направление вращения будет противоположным первоначальному. (Лента Мёбиуса представляет пример такой поверхности). В противном случае поверхность называется ориентируемой.

Фактором графа G называется подграф, который включает все его вершины, но не является вполне несвязным. Регулярный подграф степени n является n-фактором. Граф G является суммой факторов, если он представляет собой объединение непересекающихся множеств их рёбер. В этом случае имеют дело с факторизацией графа G.

Два графа называются изоморфными тогда и только тогда, когда имеется взаимно-однозначное соответствие между их вершинами и рёбрами при сохранении отношений инцидентности.

Замечание. Изоморфизм графа с самим собой называется автоморфизмом. Автоморфизм есть перестановка вершин, сохраняющая смежность. Последовательное выполнение двух автоморфизмов есть также автоморфизм. Фактически такая совокупность автоморфизмов образует группу, называемую группой графа.

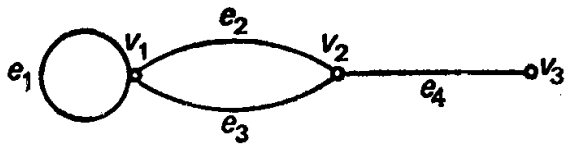

Рис. 3.3. Иллюстрация графа

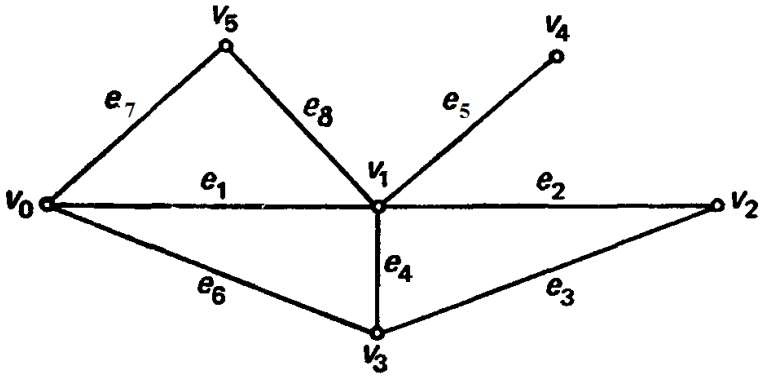

Рис. 3.4. Иллюстрация цепей, маршрутов, цикла

Для уяснения соответствующих определений обратимся к рис. 3.3 и 3.4. В графе на рис. 3.3: e_1 – петля; e_2 и e_3 – параллельные рёбра; v_2 и v_3 – смежные вершины; e_2 и e_4 – смежные рёбра. Обозначим число элементов множества S через $|S|$, так что $|V|$ – это число вершин, а $|E|$ – число рёбер.

На рисунке 3.4: $e_1e_2e_3e_4e_2$ – маршрут; $e_1e_2e_3e_4$ – цепь; $e_1e_2e_3e_4e_8e_7$ – цикл; $e_1e_2e_3$ – простая цепь; $e_2e_3e_4$ – простой цикл; e_5 – разрез.

Пример. Задача о мостах (Эйлеров путь). Одна из самых первых задач теории графов – это известная задача о кёнигсбергских мостах, интриговавшая жителей этого прусского города до тех пор, пока в 1736 г. не была решена Эйлером. Эйлер писал: «В городе Кёнигсберге имеется остров А, называемый Кнайпхофом, который охвачен двумя рукавами реки Прегель (рис. 3.5). Через эти два рукава перекинуты семь мостов *a*, *b*, *c*, *d*, *e*, *f* и *g*. Вопрос состоит в следующем: можно ли спланировать прогулку таким образом, чтобы по каждому из этих мостов пройти один и только один раз».

Поставим в соответствие каждой области суши вершину и соединим две вершины ребром, если между соответствующими областями имеется мост. В результате получится граф (рис. 3.6), соответствующий карте.

Эйлеровым путём в графе G называется такой путь, в котором каждое из рёбер встречается только один раз. Эйлер показал, что такой путь существует тогда и только тогда, когда граф G содержит не более двух вершин нечётной степени и является связным (очевидное необходимое условие). Так как в рас-

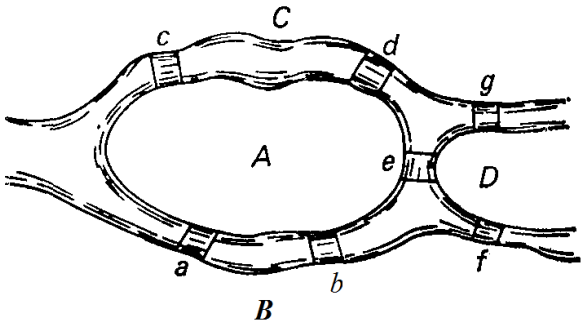

Рис. 3.5. Мосты Кёнигсберга

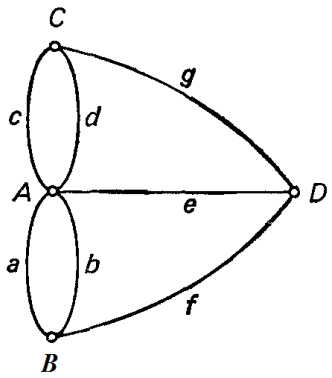

Рис. 3.6. Граф, соответствующий карте рис. 3.5

сматриваемом графе имеется четыре вершины нечётной степени, решение задачи о кёнигсбергских мостах даёт отрицательный ответ. Если граф содержит точно две вершины нечётной степени *A* и *B*, то в эйлеровом пути эти вершины должны быть конечными. Если вершин нечётной степени нет, то граф имеет замкнутый эйлеров путь (эйлеров контур).

Чтобы показать, как построить эйлеров путь, допустим, что граф *G* имеет точно две вершины нечётной степени *A* и *B*. Начнём движение с вершины *A* и используем последовательность неповторяющихся рёбер до тех пор, пока это возможно. Такой путь не может окончиться в вершине чётной степени, так как из любой такой вершины возможно продолжение пути по ребру, отличному от того, по которому мы в эту вершину попали. Он не может окончиться и в вершине *A*, так как первое ребро пути смежно с вершиной *A* и после его прохождения остаётся чётное число неиспользованных рёбер для продолжения пути через точку *A*. Такой путь заканчивается в другой конечной вершине *B*. Если окажется, что на пройденном пути использованы все рёбра графа, то этот путь и является искомым эйлеровым путём. В противном случае мы удалим использованные рёбра и получим новый граф *G'*, каждая вершина которого имеет чётную степень, так как путь через каждую вершину содержит чётное (возможно,

нулевое) число рёбер, за исключением вершин A и B. В силу связности графа G на рассматриваемом пути существует вершина C, смежная с некоторым ребром графа G'. Начиная движение от вершины C построим в графе G' новый путь, который должен закончиться в C, так как все вершины G' имеют чётную степень. Продлим первый путь, «сращивая» его с путём от A к C, а затем со вторым путём от C к C и в конце концов с путём от C к B. Будем повторять этот процесс до тех пор, пока не используем все рёбра графа G. Таким способом можно показать, что граф, имеющий $2n > 0$ вершин нечётной степени, может быть покрыт точно n путями.

Интересная задача организации движения городского транспорта связана с выбором улиц для одно- и двустороннего движения. Одностороннее движение следует организовывать на максимально возможном числе улиц. Каковы условия, которым должна удовлетворять схема движения по улицам, чтобы любые два пункта города были взаимно достижимы? Это оказывается возможным, если на всех улицах, принадлежащих контурам, организовано одностороннее движение, а все улицы, не принадлежащие контурам, используются для двустороннего движения.

Многочисленные задачи и алгоритмы их решения с использованием теории графов рассмотрены в [1, 8]

3.1.4. ГЕОМЕТРИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Геометрическое программирование – раздел математического программирования, изучающий подход к решению нелинейных задач оптимизации специальной структуры [2, 3]. Название объясняется тем, что данная теория основана на неравенстве между средним геометрическим и средним арифметическим (геометрическое неравенство Коши).

Неравенство Коши устанавливает, что среднее арифметическое n неотрицательных чисел не меньше их среднего геометрического:

$$\frac{x_1 + x_2 + \ldots + x_n}{n} \geq \sqrt[n]{x_1 x_2 \ldots x_n}, \quad x_i \geq 0, \quad i = 1, 2, \ldots, n.$$

Равенство достигается только при $x_1 = x_2 = \ldots = x_n$.

Пример 1: задача Дидоны. Среди замкнутых плоских кривых, имеющих заданную длину, найти кривую, охватывающую максимальную площадь. Рассмотрим частный случай – задачу для прямоугольников: найти длины сторон прямоугольника с периметром P, имеющего наибольшую площадь.

Обозначим длины сторон прямоугольника через x_1 и x_2, а его площадь – через S. Тогда математическая постановка задачи примет вид: найти x_1 и x_2, при которых

$$S = x_1 x_2 \to \max,$$

при ограничениях:

$$2x_1 + 2x_2 = P, \quad x_1 \geq 0, \; x_2 \geq 0.$$

Воспользуемся неравенством Коши при $n = 2$:

$$\frac{x_1 + x_2}{2} \geq \sqrt{x_1 x_2}\,. \tag{3.11}$$

Поскольку $x_1 + x_2 = P/2$, то из (3.11) следует:

$$\frac{P^2}{16} \geq x_1 x_2 = S\,. \tag{3.12}$$

Неравенство (3.12) обращается в равенство при $x_1 = x_2 = P/4$. Таким образом, прямоугольником наибольшей площади, имеющим заданный периметр P, является квадрат, длина стороны которого равна $P/4$.

При решении более сложных задач применяется также геометрическое неравенство или обобщённое неравенство Коши:

$$\sum_{j=1}^{n} w_j x_j \geq \prod_{j=1}^{n} x_j^{w_j} \tag{3.13}$$

при

$$\sum_{j=1}^{n} w_j = 1 \text{ – условие нормальности;}$$

$$w_j > 0, \quad j = 1, 2, \ldots, n \text{ – условие положительности;}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n.$$

Используя неравенство (3.13), можно доказать теорему, которая широко применяется для оценивания нелинейных функций.

Теорема. Решением экстремальной задачи

$$\prod_{i=1}^{n} x_i^{\beta_i} \to \max$$

при ограничениях:

$$\sum_{j=1}^{n} \alpha_j x_i = S;$$

$$x_i > 0, \;\; x_i \in R,$$

где

$$\beta_i > 0, \quad \alpha_i > 0, \quad \beta_i \in R, \quad \alpha_i \in R, \quad i = 1, \; 2, \ldots, n$$

является вектор X^* с компонентами

$$x_i^* = \frac{\beta_i S}{\alpha_i \beta},$$

где

$$\beta = \sum_{i=1}^{n} \beta_i .$$

Максимальное значение целевой функции M вычисляется по формуле

$$M = \left(\frac{S}{\beta} \right)^{\beta} \prod_{i=1}^{n} \left(\frac{\beta_i}{\alpha_i} \right)^{\beta_i} .$$

МОНОМЫ И ПОЗИНОМЫ

Функция f, определённая по правилу:

$$f(x) = c x_1^{\alpha_1} x_2^{\alpha_2}, \ldots, x_n^{\alpha_n},$$

называется мономом.

В дальнейшем мы будем предполагать, что параметры α_i (i = 1, ..., n) могут быть любыми действительными числами, но коэффициент c должен быть положительным.

Сумма K мономов называется позиномом.

Постановка задачи геометрического программирования: найти x_1^*, x_2^*, …, x_n^*, при которых

$$f(\boldsymbol{x}) \to \min;$$

$$g_i(\boldsymbol{x}) \le 1, \quad i = 1, \ldots, p;$$

$$h_i(\boldsymbol{x}) = 1, \quad i = 1, \ldots, q;$$

$$x_j > 0, \quad j = 1, \ldots, n,$$

где $f, g_1, \ldots, g_p$ – позиномы; $h_1, \ldots, h_q$ – мономы.

При помощи позиномов описывается большое число закономерностей и отношений, возникающих в различных областях, среди которых: оптимальное планирование, техническое проектирование, исследование химического равновесия, потоки в сетях, оптимальное управление, теория кодирования, управление запасами, системы связи, региональная экономика, автоматизированное проектирование, расчёт рисков.

Пример задачи оптимизации с позиномами. Требуется перевезти V кубометров угля из шахты на завод. Для транспортировки угля необходимо изготовить открытый прямоугольный контейнер. Стоимость материала, идущего на изготовление боковых сторон и дна контейнера, составляет b (р./м^2), на фронтальные стороны – d (р./м^2). Стоимость доставки одного контейнера не зависит от его размера и составляет C (р.).

Затраты на доставку угля складываются из транспортных расходов и стоимости материала, идущего на изготовление контейнера, в котором он будет перевозиться. Требуется выбрать размеры контейнера, при которых эти затраты будут минимальны.

Обозначим через L – длину, W – ширину и H – высоту контейнера. Тогда контейнер будет использоваться $V/(LWH)$ раз. Тогда транспортные расходы равны $CV/(LWH)$ (р.). Стоимость материала, потраченного на изготовление фронтальных сторон, составит $2dLH$ (р.), боковых сторон – $2bHW$ (р.), дна – bLW (р.).

Таким образом, задача свелась к минимизации позинома, выражающего суммарные затраты:

$$g(L,W,H) = \frac{CV}{LWH} + 2dLH + 2bHW + bLW \to \min.$$

Для решения задачи геометрического программирования обычно применяют метод сведения исходной задачи к задаче выпуклого программирования [2, 3].

3.1.5. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

Постановка задачи: найти такие значения варьируемых переменных (чисел) $x_1^*, x_2^*, \ldots, x_n^*$, при которых

$$R = f_0(x_1^*, x_2^*, \ldots, x_n^*) \to \min \qquad (3.14)$$

при ограничениях

$$f_i(x_1, x_2, \ldots, x_n) \geq 0, \quad i = 1, 2, \ldots, p; \tag{3.15}$$

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = p + 1, \; p + 2, \ldots, m. \tag{3.16}$$

При этом целевая функция f_0 в общем случае нелинейная.

В выражении (3.14) использована задача минимизации. Задачу максимизации функции R всегда можно заменить задачей минимизации функции $(-R)$.

В постановке (3.14) – (3.16) рассмотрен наиболее общий случай – так называемая задача условной оптимизации. При отсутствии ограничений (3.15), (3.16) задача упрощается и носит название задачи безусловной оптимизации.

Для решения безусловной задачи нелинейного программирования применяются методы нулевого порядка (не использующие производную целевой функции) – полного перебора, покоординатного спуска, симплексный, случайного поиска; первого порядка – градиентный, наискорейшего спуска; второго порядка – «тяжёлого шарика» и др. [4, 7].

МЕТОДЫ ПОИСКА ЭКСТРЕМУМА ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ

Постановка задачи: найти число x^* на интервале $[a, b]$, при котором $f_0(x) \rightarrow \min$.

Для решения поставленной задачи могут использоваться следующие методы.

а) *Метод сканирования* (*перебора*). Весь интервал $[a, b]$ разбивается на N равных частей. Во всех точках, включая a и b, вычисляются значения функции $f_0(x)$.

Среди полученных значений $f_0(x_i)$, $i = 1, 2, \ldots, N + 1$ находится наименьшее.

б) *Метод половинного деления*. Отрезок $[a, b]$ делится на четыре равные части (рис. 3.7).

Вычисляются значения $f_0(a)$, $f_0(x_1)$, $f_0(x_2)$, $f_0(x_3)$, $f_0(b)$, которые сравниваются между собой, и находится минимальное значение. Далее выбирается новый интервал, включающий два подынтервала с наименьшим вычисленным значением $f_0(x)$ на их общей границе. В примере рис. 3.7 таким интервалом является $[x_2, b]$.

Применяя к новому интервалу тот же приём, можно ещё более сузить интервал, где находится минимум. При этом для каждого нового разбиения нужно вычислять значения целевой функции только в двух новых точках, так как её значения на концах нового интервала и в его середине известны из предыдущих расчётов.

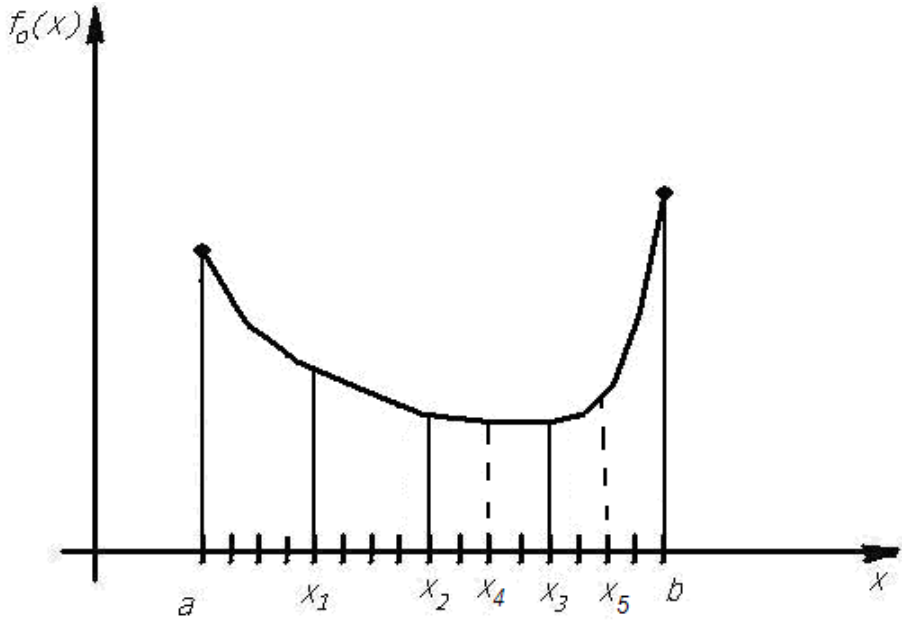

Рис. 3.7. Метод половинного деления

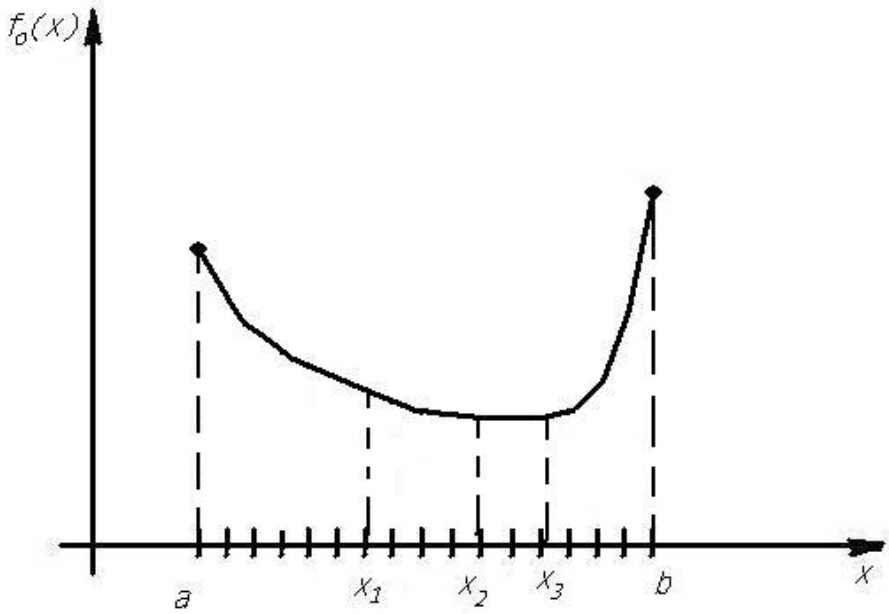

Рис. 3.8. Метод «золотого» сечения

Алгоритм повторяется до тех пор, пока для очередного отрезка не будет выполнено условие $b_i - a_i < \varepsilon$, где ε – заранее заданная точность вычислений.

в) *Метод «золотого» сечения.* По сравнению с методом половинного деления, лучшие результаты могут быть получены, если деление отрезка $[a, b]$, на котором отыскивается минимум, производить в определённом иррациональном отношении (рис. 3.8).

Определим отношение длины одинаковых отрезков $[a, x_1]$ и $[x_2, b]$ к длине отрезка $[a, b]$ из условия того, чтобы на всех итерациях отрезок локализации

разбивался бы в одном и том же отношении, и при этом, начиная со второй итерации, на каждой итерации вычислялась бы только одна новая точка, а вторая использовалась бы от предыдущей итерации. Полученное иррациональное отношение равно

$$y = \frac{3-\sqrt{5}}{2} \approx 0{,}38,$$

и носит название «золотого» сечения.

Итак, исходный отрезок делится на три неравные части, где $x_1 = a + 0{,}38 \times (b - a)$; $x_2 = b - 0{,}38(b - a)$ (рис. 3.8). Вычисляются значения $f_0(a)$, $f_0(x_1)$, $f_0(x_2)$, $f_0(b)$, которые сравниваются между собой и находится минимальное среди них. Далее выбирается новый интервал, включающий два подынтервала с наименьшим вычисленным значением $f_0(x)$ на их общей границе. В примере рис. 3.8 таким интервалом является $[x_1, b]$. Тем самым искомый минимум локализован в интервале, размеры которого на 38% меньше исходного. Новый интервал состоит из двух подынтервалов неравной длины. Внутри большего подынтервала ищем точку x_3 из условия $\frac{b - x_3}{b - x_1} = 0{,}38$. В точке x_3 вычисляется значение целевой функции, после чего процедура повторяется до тех пор, пока размер очередного отрезка локализации не станет меньше заранее заданной точности ε.

Таким образом, на каждом шаге вычисляется одно значение целевой функции, а не два, как в методе половинного деления.

г) *Метод поиска с использованием чисел Фибоначчи*. Последовательность чисел Фибоначчи задается рекуррентной формулой: $F_k = F_{k-1} + F_{k-2}$; $F_0 = F_1 = 1$. Таким образом, получается последовательность чисел: 1, 2, 3, 5, 8, 13, 21, … .

Алгоритм поиска. По заданной точности Δ, с которой необходимо найти положение экстремума функции $f_0(x)$ на отрезке $[a, b]$, рассчитывается вспомогательное число N:

$$N = \frac{b - a}{\Delta}.$$

Находится такое число Фибоначчи F_s, чтобы выполнялось неравенство: $F_{s-1} < N < F_s$.

Определяется шаг поиска по формуле:

$$h = \frac{b - a}{F_s}.$$

Рассчитывается значение $f_0(a)$. Рассчитывается $x_1 = a + hF_{S-2}$ и $f_0(x_1)$. Если $f_0(x_1) < f_0(a)$, то $x_2 = x_1 + hF_{S-3}$. Если $f_0(x_1) > f_0(a)$, то $x_2 = x_1 - hF_{S-3}$.

Последующие шаги выполняются с уменьшающейся величиной шага, так как на каждом последующем шаге будет использоваться меньшее число Фибоначчи (для i-го шага F_{S-i-2}).

На i-м шаге правило следующее. Вычисляем $x_{i+1} = x_i + hF_{S-i-2}$. Если $f_0(x_{i+1}) < f_0(x_i)$, то $x_{i+2} = x_{i+1} + hF_{S-i+1}$.

Если $f_0(x_{i+1}) > f_0(x_i)$, то $x_{i+2} = x_i - hF_{S-i+1}$.

Указанный процесс продолжается до тех пор, пока не будут исчерпаны все числа Фибоначчи в убывающей последовательности до $F_1 = 1$.

МЕТОДЫ ПОИСКА ЭКСТРЕМУМА ФУНКЦИИ МНОГИХ ПЕРЕМЕННЫХ

а) *Метод сканирования*. Метод сканирования заключается в последовательном вычислении целевой функции в ряде точек, принадлежащих области изменения варьируемых переменных, и нахождении среди них такой, в которой целевая функция минимальна.

Достоинства метода: независимость поиска от вида целевой функции (глобальный минимум будет найден и при наличии оврагов, и локальных минимумов); независимость от видов ограничений.

Недостатки метода: необходимость вычисления значений целевой функции для большого числа точек. Однако при современном развитии вычислительной техники этот недостаток не существенен. Ещё один недостаток – метод не работает для незамкнутой области определения варьируемых переменных.

б) *Метод покоординатного спуска* (поочередного измерения переменных, Гаусса-Зейделя). В этом методе поочерёдно изменяются все независимые варьируемые переменные так, чтобы по каждой из них достигалось наименьшее значение целевой функции. Очередность варьирования независимых переменных при этом устанавливается произвольно и обычно не меняется в процессе поиска.

Каждая уточняемая переменная варьируется до тех пор, пока в данном осевом направлении не будет найден минимум, после чего начинается процесс шагового поиска по следующему осевому направлению. Стратегия поиска минимума по каждой переменной может быть любая, например, использоваться один из методов одномерной оптимизации (половинного деления, «золотого» сечения, чисел Фибоначчи).

в) *Градиентные методы*. В основу градиентных методов поиска оптимума положены вычисление и анализ частных производных целевой функции $f_0(x)$. Поиск оптимума при использовании метода градиента производится в два этапа. На первом находятся значения частных производных по всем независимым переменным, которые определяют направление градиента в рассматриваемой точке. На втором этапе осуществляется шаг в направлении, обратном направлению градиента, т.е. в направлении наибыстрейшего убывания целевой функции. При выполнении шага одновременно изменяются значения всех независимых переменных. Каждая из них получает приращение, пропорциональное соответствующей составляющей градиента по данной оси.

Алгоритм градиентного метода может быть записан следующим образом:

$$x_j^{(k+1)} = x_j^{(k)} - h^{(0)} \frac{\partial f_0(x^{(k)})}{\partial x_j}, \quad j = 1, ..., n,$$

где h – значение шага.

г) *Метод «тяжёлого шарика»*. Используется в задачах с целевыми функциями, имеющими несколько локальных экстремумов, и в этом смысле может быть охарактеризован как метод поиска глобального экстремума.

В данном методе используется вторая производная целевой функции. Новое значение варьируемых переменных определяется по формуле:

$$x_j^{(k+1)} = x_j^{(k)} - h^{(0)} \frac{\partial f_0(x^{(k)})}{\partial x_j} - \rho \frac{\partial^2 f_0(x^{(k)})}{\partial x_j^2}, \quad j = 1, ..., n,$$

где ρ – коэффициент, позволяющий «проскакивать» небольшие локальные минимумы целевой функции.

д) *Методы случайного поиска*. Основная идея методов случайного поиска заключается в том, чтобы перебором случайных совокупностей значений независимых переменных найти оптимум целевой функции или направление движения к нему.

Общим для всех методов случайного поиска является применение случайных чисел в процессе поиска. Существует большое разнообразие методов и реализующих их программ для получения случайных чисел. Кроме того, практически во всех современных языках программирования высокого уровня имеются встроенные генераторы случайных чисел (функции RND, RANDOMIZ и т.д.). В результате получается последовательность случайных чисел, лежащих в диапазоне [0, 1].

е) *Метод случайных направлений*. Вычисляется значение целевой функции в точке начального приближения x. Генератором получают n случайных чисел r_i, $i = 1, 2, \ldots, n$ для всех n независимых переменных (для двухмерного случая – 2). Случайные числа центрируются: $r_i = r_i - 0{,}5$. После этого случайные числа лежат в диапазоне [–0,5; 0,5]. Делается шаг в полученном случайном направлении:

$$x_i^1 = x_i^0 + h_i r_i, \ i = 1, 2, ..., n,$$

где h_i – шаг по i-му направлению; r_i – i-е случайное число.

Вычисляется значение целевой функции в точке x^1. Если новое значение меньше предыдущего, шаг считается удачным и запоминаются значения x^1 и $f_0(x^1)$, после чего делается шаг в новом случайном направлении.

Если же случайный шаг оказался неудачным, то генерируются новые случайные числа и делается новый случайный шаг из точки x^0.

Поиск заканчивается, если после выполнения серии из n шагов меньшего значения функции цели найти не удаётся.

Для решения условной задачи нелинейного программирования применяются методы сканирования, штрафов, проектирования вектора-градиента, прямого поиска с возвратом, случайного поиска [4, 7].

ж) *Симплексный метод*. Метод основан на том, что по известным значениям целевой функции в вершинах выпуклого многогранника, называемого симплексом, находится направление, в котором следует сделать следующий шаг, чтобы получить наибольшее уменьшение целевой функции. При этом под симплексом в n-мерном пространстве понимается многогранник, имеющий $n + 1$ вершину, каждая из которых определяется пересечением n гиперплоскостей данного пространства.

Примером симплекса на плоскости является треугольник. В трёхмерном пространстве симплексом будет четырёхгранная пирамида. Рассмотрим наглядную иллюстрацию алгоритма симплексного метода на примере задачи отыскания наименьшего значения целевой функции двух независимых переменных с линиями постоянного уровня, изображенными на рис. 10.

Прежде всего, производится расчёт значений целевой функции в трёх точках S_{10}, S_{20} и S_{30}, соответствующих вершинам симплекса (треугольника). Из найденных значений целевой функции выбирается наибольшее. В представленном на рис. 3.9 случае наибольшее значение целевой функции получается в точке S_{10}.

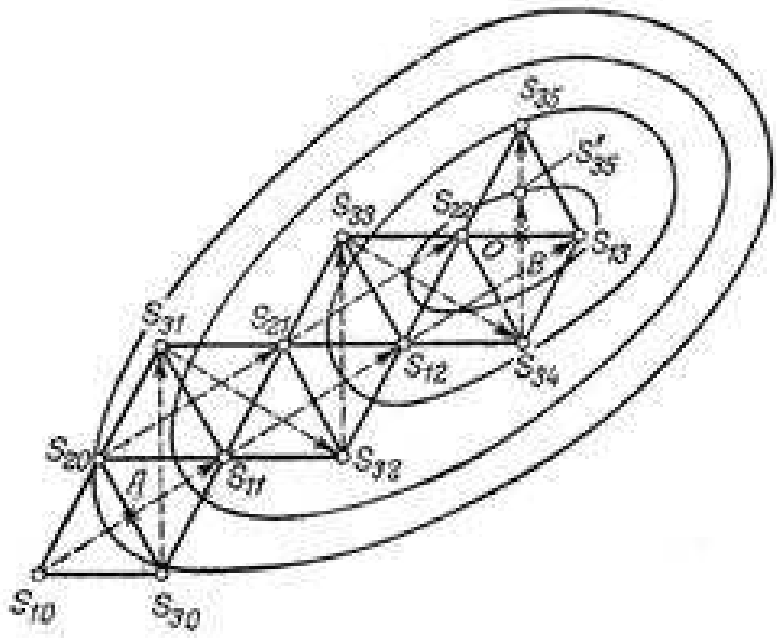

Рис. 3.9. Симплексный метод

Далее строится новый симплекс, для чего вершина S_{10} исходного симплекса заменяется вершиной S_{11}, расположенной симметрично вершине S_{10} относительно центра грани симплекса, находящейся против вершины S_{10}. Для варианта, изображённого на рис. 3.9, построение нового симплекса осуществляется определением центра A стороны треугольника $S_{20}S_{30}$, после чего на продолжении прямой, проведённой через вершину S_{10} и точку A, откладывается отрезок $AS_{11} = AS_{10}$. Пунктирная стрелка, соединяющая прежнюю вершину с новой, показывает путь преобразования симплекса.

В новой вершине S_{11} вычисляется значение целевой функции, которое сравнивается с известными значениями для других вершин нового симплекса (S_{20} и S_{30}), и снова находится вершина S_{30} с наибольшим значением целевой функции, подлежащая исключению при построении следующего симплекса $S_{11}S_{20}S_{31}$, и т.д.

В результате применения рассмотренной процедуры исключения вершин симплексов с наибольшим значением целевой функции, процесс сходится к минимальному значению. На рисунке 3.9 видно, что вблизи от оптимума может возникнуть зацикливание, которое для рассматриваемого случая двух переменных сводится к тому, что вновь полученная вершина S_{35} последнего симплекса $S_{13}S_{22}S_{35}$ исключается и образуется предыдущий симплекс $S_{13}S_{22}S_{34}$. Для того, чтобы устранить зацикливание, достаточно изменить размеры симплекса в сторону его уменьшения. Наиболее просто это можно сделать, если вдоль прямой

$S_{34}BS_{35}$ от точки B отложить отрезок, равный половине отрезка $S_{34}B$, в результате чего во вновь полученном деформированном симплексе $S_{13}S_{22}S'_{35}$ исключению уже будет подлежать вершина S_{13}. Если зацикливание возникнет снова, размеры симплекса опять уменьшатся, пока не будет достигнута требуемая точность определения оптимума.

Критерием окончания поиска могут служить размеры симплекса. Поиск можно прекратить, например, если все рёбра симплекса станут меньше заданной достаточно малой величины.

Таким образом, алгоритм симплексного метода допускает автоматическое изменение величины шага, при использовании которого вдали от оптимума возможно применение симплексов большого размера, что обеспечивает более быстрый спуск.

Задачи на условный экстремум с ограничениями типа неравенств. Рассмотрим задачу условной оптимизации (3.14) с ограничениями (3.15). Число ограничений типа неравенств может быть любым, т.е. меньше и больше числа независимых переменных: $p > n$, $p < n$, $p = n$. В данной задаче возможны два варианта:

1) оптимум лежит внутри допустимой области X изменения независимых переменных, ограниченной неравенствами (3.15). В этом случае задачу можно решить любым методом безусловной оптимизации;

2) оптимум лежит на границе. Тогда используются специальные методы, рассматриваемые ниже.

а) *Метод штрафов.* На основе функций $f_0(X)$ и $f_i(X)$, $i = 1, \ldots, p$ строится функция $R(X, K)$ следующего вида:

$$R(X, K) = f_0(X) + S(K, f_1(X), ..., f_m(X)),$$

где K – параметр, называемый коэффициентом штрафа; S – функция штрафа за нарушения ограничений (штрафная функция).

В зависимости от вида различают внутренние (барьерные) и внешние функции штрафа, а методы построения последовательности задач для минимизации функций $R(X, K)$ – соответственно методами внутренней и внешней точек.

В обоих методах решается последовательность задач безусловной минимизации функций $R(X, K_i)$, $i = 0, 1, 2, ...$, локальные минимумы которых $X^*(K_i)$ при $K_i \to \infty$ стремятся к точке X^*, являющейся решением исходной задачи.

В методах внутренней точки функция штрафа строится таким образом, чтобы обеспечивалось приближение к решению X^* внутри допустимой области. В этом случае функция штрафа должна резко возрастать при приближении к границе допустимой области изнутри, тем самым препятствуя нарушению ограничений. На границе области функция штрафа либо не существует, либо имеет разрыв. Примерами внутренних функций штрафа являются следующие:

$$S_1(X,K) = -\frac{1}{K}\sum_{i=1}^{m} \ln\, f_i(X); \quad S_2(X,\, K) = \frac{1}{K}\sum_{i=1}^{m} \frac{1}{f_i(X)}.$$

В методах внешней точки функция штрафа резко возрастает при выходе за границы допустимой области с тем, чтобы предотвратить блуждание точек слишком далеко от неё. Примером внешней функции является квадратичная функция штрафа вида:

$$S_3(X,\, K) = K\sum_{i=1}^{m} (\min(0,\, f_i(X)))^2 .$$

б) *Метод прямого поиска с возвратом.* Основная идея этого метода состоит в том, что оптимум ищется с применением любого метода безусловного спуска до тех пор, пока некоторые из неравенств не окажутся нарушенными. Тогда спуск прекращается и осуществляется возврат в допустимую область по направлению антиградиентов к тем ограничениям, которые оказались нарушенными.

Шаг в сторону исправления нарушенных ограничений производится по формуле:

$$x' = x - h\nabla f_i(X),$$

где x – точка нарушения ограничений; $\nabla f_i(X)$ – градиент функции $f_i(X)$, для которой нарушено ограничение; h – шаг в направлении исправления ограничений.

Задачи на условный экстремум с ограничениями типа равенств. Рассмотрим задачу условной оптимизации (3.14) с ограничениями (3.16). Число ограничений типа равенств должно быть меньше числа независимых переменных: $p < n$.

а) *Метод прямого поиска с возвратом.* Ограничения типа равенств преобразуются к виду:

$$\sum_{i=1}^{m} [\, f_i(x_1,\, x_2,\, ...,\, x_n)]^2 < \delta, \qquad (3.17)$$

где δ – некоторая заранее заданная величина.

В этом случае из точки начального приближения осуществляется спуск любым методом вплоть до нарушения условия (3.17), после чего производится возврат на ограничения (3.16) по нормали к нему. Поиск прекращается, если расстояние между точками, лежащими на ограничениях, и которые происходит возврат после двух последующих нарушений условия (3.17) не превышает заданной погрешности.

б) *Метод обобщенного критерия* (*метод* «*штрафов*»). Вводится вспомогательная функция вида:

$$R = f_0(X) + \alpha \sum_{i=1}^{m} [f_i(X)]^2 ,$$

где α – положительное число, величина которого должна быть достаточно большой.

Далее решается задача безусловной оптимизации для функции *R*.

Общая задача математического программирования. Рассмотрим задачу условной оптимизации (3.14) с ограничениями (3.15) и (3.16). Для решения задачи прежде всего ищется точка начального приближения x^0, удовлетворяющая всем условиям (3.15), (3.16).

Далее строится комбинированная внутренне – внешняя функция штрафа вида:

$$R = f_0(X) - \frac{1}{K} \sum_{i-1}^{m} \ln f_i(X) + K \sum_{i=m+1}^{q} [f_i(X)]^2 .$$

В данном случае для всех ограничений выбран один и тот же штраф *K*. Экстремум функции *R* ищется любым методом безусловной оптимизации.

3.1.6. ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Рассмотрим многостадийные процессы, которые характеризуются следующими особенностями [4, 7]:

– процесс дискретно распределён во времени или пространстве;

– отдельные стадии процесса обладают относительной независимостью, т.е. вектор выходных координат любой стадии зависит только от вектора входных координат на эту стадию и управления на ней;

– критерий оптимальности (целевая функция) всего процесса является суммой критериев оптимальности каждой стадии.

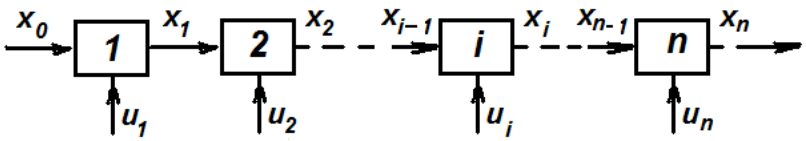

Рис. 3.10. Многостадийный процесс

Схема многостадийного процесса приведена на рис. 3.10. На каждой i-й стадии (i = 1, 2, …, n), x_{i-1} – входная величина; x_i – выходная величина; u_i – управление.

Пусть известна математическая модель каждой i-й стадии, позволяющая рассчитать выходную координату по значениям входной координаты и управления:

$$x_i = \varphi_i(x_{i-1}, u_i).$$

Постановка задачи динамического программирования:

Найти управления на каждой стадии $u_1^*, u_2^*, ..., u_n^*$, $u \in U$, при которых

$$R = \sum_{i=1}^{n} r_i(x_{i-1}, u_i) \to \min(\max),$$

где U – допустимая область изменения управлений; r_i – критерий оптимальности (целевая функция) i-й стадии; R – критерий оптимальности всего процесса.

Ричард Беллман сформулировал принцип оптимальности для многостадийных процессов: оптимальная стратегия обладает таким свойством, что, каково бы ни было начальное состояние и начальное решение, последующие решения должны приниматься, исходя из оптимальной стратегии относительно состояния, получаемого в результате первого решения.

Метод поиска оптимальных управлений, использующий принцип Беллмана, называется динамическим программированием.

Основная идея динамического программирования заключается в том, что если какой-либо поток изменяется на каждой стадии процесса, то, если на последней стадии режим работы (независимо от режима работы на всех стадиях) не будет оптимальным по отношению к поступающему на неё потоку, не будет оптимальным и режим всего многостадийного процесса в целом.

Применение метода динамического программирования состоит в определении такого режима работы стадии, который минимизирует (максимизирует) критерий на этой и всех последующих стадиях для любых возможных состояний поступающего на неё потока. Обычно рассмотрение начинается с послед-

ней стадии процесса. Оптимальный режим всего процесса определяется постадийно.

Основным уравнением динамического программирования является функциональное уравнение вида

$$R = \max_{u_1 \in U} \left\{ r_1(x_0, u_1) + \sum_{i=2}^{n} r_i(x_{i-1}, u_i) \right\}.$$

Здесь для определённости рассматривается максимизация целевой функции.

Процедура применения принципа оптимальности для оптимизации n-стадийного процесса, очевидно, должна начинаться с последней стадии процесса, для которой не существует последующих стадий, могущих повлиять (согласно принципу оптимальности) на выбор управления u_n на этой стадии. После того, как оптимальное управление u_n найдено для всех возможных значений входа последней стадии, можно приступить к определению оптимального управления на предыдущей стадии и т.д.

Решение задач методом динамического программирования проводится в два этапа.

На первом этапе определяются оптимальные управления как функции входных параметров, при этом начало рассмотрения – последняя стадия процесса:

$$R_1 = \max_{u_n \in U} r_n(x_{n-1}, u_n).$$

Предположим, что каким-либо методом найдено управление u_n^*, доставляющее максимум функции r_n. Переходим ко второй стадии от конца, для которой оптимизируемая функция имеет вид

$$R_2 = \max_{u_{n-1} \in U} \{ r_{n-1}(x_{n-2}, u_{n-1}) + R_1 \}.$$

На этой стадии ищется оптимальное значение управления u_{n-1}.

Далее процедура повторяется до тех пор, пока не дойдём до первой стадии.

Второй этап решения представляет собой последовательный расчёт оптимальных управлений от первой стадии к последней.

Если значение входной координаты x_0 известно, оптимальное управление на первой стадии вычисляется по соотношению:

$$u_1^* = u_1(x_0).$$

Зная u_1^* и математическую модель, описывающую первую стадию, можно определить выход первой стадии:

$$x_1 = \varphi_1(x_0, u_1^*).$$

Теперь можно определить оптимальное управление на второй стадии:

$$u_2^* = u_2(x_1).$$

Вычисления продолжаются до тех пор, пока не получим значения оптимальных управлений на всех стадиях.

Если значение входной координаты x_0 неизвестно, оно ищется на последнем шаге первого этапа, т.е. находится не только u_1^*, но и x_0^*, максимизирующие R. Дальнейшая процедура – та же самая, что и для случая, когда x_0 задано.

Преимущество метода по сравнению с методом полного перебора – снижение количества вычислений. Поясним это на примере. Рассмотрим n-стадийный процесс, в котором на каждой i-й стадии управления u_i представляют дискретные величины, принимающие k различных значений. Для каждого возможного решения, принимаемого на i-й стадии, имеется k возможных значений, принимаемых на $(i - 1)$-й стадии, и т.д. Таким образом, чтобы найти оптимальное решение методом полного перебора, нужно проанализировать k^n возможных путей решения. При использовании метода динамического программирования требуется проанализировать только kn комбинаций.

К недостаткам метода динамического программирования следует отнести отсутствие информации о том, как выполнить оптимизацию на каждой стадии, а также сложность его применения в случае процессов с рециклами.

Пример. Использование метода динамического программирования для задачи трассировки трубопроводов. Пусть расположение цехов предприятия иллюстрируется схемой, показанной на рис. 3.11.

Требуется соединить цеха трубопроводами так, чтобы каждый цех имел выход на конечную точку K напрямую или через другие цеха, при этом суммарные затраты на трубопроводы должны быть минимальными.

Затраты на прокладку трубопроводов между цехами обозначены цифрами над каждым возможным трубопроводом. Всего в данном примере число возможных труб равно 32. Проложив все 32 трубы все цеха будут гарантировано связаны с конечной точкой, однако такое решение не будет оптимальным с точки зрения выбранного критерия.

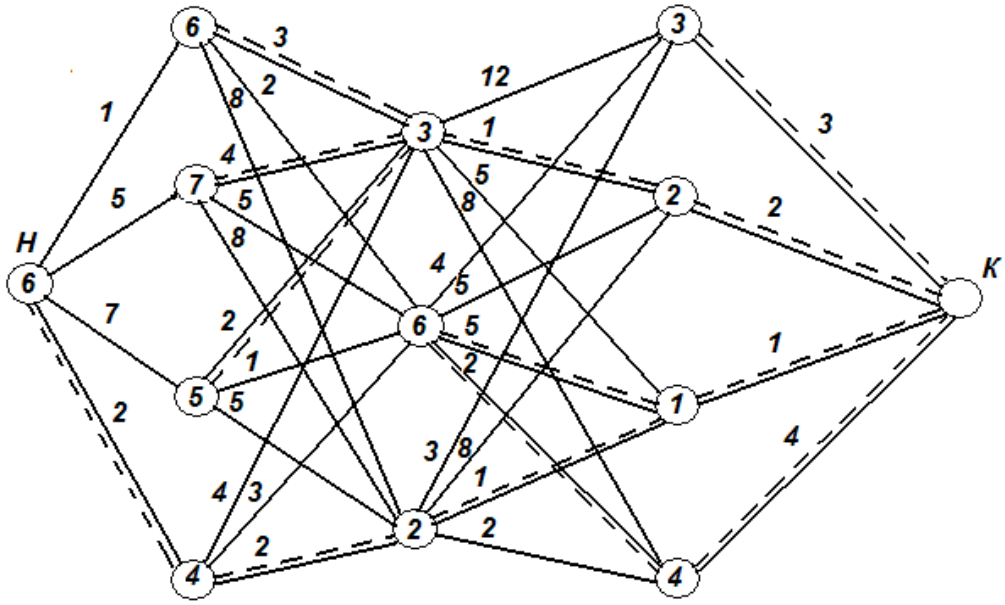

Рис. 3.11. Схема расположения цехов предприятия

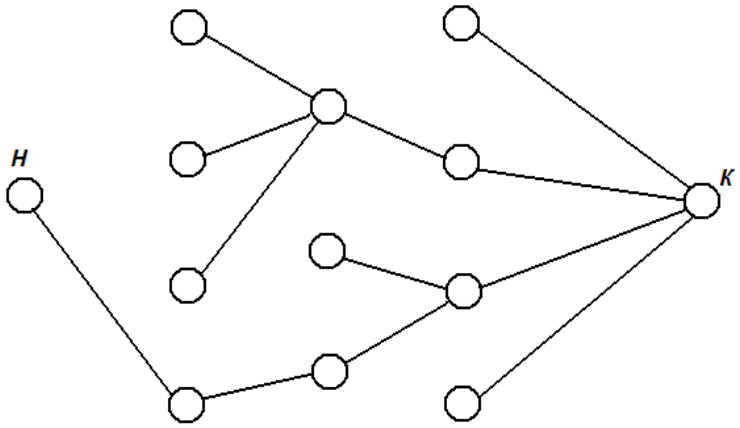

Рис. 3.12. Оптимальная трассировка

Применим принцип оптимальности. Для последнего ряда цехов в пункт *K* ведёт один путь. Значит, выбор трассировки труб из каждого цеха последнего ряда приводит к единственной возможности (она и оптимальна). Соответствующие затраты обозначим цифрами внутри кружка на схеме рис. 3.11, а оптимальную трассировку – пунктирной линией.

В предпоследнем ряду для каждого цеха перебираем все трассировки. Находим ту из них, для которой сумма затрат на прокладку трубопроводов до последнего ряда и от последнего ряда цехов до пункта *K* будет минимальна. Так продолжается до точки «*H*».

В результате получается трассировка, показанная на рис. 3.12.

Обратим внимание на тот факт, что если бы рассмотрение велось не от точки «*K*» к точке «*H*», а наоборот, то на первом шаге была бы выбрана трасса,

затраты на которую равны 1 (рис. 3.11). Такая стратегия, оптимальная на первом шаге, привела бы к неоптимальной глобальной стратегии («близорукая оптимизация» по Р. Беллману).

Подобным методом может быть решена другая, более простая задача: найти вариант прокладки трубопроводов от одной заданной точки (например, «*H*») до другой заданной точки (например, «*K*») через промежуточные точки, заданные на схеме, с минимальными затратами.

3.1.7. ВАРИАЦИОННЫЕ ЗАДАЧИ ОПТИМИЗАЦИИ

На практике распространены задачи, в которых необходимо найти одну или несколько функций, при которых заданный критерий достигает экстремума. Такой критерий, который ставит в соответствие функции число, называется функционал.

Переменная величина J называется функционалом, зависящим от функции $x(t)$, и обозначается $J = J[x(t)]$, если каждой функции $x(t)$ из некоторого класса функций соответствует значение J.

Функционалы бывают разных видов, при этом наиболее часто встречается на практике функционал Лагранжа вида:

$$J = \int_{t_0}^{t_1} f(t,\, x(t),\, x'(t))dt\,. \qquad (18)$$

Постановка простейшей вариационной задачи имеет следующий вид. Пусть необходимо отыскать функцию $x(t)$, доставляющую экстремум функционалу вида (3.18) с закреплёнными концами $x(t_0) = x_0$, $x(t_1) = x_1$.

Точное решение поставленной задачи можно отыскать, используя необходимое и достаточное условия экстремума функционала [9, 15, 16]. Необходимое условие – равенство нулю вариации функционала – приводит к уравнению Эйлера:

$$\frac{\partial f}{\partial x} - \frac{d}{dt}\frac{\partial f}{\partial x'} = 0\,.$$

Функция $x^*(t)$, являющаяся решением уравнения Эйлера (называемая экстремалью), должна быть проверена на достаточное условие, которое формулируется следующим образом.

Экстремаль $x^*(t)$ может быть включена в поле экстремалей – это будет, если выполнено условие Якоби. Для проверки данного условия подставим найденное решение в уравнение Якоби:

$$\left(\frac{\partial^2 f}{\partial x^2} - \frac{d}{dt}\left(\frac{\partial^2 f}{\partial x \partial x'}\right)\right)u - \frac{d}{dt}\left(\frac{\partial^2 f}{\partial (x')^2}u'\right) = 0\,.$$

Если решение $u = u(t)$ уравнения Якоби, удовлетворяющее условию $u(t_0) = 0$ не обращается в ноль ни в одной точке полуинтервала $t_0 < t \le t_1$, то решение $x(t, c_0)$ принадлежит полю экстремалей.

Функция Вейерштрасса:

$$E = f(t, x, x') - f(t, x, p) - (x' - p)\frac{\partial f(t, x, p)}{\partial p},$$

где p – наклон поля экстремалей, должна сохранять знак во всех точках (t, x), близких к экстремали x^*, и для близких к p значений x'.

Это условие слабого экстремума. Условие сильного экстремума: функция E сохраняет знак во всех точках (t, x) и для произвольных значений x'. Функционал $J[x]$ будет иметь максимум, если $E \le 0$, и минимум, если $E \ge 0$.

К сожалению, применение аналитического метода позволяет решить только задачи небольшой сложности. В связи с этим был развит математический аппарат численных методов, наиболее привлекательными из которых являются прямые методы решения вариационных задач.

Идея прямых методов заключается в сведении вариационной задачи к конечномерной и поиске вместо функций массива чисел. При этом не используется необходимое условие экстремума и не решается уравнение Эйлера.

Рассмотрим наиболее применяемые на практике прямые методы.

Метод Ритца. Идея метода заключается в том, что значения некоторого функционала рассматриваются не на произвольных функциях, а на возможных линейных комбинациях известных функций

$$x_n = \sum_{i=0}^{n} a_i w_i(t)$$

с постоянными коэффициентами a_i, составленных из n заранее заданных функций $w_i(t)$.

Функции x_n должны быть допустимыми в рассматриваемой задаче – прежде всего должны удовлетворять граничным условиям.

Коэффициенты a_1, a_2, …, a_n ищутся таким образом, чтобы функционал достиг экстремального значения.

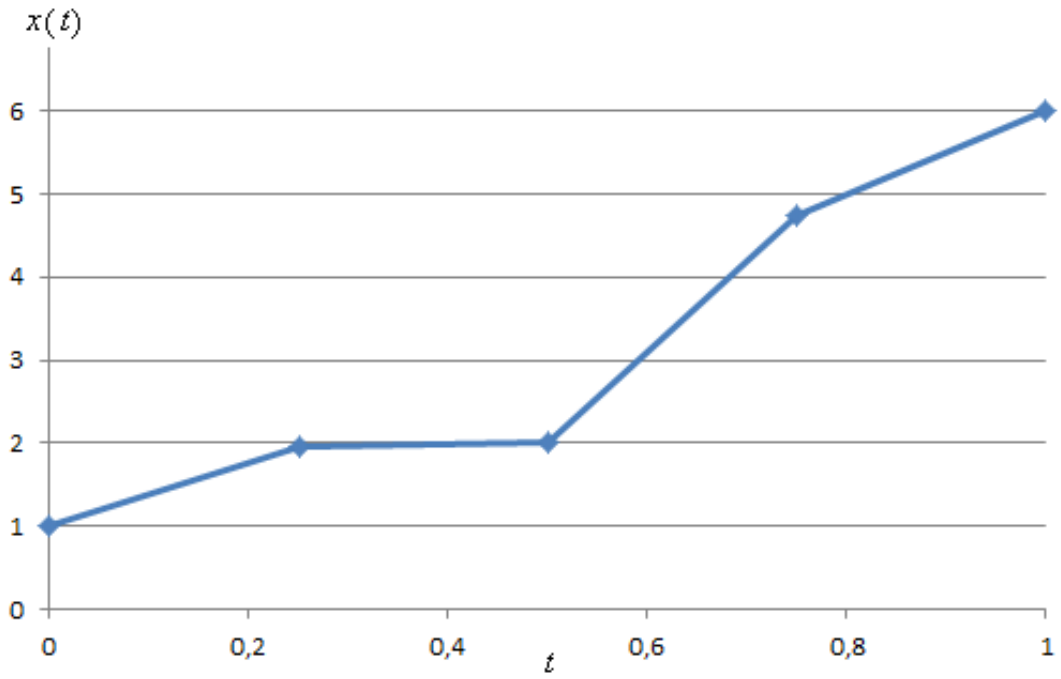

Рис. 3.13. Конечно-разностный метод Эйлера

Конечно-разностный метод Эйлера. Идея метода заключается в том, что решение ищется не на произвольных функциях, а лишь на ломаных, составленных из заданного числа n прямолинейных звеньев с заданными через Δt абсциссами вершин. Таким образом, требуется найти n значений x_i в точках $t_0 + i\Delta t$, при которых функционал экстремален (рис. 3.13).

Вариационные задачи с подвижными границами. Ранее при исследовании на экстремум функционала (3.18) предполагалось, что граничные точки (t_0, x_0) и (t_1, x_1) заданы. Предположим теперь, что одна или обе граничные точки могут перемещаться, т.е. концы экстремалей лежат на кривых $x = \varphi(t)$ и $x = \psi(t)$.

Задача будет ставиться следующим образом.

Необходимо найти экстремали и экстремум функционала

$$J = \int_{\gamma} f(t, x, x')dt ,$$

определённого на гладких кривых $x = x(t)$, концы которых $A(t_0, x_0)$ и $B(t_1, x_1)$ лежат на кривых: $x = \varphi(t)$ и $x = \psi(t)$.

Прямой метод Ритца решения задачи с подвижными границами

1. Задаётся некоторая функция $x_n = \sum_{i=1}^{n} a_i w_i(t)$.

2. Задаются начальные приближения коэффициентов $a_i^{(0)}$.

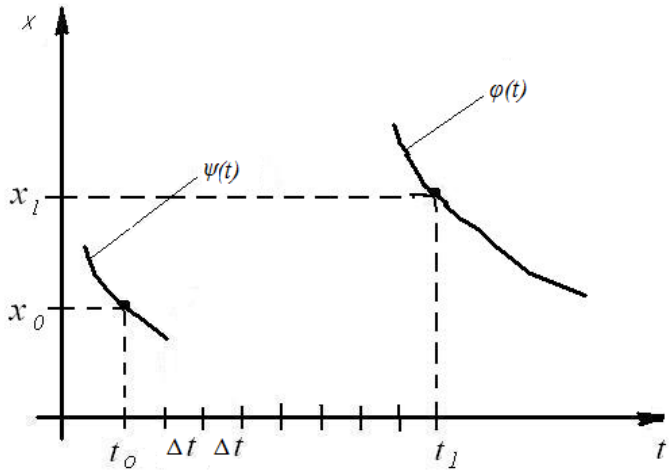

Рис. 3.14. Конечно-разностный метод Эйлера для задачи с подвижными границами

3. Решается уравнение $x_n(t) = \varphi(t)$, из которого находим t_0, x_0 – левую границу; решается уравнение $x_n(t) = \psi(t)$, из которого находим t_1, x_1 – правую границу.

4. Находим значение функционала $J = \int_{t_0}^{t_1} f(x, x_n, x'_n)$.

Меняем a_i, возвращаемся к п. 3, добиваемся экстремума функционала.

Таким образом, алгоритм отличается от прямого метода Ритца для простейшей вариационной задачи лишь п. 3.

Конечно-разностный метод Эйлера. Алгоритм иллюстрируется на рис. 3.14.

Задаём t_0, ищем $x_0 = \varphi(t_0)$.

Задаём t_1, ищем $x_1 = \psi(t_1)$.

Далее разбиваем отрезок $\{t_0, t_1\}$ на равные части с шагом Δt и ищем x_i по алгоритму для простейшей вариационной задачи.

Потом меняем t_0, t_1 и всё сначала, пока функционал не достигнет экстремума.

Вариационные задачи со связями. Вариационными задачами на условный экстремум называются задачи, в которых требуется найти экстремум функционала:

$$J[x_1, x_2, ..., x_n] = \int_{t_0}^{t_1} f(t, x_1, x_2, ..., x_n, x_1', x_2', ..., x_n')dt,$$

причём на функции x_i наложены некоторые связи.

а) *Голономные* (*алгебраические*) *связи*:

$$\varphi_i(t, x_1, x_2, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, m; \quad m < n.$$

б) *Неголономные связи*. Представляют собой обыкновенные дифференциальные уравнения:

$$\varphi_i(t, x_1, x_2, ..., x_n, x'_1, x'_2, ..., x'_n) = 0, \; i = 1, 2, ..., m.$$

в) *Изопериметрические связи* представляют из себя интегральные уравнения вида:

$$\int_{t_0}^{t_1} f_i(t, x_1, x_2, ..., x_n, x'_1, x'_2, ..., x'_n)dt = l_i, \quad i = 1, 2, ..., m.$$

где l_i – константы; m – может быть больше, меньше или равно n.

Прямые методы решения задач со связями. Покажем методику на примере задачи с изопериметрическими связями. Составляется вспомогательный функционал вида:

$$J^* = \int_{t_0}^{t_1} [f(t, x, x') + \lambda f_1(t, x, x')]dt.$$

Далее ищется безусловный экстремум вспомогательного функционала методом Ритца или конечно-разностным методом Эйлера; при этом дополнительно необходимо найти коэффициент λ при котором J^* имеет экстремум.

Для задач с голономными и неголономными связями алгоритм аналогичен.

3.1.8. ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ

В теории оптимального управления в общем случае рассматриваются системы, на поведение которых можно воздействовать (которыми можно управлять) путём изменения параметров управления. Последние выбираются с учётом определённых ограничений. Целью теории оптимального управления является разработка метода такого выбора параметров управления, при котором достигается оптимум некоторого функционала, например минимум времени, минимальный расход топлива, минимальные потери, максимум полезности и т.д.

Будем рассматривать объекты, состояние которых ($x_1(t)$, $x_2(t)$, …, $x_n(t)$) в момент t может быть описано системой из n дифференциальных уравнений первого порядка:

$$x' = \mathbf{f}(t, x, \mathbf{u}(t)), \tag{3.19}$$

где $\boldsymbol{u}(t) = (u_1(t), u_2(t), ..., u_m(t))$ – вектор параметров управления в момент t, вектор-функция f имеет компоненты φ_1 φ_2, ..., φ_n. Как известно из практики, в общем случае величины параметров управления нельзя выбирать произвольно, а только в соответствии с некоторым ограничением:

$$\mathbf{u}(t) \in U \text{ при всех } t, \tag{3.20}$$

где U – областью управления, $U \subset R^m$.

Задача состоит в том, чтобы выбрать вектор параметров управления $\boldsymbol{u}$, удовлетворяющий ограничению (3.20) и такой, что соответствующее решение уравнения (3.19) $x(t)$ (траектория процесса) из заданного начального состояния:

$$x(0) = x^{(0)} \tag{3.21}$$

попало бы в заданное конечное состояние

$$x(T) = x^{(k)} \tag{3.22}$$

в некоторый, не обязательно фиксированный момент $T > 0$, а при этом функционал

$$\int_0^T \varphi_0\big(t, \mathbf{x}(t), \mathbf{u}(t)\big)\, dt \tag{3.23}$$

достиг бы своего наименьшего (или наибольшего) возможного значения. Заметим, что специальный выбор начального состояния не приводит к потере общности, так как преобразованием координат можно всегда получить уравнение (3.21).

Управление, при котором функционал (3.23) достигает своего наименьшего (наибольшего) значения, называется оптимальным управлением, а соответствующая траектория x – оптимальной траекторией.

Задача (3.19) – (3.23) является так называемой неавтономной задачей оптимального управления с незакреплённым временем. Двумя наиболее известными случаями задачи (3.19) – (3.23) являются задача об оптимальном быстродействии и неавтономная задача с закреплённым временем. В задаче о быстродействии φ_0 $(t, \mathbf{x}, \mathbf{u}) \equiv 1$ и функционал (3.23) имеет вид:

$$T = \int_0^T dt\,, \tag{3.24}$$

т.е. необходимо минимизировать продолжительность процесса.

Если функции $\varphi_1, ..., \varphi_n$ не зависят явным образом от времени, то возникает автономная задача о быстродействии.

Поскольку искомыми неизвестными являются функции, задачи оптимального управления относятся к классу вариационных. Для их решения используются методы классического вариационного исчисления, прямые методы, принцип максимума Понтрягина [1, 7, 9].

При решении задач оптимального управления классическими вариационными методами, серьёзные, а иногда и непреодолимые трудности возникают в тех случаях, когда отыскиваемые управляющие воздействия не принадлежат к классу непрерывных функций или когда на переменные задачи наложены ограничения типа неравенств. Для решения таких задач иногда с успехом может быть использован метод, сформулированный и доказанный в работах Л. С. Понтрягина и его учеников, который получил название принципа максимума.

Принцип максимума формулируется как необходимый признак оптимальности для процессов, описываемых системами нелинейных обыкновенных дифференциальных уравнений. Показано, что если процесс характеризуется системой линейных уравнений, принцип максимума является достаточным условием оптимальности.

Проиллюстрируем принцип максимума на примере решения задачи о быстродействии.

Для простоты примем, что в рассматриваемой задаче о быстродействии используется только одно управляющее воздействие u, т.е. $m = 1$ в выражении (3.19), и процесс описывается системой уравнений:

$$x' = \mathbf{f}(t, x, u(t)). \qquad (3.25)$$

Допустим теперь, что оптимальное управление $u_{\text{опт}}(t)$, переводящее процесс из начального состояния $x^{(0)}$ в конечное $x^{(k)}$ за минимальное время T^* известно (рис. 3.16).

Это управление не обязательно должно быть непрерывной функцией t. Достаточно, чтобы оно имело лишь конечное число точек разрыва первого рода, т.е. таких точек, в которых величина функции изменяется скачком от одного конечного значения к другому. Соответствующая оптимальному управлению $u_{\text{опт}}(t)$ траектория процесса в фазовом пространстве обозначена на рис. 3.16 сплошной линией.

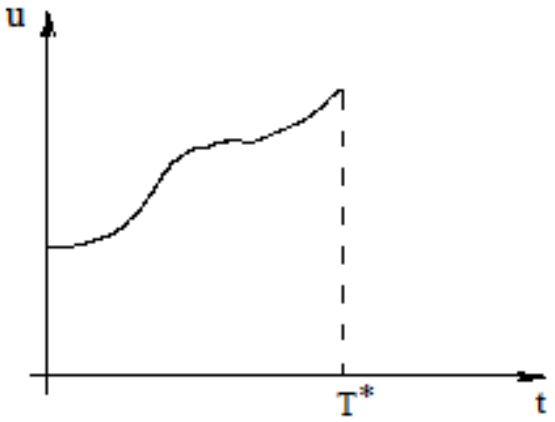

Рис. 3.15. Оптимальное управление

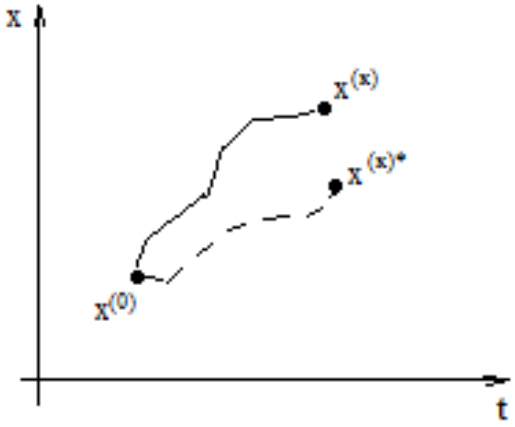

Рис. 3.16. Траектории процесса в фазовом пространстве, соответствующие оптимальному (сплошная линия) и неоптимальному (пунктир) управлениям

При любом другом управлении, отличающемся от оптимального, получаемая в результате траектория уже не будет переводить процесс в конечное состояние за минимальное время. При этом неоптимальная траектория либо попадает в конечную точку за время большее, чем T^*, либо вообще при этом управлении в конечную точку попасть невозможно.

Таким образом, при применении неоптимального управления, состояние процесса в момент времени T^* отличается от заданного конечного $x^{(k)}$. На рисунке 3.16 неоптимальная траектория показана пунктиром и заканчивается в точке $x^{(k)*}$, в которую процесс переводится при неоптимальном управлении за время T^*.

Предположим, что на некотором малом интервале времени длиной Δt оптимальное управление заменяется произвольным постоянным значением u^*. В результате найдем новый закон управлении $u(t)$, уже неоптимальный (рис. 3.17).

При использовании этого закона траектория процесса до момента времени $t_и$ совпадает с оптимальной, а с момента времени $t_и$ начнет отклоняться от нее. В момент времени T^*, определённый для оптимального процесса, новая траектория уже не попадает в заданную конечную точку $x^{(k)}$, а окажется на некотором расстоянии от неё (рис. 3.18).

Если величина интервала времени Δt, на котором оптимальное управление заменяется произвольным постоянным значением, выбрана достаточно малой, то отклонение от оптимальной траектории тоже мало. Очевидно, что это отклонение уже может дать представление о том, оптимальна или нет исходная траектория.

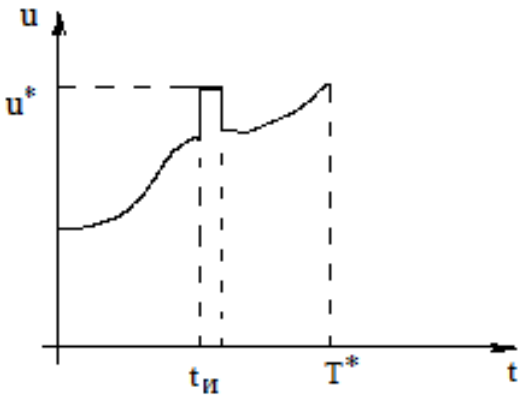

Рис. 3.17. Варьирование оптимального управления

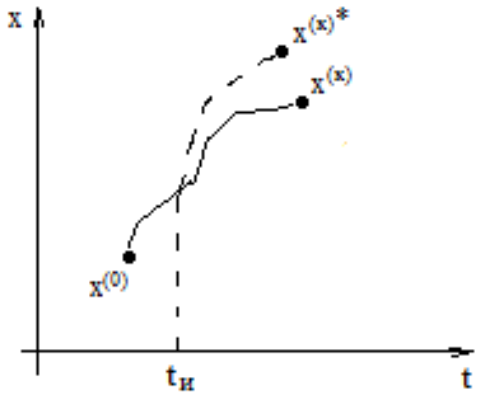

Рис. 3.18. Отклонение конечной точки траектории для варьированного управления

Такой приём носит название «игольчатой» вариации в отличие от вариации функции, использовавшейся при выводе уравнения Эйлера в классическом вариационном исчислении. Достоинство метода «игольчатой» вариации заключается в том, что разрывность функций, которая препятствует эффективному применению классических вариационных методов, в данном случае кладётся в основу математического аппарата, что позволяет включить в класс допустимых решений оптимальных задач также и разрывные функции.

Аналитическая формулировка принципа максимума выглядит следующим образом: в каждой точке траектории оптимальное управляющее воздействие должно выбираться из условия достижения максимального значения величины скалярного произведения $[\varphi(x, u), \lambda]$:

$$(\varphi'[x(t),\ u_{\text{опт}}(t)],\ \lambda(t)) = \max_{u \in U}(\varphi[x(t),\ u],\ \lambda(t)). \qquad (3.26)$$

При этом оптимальное управление определяется как функция величин x и λ.

Изменение вектора λ вдоль траектории процесса соответствует следующей системе уравнений:

$$\frac{d\lambda_i(t)}{dt} = -\sum_{k=1}^{m} \frac{\partial \varphi_k[x(t),\ u_{\text{опт}}(t)]}{\partial x_i} \lambda_k(t), \quad i = 1, \ldots, m. \qquad (3.27)$$

Итак, для оптимизации процесса, описываемого системой уравнений (3.25), в смысле наискорейшего перевода его из заданного начального состояния $x^{(0)}$ в заданное конечное $x^{(k)}$, можно воспользоваться условием (3.26), в котором для определении компонентов вектора $\lambda(t)$ применяется нетривиальное решение системы уравнений (3.27).

Принцип максимума можно записать в более компактной форме, если в условии (26) величину скалярного произведения обозначить как

$$H[\lambda(t),\, x(t),\, u] = (\varphi[x(t),\, u],\, \lambda(t)) = \sum_{k=1}^{m} \lambda_k(t)\varphi_k[x(t),\, u].$$

Функция $H(\lambda, x, u)$ при этом рассматривается как функция векторных переменных λ, x и u, и условие (3.26) может быть представлено в виде:

$$H[\lambda(t),\, x(t),\, u_{\text{опт}}(t)] = \max_{u \in U} H[\lambda(t),\, x(t),\, u], \qquad (3.28)$$

где максимум функции H ищется по всем возможным значениям вектора управления u из области его определения U.

Функцию H иногда называют гамильтонианом, подчеркивая тем самым её сходство с гамильтонианом уравнений движения материальной точки, в которых роль вектора λ выполняет вектор импульса движения.

С помощью функции H систему уравнений (3.25) можно также записать в следующей компактной форме:

$$\frac{\partial x_i}{\partial t} = \frac{\partial H}{\partial \lambda_i}, \quad i = 1, \ldots, m.$$

Нетрудно показать, что система уравнений (3.27) с учётом функции H представляется как:

$$\frac{\partial \lambda_i}{\partial t} = -\frac{\partial H}{\partial x_i}, \quad i = 1, \ldots, m.$$

Вычислительные аспекты принципа максимума

Изложим методику решения оптимальной задачи с применением соотношения максимума (3.28), т.е. на примере задачи о быстродействии.

Итак, ставится задача отыскания оптимального управления для процесса, описываемого системой дифференциальных уравнений (3.25), которое переводит его из некоторого начального состояния в конечное, причём в общем случае оба состояния не обязательно фиксируются заданием всех переменных.

Поскольку оптимальное управление удовлетворяет в каждой точке траектории соотношению максимума (3.28), оно устанавливает связь данного управления со значениями $x(t)$ и $\lambda(t)$ в этой точке и, следовательно, его можно записать как:

$$u_{\text{опт}}(t) = u_{\text{опт}}[\lambda(t), x(t)]. \quad (3.29)$$

Выражение (3.29), разумеется, не всегда может быть получено в аналитической форме и тогда его представляют в виде результатов численного решения задачи максимизации функции H выбором управляющих воздействий u_i (i = 1, ..., r), т.е. в форме таблиц или графиков.

В частном случае, когда правые части системы уравнений математического описания процесса (3.25) относительно просты, для нахождения зависимости (3.29) можно использовать уравнения:

$$\frac{\partial H}{\partial u_i} = 0, \ i = 1, \ldots, r,$$

определяющие экстремальные точки функции H внутри допустимой области U изменения управляющих воздействий u_i.

3.2. СТОХАСТИЧЕСКИЕ МЕТОДЫ

Детерминированные модели часто неадекватны реальным процессам, поскольку в практических задачах бывает трудно определить точные значения параметров модели. Это объясняется неполнотой и неточностью данных, на основе которых формируется модель. Наличие неопределённых факторов переводит задачу в новое качество: она из задачи нахождения оптимального решения превращается в задачу выбора решения в условиях неполной информации и неопределённости, обеспечивающего по возможности максимальное значение показателя эффективности.

В некоторых случаях параметры носят вероятностный характер и представляют собой случайные величины, характеристики которых нам известны или в принципе могут быть получены. Оптимизационные задачи, в которых отдельные или все параметры являются случайными величинами, называются стохастическими. Ограниченность и неточность информации о задаче приводит к ситуации, в которой приходится принимать решение в условиях риска. Теория и методы решения экстремальных задач в условиях неполной информации изучаются в разделе стохастического программирования.

Стохастические задачи – такие задачи исследования операций, когда принимающий решение не может сопоставить вероятности результатов, которые могут быть получены при выборе той или иной стратегии (т.е. не имеет основания полагать, что какой-либо результат более вероятен, чем любой другой, хотя сам возможный набор результатов известен).

Формулировки многих оптимизационных задач включают неопределённые параметры. Имеется несколько подходов к решению таких задач. В стохастическом программировании предполагается, что все неопределённые параметры являются случайными величинами с известными распределениями вероятностей. Робастная оптимизация используется тогда, когда требуется, чтобы решение было приемлемым для всех возможных значений неопределённых параметров. Последнее очень важно в тех ситуациях, когда небольшие изменения исходных данных задачи могут привести к тому, что её решение меняется кардинальным образом [2].

3.2.1. СТОХАСТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Основой для построения всех моделей стохастического программирования служит преобразование исходной задачи в вероятностной постановке в эквивалентную задачу, обладающую детерминированной структурой.

В стохастическом программировании исследуются одношаговые и многошаговые задачи. Задачи, в которых решение выбирается один раз и оно не изменяется при поступлении информации о реализациях случайных параметров, называются одношаговыми. В задачах этого класса целевая функция и ограничения могут быть выбраны разными способами. В качестве целевой функции могут быть приняты:

– математическое ожидание некоторой функции от решения;

– комбинация математического ожидания и среднего квадратического отклонения;

– вероятность попадания функции от решения в некоторую область.

Ограничения задачи в одних случаях могут выполняться при всех возможных значениях случайных параметров (жёсткая постановка), а в других случаях требуется, чтобы вероятность попадания решения в допустимую область была не меньше заданной (модель с вероятностными ограничениями).

В каждой конкретной задаче приходится специально оговаривать, что понимается под допустимым и оптимальным решениями. В многошаговых задачах по мере получения дополнительной информации о развитии процесса имеется возможность неоднократной корректировки решений, принятых ранее. Естественно, что для нужд практики многошаговые задачи более предпочтительны, чем одношаговые.

Модели стохастического программирования могут включать два типа переменных: ожидаемые и адаптивные переменные. Ожидаемые переменные представляют те решения, которые нужно принять здесь-и-сейчас: они не зависят от будущей реализации случайных параметров.

Рассмотрим пример задачи стохастического программирования. В процессе эксплуатации станка могут возникать неисправности, и станок останавливается для их устранения. Время устранения неисправности постоянно и равно $T_{\text{В1}}$. После ремонта станок вновь включается и продолжает свою работу. Предположим, что в течение времени t станок $n(t)$ раз выходит из строя, и пусть эта величина имеет порядок $n(t) = ct^{\alpha}$, где c – случайная величина с математическим ожиданием $m = M(c)$ и средним квадратическим отклонением $\sigma = \sigma(c)$. Показатель $\alpha > 1$, так как с течением времени станок вырабатывает свой ресурс, и его надёжность уменьшается (значение $\alpha = 1$ означает, что отказы станка растут пропорционально времени и «старение» не происходит). Через определённое время проводится профилактика станка, продолжающаяся время $T_{\text{В2}}$, и он восстанавливает все свои характеристики надёжности до первоначального состояния. Требуется определить оптимальное время между соседними профилактиками, при котором коэффициент простоя $K_{\text{п}}$ станка был бы минимальным. Под коэффициентом простоя будем понимать долю времени, в течение которого станок не работает.

В течение периода длительности t станок простаивает время $n(t)T_{\text{В1}} + T_{\text{В2}}$, и, следовательно, коэффициент простоя станка равен:

$$K_{\text{п}} = \frac{n(t)T_{\text{В1}} + T_{\text{В2}}}{t} = \frac{cT_{\text{В1}}t^{\alpha} + T_{\text{В2}}}{t}. \qquad (3.30)$$

Задача минимизации функции (3.30) является задачей стохастического программирования, поскольку параметр α случаен.

3.2.2. РОБАСТНАЯ ОПТИМИЗАЦИЯ

Робастная оптимизация – это задача оптимизации с неопределёнными параметрами, которая отличается от стохастической оптимизации. Неопределённые параметры стохастической оптимизации – это случайные величины с детерминированным распределением вероятностей. Неопределённость робастной оптимизации означает, что в задаче оптимизации соответствующие неопределённые параметры находятся в пределах установленного диапазона и нет определённого распределения вероятностей. Параметрами здесь могут быть коэффициенты целевой функции или коэффициенты условий ограничения. Требуется, чтобы решение было приемлемым для всех возможных значений неопределённых параметров.

В идеале необходимо мгновенно принять решение, которое будет оставаться допустимым всегда, вне зависимости от обстоятельств. Люди, которые проектировали кастрюли, утюги, холодильники и т.д. в СССР делали это в канве робастной оптимизации: продукт должен работать в любых обстоятельствах. Это породило легенду, согласно которой Генри Форд произнёс: «Только такая богатая страна, как СССР, может позволить себе тратить 135 кг железа на одну кровать».

Одним из методов решения задачи является перенос неопределённостей в ограничения.

3.2.3. ТЕОРИЯ МАССОВОГО ОБСЛУЖИВАНИЯ

В теории массового обслуживания одним из основных является понятие очереди. Под очередью подразумевают линейную цепочку выстроившихся один за другим объектов, нуждающихся в том или ином виде обслуживания. Будем называть подлежащие обслуживанию объекты и обслуживающие «единицы» соответственно заявками на обслуживание (или требованиями) и обслуживающими приборами.

Примеры:

а) поступление на ремонтную базу требующих технического обслуживания транспортных средств, когда в зависимости от численности технического персонала ремонтная база может одновременно обслуживать одно или несколько транспортных средств;

б) прибытие пациентов на приём к врачу, когда даже при наличии определённой системы предварительной записи по целому ряду причин число ожидающих может флуктуировать и, таким образом, возможно образование очереди.

Во всех системах массового обслуживания имеют место следующие основные элементы.

а) *Входной поток* (*поток поступающих требований*, *или заявок на обслуживание*). Если акты поступления требований на обслуживание и процедуры обслуживания реализуются строго по графику, то очереди можно избежать. Но на практике всё происходит иначе, и в большинстве случаев поступление требований зависит от внешних обстоятельств. Следовательно, лучшее, что можно сделать, – это описать процесс поступления требований через случайные величины. К числу основных показателей, необходимых для описания входного потока, относятся характеристики источника требований, тип требований и длина интервалов времени между поступлениями требований.

б) *Механизм обслуживания*. Системы массового обслуживания различаются числом обслуживающих приборов, количеством одновременно обслуживаемых требований, продолжительностью и типом обслуживания. Например, в задачах, возникающих при рассмотрении потоков в сетях, и в задачах составления графиков работы ремонтных предприятий приходится иметь дело с несколькими обслуживающими приборами, функционирующими последовательно и(или) параллельно. Указанные выше характеристики процесса обслуживания также описываются с помощью случайных величин.

в) *Дисциплина очереди*. Все другие факторы, ассоциированные с правилами поведения очереди, можно включить в этот пункт. К их числу относятся, прежде всего, правила, в соответствии с которыми обслуживающий механизм принимает поступившую заявку к обслуживанию. Правила: «первым пришёл – первым обслуживаешься», «последним пришёл – первым обслуживаешься», «случайный отбор заявок». Во многих ситуациях с целью придания описанию исследуемого процесса большего реализма приходится прибегать к понятию «приоритет».

Классификация целевых назначений теории массового обслуживания основана на выделении в её структуре различных классов задач и, соответствен-

но, областей применения получаемых результатов. Можно выделить следующие классы задач: задачи анализа поведения системы, статистические задачи и операционные задачи.

а) *Задачи анализа поведения системы*. Цель рассмотрения задач такого рода заключается в том, чтобы с помощью математических моделей, более или менее детально отражающих свойства реальных систем массового обслуживания, выявить операционные характеристики, определяющие поведение этих систем в процессе их функционирования. К числу основных операционных характеристик любой системы массового обслуживания относятся следующие:

– $Q(t)$ – длина очереди, в момент времени t, т.е. число требований, находящихся в системе в момент времени t (некоторые авторы определяют длину очереди как число требований, ожидающих обслуживания, т.е. не включают в $Q(t)$ требования, обслуживание которых уже начато);

– Q_n – длина очереди на n-й стадии; при этом предполагается, что стадии реализуются в дискретном режиме и определяются теми или иными событиями (например, поступлением требования в систему или выбытием требования из системы);

– $W(t)$ – виртуальная продолжительность ожидания относительно момента времени t, т.е. время ожидания обслуживания для требования, которое поступит в систему в момент времени t;

– W_n – продолжительность периода, в течение которого n-е требование ожидает обслуживания;

– T_i – продолжительность периода занятости системы, начало которого соответствует $Q(0) = i$, т.е. длина периода занятости системы, начинающегося при наличии в системе i требований (при $i = 1$, так же как и при $i = 0$ в момент, непосредственно предшествующий поступлению требования на обслуживание, систему принято считать начавшей функционировать, т.е. занятой);

– I_n – продолжительность n-го периода простоя системы, т.е. длина интервала времени, в течение которого система в n-й раз оказывается незанятой (ясно, что в системах обслуживания периоды незанятости сменяются периодами занятости).

б) *Статистические задачи*. Статистическое исследование является неотъемлемой частью разработки математической модели реальной системы. В общем виде модель может существовать сама по себе, но приведение её в количественное соответствие с конкретной системой массового обслуживания достигается путём статистического анализа эмпирических данных, оценивания фигурирующих в модели параметров и проверок исходных гипотез.

Параметрами системы, по существу, являются параметры, ассоциированные с процессом поступления требований и механизмом обслуживания (либо некоторые функции этих параметров). Это утверждение нетрудно конкретизировать. Пусть требования C_1, C_2, … поступают, соответственно, в моменты времени t_1, t_2, ... Величина $U_n = t_n - t_{n-1}$ есть длина временно́го интервала между моментами поступления требований C_{n-1} и C_n. Обозначим через V_n продолжительность обслуживания требования C_n. (В случае, когда требования поступают группами, эти определения необходимо надлежащим образом модифицировать; кроме того, следует ввести понятие распределения применительно к размеру группы). Пусть $A(x) = \Pr\{U_n \le x\}$ $(x \ge 0)$ и $B(x) = \Pr\{V_n \le x\}$ $(x \ge 0)$. Таким образом, параметрами системы являются параметры функций $A(x)$ и $B(x)$, распределения вероятностей некоторых других величин (например, в случае групповых поступлений – размера групп), а также физические характеристики системы, такие, как количество обслуживающих приборов, число очередей, вместимость пространства, отведённого для требований, ожидающих обслуживания и т.д. Таким образом, статистические задачи, возникающие при исследовании систем массового обслуживания, связаны с оценкой параметров основных процессов, протекающих в системе.

в) *Операционные задачи*. Операционные задачи возникают при проектировании систем массового обслуживания, управлении реальными системами массового обслуживания и оценках их эффективности.

В зависимости от выбранных критерия и варьируемых переменных возможны самые разнообразные постановки задач массового обслуживания. Например:

1) найти количество обслуживающих приборов, при которых длина очереди будет минимальной;

2) найти количество обслуживающих приборов, при которых продолжительность ожидания обслуживания будет минимальным.

Алгоритмы решения задач массового обслуживания описаны в [1, 10].

3.2.4. ТЕОРИЯ ИГР

В позиционной игре участвует n лиц. Разрешённые ходы указаны в их логической последовательности. Каждый ход производится либо игроком (личный ход), либо выбирается случайным образом (случайный ход). В первом случае игрок выбирает ход, имея полную информацию, во втором случае задается распределение вероятностей ходов. В каждой окончательной позиции игры значение исхода можно выразить при помощи вектора $(p_1, p_2, \ldots, p_n)$, где p_i – выигрыш i-го игрока при данном исходе.

В общем случае позиционная игра может быть представлена в виде дерева с выделенным узлом (называемым начальной позицией игры). Каждый узел представляет определённую позицию игры; каждая дуга – ход в игре. Информация задается при помощи информационных множеств. Две позиции принадлежат одному и тому же информационному множеству, если один и тот же игрок, которому следует ходить в каждой из этих позиций, не может отличить одну позицию от другой.

Стратегия представляет собой некоторое правило, описывающее действия игрока, т.е. указывает, какую альтернативу следует выбирать в каждом информационном множестве. Если игрок имеет k информационных множеств, в каждом из которых содержится соответственно $j_1, j_2, \ldots, j_k$ альтернатив, то игрок располагает $j_1, j_2, \ldots, j_k$ стратегиями игры.

Если стратегии n игроков обозначить через $\sigma_1, \sigma_2, \ldots, \sigma_n$, то исход игры определён, за исключением возможных случайных ходов. Поскольку вероятности случайных ходов полностью заданы, то ожидаемый выигрыш (проигрыш) каждого игрока при таком выборе стратегий также полностью определён. Нормальной формой игры называется функция, которая задаёт вектор математического ожидания, выигрышный для каждой из n стратегий.

Таким образом, нормальная форма игры – это функция, которая ставит в соответствие каждому набору стратегий $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ вектор выигрышей $(p_1,$

p_2, …, p_n). В таком контексте p_i означает выигрыш игрока i, если применяется данный набор стратегий.

Постановка задачи: найти такую стратегию j_i для i-го игрока, при которой выигрыш игрока i будет максимальным.

Игроку обычно важно, чтобы противник не угадал, какую стратегию он будет использовать. Для осуществления этого плана следует пользоваться смешанной стратегией. В сущности, смешанная стратегия (для одного игрока) представляет собой схему случайного выбора чистой стратегии.

*Вычисление оптимальных стратегий. О*птимальные стратегии легко находятся для небольших игр, но вычисления становятся достаточно сложными с ростом числа стратегий.

Простейший метод состоит в нахождении седловой точки для чистых стратегий. Если такая седловая точка существует, то две чистые стратегии, которые к ней приводят, являются оптимальными.

Постановки задач теории игр и алгоритмы их решения рассмотрены в [1, 20].

3.2.5. ТЕОРИЯ ПРИНЯТИЯ РЕШЕНИЙ

Теория принятия решений – научная дисциплина, целью которой является помощь людям, принимающим решение в сложных условиях, для полного и объективного анализа предметной деятельности.

Поддержка принятия решений заключается в помощи лицу, принимающему решение (ЛПР), в процессе принятия решений. Она включает:

– помощь ЛПР при анализе объективной составляющей, т.е. в понимании и оценке сложившейся ситуации, и ограничений, накладываемых внешней средой;

– выявление предпочтений ЛПР, т.е. выявление и ранжирование приоритетов, учёт неопределённости в оценках ЛПР и формирование его предпочтений;

– генерацию возможных решений, т.е. формирование списка альтернатив;

– оценку возможных альтернатив, исходя из предпочтений ЛПР, и ограничений, накладываемых внешней средой;

– анализ последствий принимаемых решений;

– выбор лучшего с точки зрения ЛПР варианта.

Возможно четыре варианта задач:

1) одномерная детерминированная;
2) многомерная детерминированная;
3) одномерная с неопределённостями;
4) многомерная с неопределённостями.

Одномерная детерминированная задача. Постановка задачи. Пусть выбран один критерий, характеризующий поставленную цель и известны решения, приводящие к тем или иным последствиям, причём каждое решение приводит к строго определённому последствию (неопределённость отсутствует). Необходимо найти такие решения, при которых достигается экстремум выбранного критерия.

Многомерная детерминированная задача. Постановка задачи. Пусть выбрано n критериев, характеризующих поставленную цель и известны решения, приводящие к тем или иным последствиям, причём каждое решение приводит к строго определённым значениям выбранных критериев (неопределённость отсутствует). Необходимо найти такие решения, при которых достигается экстремум всех выбранных критериев.

В такой постановке во многих случаях задача не может быть решена, так как зачастую критерии противоречат друг другу и их экстремумы достигаются при разных решениях.

В этом случае меняется постановка задачи. Здесь используют различные методы.

а) *Метод главного критерия*. Выбирается один критерий, остальные рассматриваются как ограничения;

б) *Метод с использованием функции полезности*. Метод заключается в объединении всех критериев в один. В реальных задачах критерии, как правило, имеют различные размерности и сильно отличающиеся диапазоны изменения. В таких случаях прежде, чем объединять критерии, необходимо осуществить процедуру их нормализации. Пусть для каждого критерия R_i известны его минимальное $R_{i\min}$ и максимальное $R_{i\max}$ значения. Тогда нормализованные значения критерия примут вид:

$$\overline{R_i} = \frac{R_i - R_{i\min}}{R_{i\max} - R_{i\min}}.$$

Далее строится новый глобальный критерий (называемый «полезность»), куда в виде аддитивной или мультипликативной свёртки с весовыми коэффициентами, отражающими важность локального критерия, входят все локальные критерии.

в) *Лексикографическое упорядочение*. Упорядочение объектов осуществляется таким образом, что, например, объект A предпочитается объекту B, если он имеет бóльшую оценку по наиболее важному критерию R_1, невзирая на то, насколько он является хорошим или же плохим по другим менее важным критериям. Но если значения R_1 для объектов A и B совпадут, вводится в рассмотрение следующий по важности критерий R_2 и по нему выбирается предпочитаемый объект. Соответственно, в случае совпадения оценок по критериям R_1, R_2 вводится критерий R_3 и т.д. Термин «лексикографическое» объясняется тем, что эта процедура напоминает построение словаря.

г) *Оптимизация по Парето*. Пусть действия a_1 и a_2 приводят к последствиям, характеризующимся наборами критериев R_1^0, R_2^0, …, R_n^0 (для действия a_1) и R_1^1, R_2^1, …, R_n^1 (для действия a_2). Вводится понятие доминирование: решение a_1 доминирует над решением a_2, если:

$$R_i^0 \geq R_i^1 \text{ для всех } i;$$

$$R_i^0 > R_i^1 \text{ хотя бы для одного } i,$$

здесь имеется в виду, что все критерии максимизируются.

Решение a^* является эффективным (парето-оптимальным, недоминируемым, неподчинённым), если не существует решения a, для которого $R_i(a) \geq R_i(a^*)$, $i = 1, 2, ..., n$ и значение хотя бы одного критерия лучше (больше), нежели у a^*. Совокупность всех возможных эффективных решений a^* образует множество Парето.

Одномерная задача с неопределённостями. Принимающий решение должен выбрать одну из нескольких альтернатив (способов действий) a_1, a_2, ..., a_n,

каждая из которых приведёт к некоторому результату. Оценка предпочтительности возможных результатов осуществляется с помощью только одного критерия R. Лицо, принимающее решение, не знает к какому именно результату приведёт любая из выбранных альтернатив, но для каждого способа действий он в состоянии установить вероятности получения различных исходов. В этом случае необходимо решить одномерную задачу с неопределённостями.

Постановка задачи будет следующей. Пусть имеется m возможных последствий, которые могут быть упорядочены по их предпочтительности (например, x_1 менее предпочтительно, по сравнению с x_2, и т.д. до x_n). Пусть известно, что решение A приводит к исходам x_i с вероятностью p_i^A, $i = 1, \ldots, m$, $p_i^A \geq 0, \sum p_i^A = 1$.

Пусть действие B приводит к исходам x_i с вероятностью p_i^B, $i = 1, \ldots, m$, $p_i^B \geq 0, \sum p_i^B = 1$.

Необходимо выбрать наиболее предпочтительное решение среди A и B.

Многомерная задача с неопределённостями. Постановка многомерной задачи с неопределённостями отличается от постановки одномерной задачи с неопределённостями наличием n критериев.

Для решения поставленной задачи используются подходы, развитые при решении многокритериальных задач в условиях определённости и однокритериальных задач в условиях неопределённости. В этом случае необходимо построить многомерную функцию полезности.

Постановки задач теории принятия решений и алгоритмы их решения рассмотрены в [1, 11 – 13].

4. ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ РЕШЕНИЯ ЗАДАЧ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Анализ рассмотренных выше задач исследования операций показывает, что некоторые алгоритмы позволяют распараллелить их работу для ускорения расчётов. В качестве наиболее яркого примера можно привести решение различных задач оптимизации методом сканирования (полного перебора). Каждый вариант, рассчитываемый при реализации метода сканирования, никак не связан с остальными (в отличие, например, от методов покоординатного спуска, градиентов и т.д.). Выполнив одновременно расчёт многих вариантов, и получив значения критерия для каждого варианта, остаётся только найти в полученном массиве данных наименьший (или наибольший – в зависимости от поставленной задачи) вариант, что можно сделать достаточно быстро.

В настоящее время возможно использование нескольких технологий для организации параллельных вычислений:

- применение многопроцессорных видеокарт;
- проведение вычислений в «облаке»;
- создание вычислительного кластера из нескольких компьютеров.

Современная производительная видеокарта имеет большое количество вычислительных блоков. Например, видеокарта Nvidia GeForce RTX 3090 Ti оснащена 10 752 вычислительными блоками. Видеокарты поддерживают технологию CUDA – программно-аппаратную архитектуру параллельных вычислений, которая позволяет существенно увеличить вычислительную производительность, благодаря использованию графических процессоров фирмы Nvidia. CUDA SDK позволяет программистам реализовывать на специальных упрощённых диалектах языков программирования C, C++ и Фортран алгоритмы, выполнимые на графических и тензорных процессорах Nvidia. Архитектура CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического или тензорного ускорителя и управ-

лять его памятью. Функции, ускоренные при помощи CUDA, можно вызывать из различных языков программирования, в том числе Python.

У технологии CUDA есть множество преимуществ:

– наличие понятной документации от разработчика;
– широкий набор готовых инструментов, включающий профайлер;
– наличие готовых библиотек;
– кроссплатформенность технологии.

Кроме того, технология CUDA является бесплатной.

Работу технологии CUDA можно описать следующим образом. Графический процессор выступает в роли массивно параллельного сопроцессора к центральному процессору. Программа на CUDA задействует как центральный процессор, так и графический. При этом обычный (последовательный, т.е. непараллельный) код выполняется на центральном процессоре, а для массивно-параллельных вычислений соответствующий код выполняется на графическом процессоре как набор одновременно выполняющихся нитей (потоков) [17].

При этом очень важно понимать, что между нитями на центральном процессоре и нитями на графическом процессоре есть принципиальные различия.

1. Нити на графическом процессоре обладают крайне небольшой стоимостью создания, управления и уничтожения (контекст нити минимален, все регистры распределены заранее).

2. Для эффективной загрузки графического процессора необходимо использовать много тысяч отдельных нитей, в то время как для центрального процессора обычно достаточно 10…20 нитей.

За счёт того, что программы в CUDA пишутся фактически на обычном языке C, в котором добавлено небольшое число новых конструкций, программирование с применением технологии CUDA значительно сокращает время на разработку и повышает удобство работы с кодом, по сравнению с использованием традиционного GPGPU (техника использования графического процессора видеокарты, использующая графические программные интерфейсы приложений для доступа к графическому процессору). Также стоит отметить, что при

использовании технологии CUDA в распоряжении программиста оказывается существенно больше контроля и возможностей при работе с видеокартой.

Приведём основной алгоритм работы с технологией CUDA [18].

1. Выделяем память на графическом процессоре.
2. Копируем данные из памяти центрального процессора в выделенную память графического процессора.
3. Осуществляем запуск ядра (или последовательно запускаем несколько ядер).
4. Копируем результаты вычислений обратно в память центрального процессора.
5. Освобождаем выделенную память графического процессора.

Все нити выполняются параллельно, и каждая нить может получить информацию о себе через встроенные переменные.

Во время работы программы нити разбиваются на группы по 32 нити, называемыми warp. Только нити в пределах одного warp выполняются физически одновременно. Нити из разных warp могут находиться на разных стадиях выполнения программы. При этом управление warp прозрачно осуществляет сам графический процессор [19].

Для решения задач CUDA использует очень большое количество параллельно выполняемых нитей, при этом обычно каждой нити соответствует один элемент вычисляемых данных.

Кроме применения высокопроизводительной видеокарты в ЭВМ оператора существуют и другие способы увеличения скорости расчёта. На данный момент популярным способом проведения сложных вычислений являются вычисления в «облаке». Облачные вычисления – это предоставление вычислительных ресурсов по запросу через интернет. Ресурсами могут быть серверы, системы хранения, сети передачи данных, программное обеспечение, платформенные сервисы. Поставщики облачных услуг распределяют вычислительные ресурсы между разными заказчиками. Каждый из заказчиков получает требуемый пул мощностей или сервисов, которые при необходимости масштабируются.

Применение облачных вычислений – удобный способ наращивания вычислительных мощностей. При реализации крупных проектов и при расчёте больших массивов данных данный способ крайне актуален.

Создание вычислительного кластера из нескольких компьютеров – ещё один способ увеличения скорости вычислений при их распараллеливании. Вычислительный кластер – это набор соединённых между собой компьютеров (серверов), которые работают вместе и могут рассматриваться как единая система. В отличие от грид-вычислений, все узлы компьютерного кластера выполняют одну и ту же задачу и управляются одной системой управления.

Серверы кластера обычно соединяются между собой по быстродействующей локальной сети, причём на каждом из серверов работает собственный экземпляр операционной системы. В большинстве случаев все вычислительные узлы кластера используют одинаковое оборудование и одну и ту же операционную систему. Однако, в некоторых инсталляциях, например, с использованием платформы приложений для организации кластеров OSCAR (Open Source Cluster Application Resources), могут использоваться различные операционные системы или разное серверное оборудование. Кластеры обычно развертываются для большей производительности и доступности, чем то, что можно получить от одного компьютера, пусть даже очень мощного. Часто такое решение более экономично, чем отдельные компьютеры.

Запуск параллельных процессов в распределённой компьютерной среде можно осуществлять с помощью библиотеки MPICH C.

Отметим, что в случае использования многоядерных процессоров, объединяемых в кластеры, производительность может быть ещё более высокой.

Данный способ распараллеливания вычислений актуален для задач малой и средней размерности. Для каждой задачи необходимо обосновывать количество рабочих станций кластера, чтобы их суммарной производительности было достаточно для получения решения за требуемое время.

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа 1

РЕШЕНИЕ ЗАДАЧИ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ ДЕТЕРМИНИРОВАННЫМ МЕТОДОМ

Цель: приобретение навыков решения задач исследования операций.

Задание: произвести решение задачи исследования операций детерминированным методом.

Общие положения

Основная задача исследования операций – предварительное количественное обоснование оптимальных решений. Работая над выполнением магистерской диссертации, необходимо продумать математическую постановку задачи оптимизации, а также выбрать и обосновать метод решения оптимизационной задачи.

Порядок выполнения работы

1. Поставить задачу исследования операций для объекта, являющегося предметной областью магистерской диссертации.
2. Разработать алгоритм решения поставленной задачи одним из детерминированных методов: линейного программирования, целочисленного программирования, теории графов, геометрического программирования, нелинейного программирования, динамического программирования, оптимального управления в зависимости от поставленной задачи.
3. Подготовить программу для вычислительной машины, реализующую алгоритм из п. 2.

Содержание отчёта

Результаты решения поставленной задачи.

Лабораторная работа 2

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ ТЕХНИЧЕСКИХ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Цель: приобретение навыков решения задач исследования операций с использованием параллельных вычислений.

Задание: произвести решение задачи исследования операций с использованием параллельных вычислений.

Общие положения

При выполнении магистерской диссертации возможна ситуация, когда некоторая задача (решение систем уравнений; поиск оптимальных значений; проведение расчётов и т.д.) позволяет распараллеливание вычислений для ускорения получения результата. В этом случае необходимо решить задачу с использованием параллельных вычислений. Для этого можно применить кластер вычислительных устройств, видеокарту (использующую технологию CUDA) или облачные вычисления.

Порядок выполнения работы

1. Выявить среди задач магистерской диссертации такие, которые позволяют распараллелить вычисления.
2. Разработать алгоритм решения поставленной задачи одним из методов параллельных вычислений, используя кластер вычислительных устройств, видеокарту или облачные вычисления.
3. Подготовить программу для вычислительной машины, реализующую алгоритм из п. 2 и решить поставленную задачу.
4. Провести расчёты с использованием традиционного последовательного алгоритма и с помощью параллельных вычислений.
5. Провести сравнительный анализ результатов расчёта традиционным последовательным алгоритмом и с помощью параллельных вычислений с точки зрения затрат времени на решение задачи.

Содержание отчёта

Результаты решения поставленной задачи.

Примерные темы курсовой работы.

Вариант 1. Поставить задачу оптимизации сложного объекта по теме магистерской диссертации и разработать математическую модель объекта, необходимую для решения задачи оптимизации.

Вариант 2. Разработать параллельный алгоритм решения системы дифференциальных уравнений в частных производных математической модели сложного объекта и отладить программу для ЭВМ, реализующую разработанный алгоритм.

Вариант 3. Разработать параллельный алгоритм решения задачи оптимизации сложного объекта по теме магистерской диссертации.

Вариант 4. Разработка системы регулирования дорожного трафика (управления светофорами) для снижения времени ожидания на светофоре с использованием технологий теории массового обслуживания.

Вариант 5. Разработка системы составления расписаний методом целочисленного программирования.

Вариант 6. Разработка системы проектирования сетей связи с использованием теории графов.

ТРЕБОВАНИЯ К ОСНОВНЫМ РАЗДЕЛАМ ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ КУРСОВОЙ РАБОТЫ

1. **Введение** – отражается актуальность выбранной темы.
2. **Анализ предметной области** – приводится описание основных подходов к решению задач предметной области, проводится анализ существующих разработок и выявляются нерешённые задачи, требующие дополнительных исследований. Здесь же приводится цель работы и формулируется постановка задачи.
3. **Математические методы** – приводится описание математического аппарата, используемого для решения задачи исследования операций. Подробно описывается алгоритм решения поставленной задачи.
4. **Пример работы** – подробно рассматривается работа алгоритма для конкретного примера и анализируется полученное решение.

5. **Выводы** – приводятся основные результаты, из которых должно следовать, что цель, сформулированная в разделе «Анализ предметной области», достигнута.

6. **Литература** – указать литературные источники, используемые при выполнении курсовой работы в соответствии с ГОСТ Р 7.0.100–2018.

7. **Приложение** – распечатки кодов программ. Реализация алгоритмов допускается на любом языке программирования.

ТРЕБОВАНИЯ ДЛЯ ДОПУСКА КУРСОВОЙ РАБОТЫ К ЗАЩИТЕ

Курсовая работа должна соответствовать выбранной теме, содержать все основные разделы в соответствии с заданием, должна быть оформлена в соответствии с СТО ФГБОУ ВО «ТГТУ» 07–2017 «Выпускные квалификационные работы и курсовые проекты (работы). Общие требования».

ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ВОПРОСОВ К ЗАЧЁТУ И ЭКЗАМЕНУ

ТЕОРЕТИЧЕСКИЕ ВОПРОСЫ К ЗАЧЁТУ 2 СЕМЕСТР

1. Основные понятия и принципы исследования операций.
2. Математические модели операций.
3. Прямые и обратные задачи исследования операций.
4. Детерминированные задачи.
5. Выбор решения в условиях неопределённости.
6. Многокритериальные задачи исследования операций.
7. Линейное, нелинейное, целочисленное, динамическое программирование.
8. Марковские случайные процессы.
9. Теория массового обслуживания.
10. Статистическое моделирование случайных процессов (метод Монте-Карло).
11. Игровые методы обоснования решений.
12. Методы анализа сложных систем.
13. Свойства сложных систем.

ПРИМЕРЫ ТИПОВЫХ ПРАКТИЧЕСКИХ ЗАДАНИЙ К ЗАЧЁТУ 2 СЕМЕСТР

1. Предложите структурную схему сложного технического объекта по теме магистерской диссертации.

2. Приведите пример системы уравнений, используемых для описания сложного технического объекта по теме магистерской диссертации.

ТЕОРЕТИЧЕСКИЕ ВОПРОСЫ К ЭКЗАМЕНУ 3 СЕМЕСТР

1. Метод конечных элементов решения дифференциальных уравнений в частных производных с краевыми условиями 1 – 4 рода.
2. Сходимость разностных схем для решения дифференциальных уравнений в частных производных.
3. Основные понятия и принципы исследования операций.
4. Метод, использующий функции Грина для решения дифференциальных уравнений в частных производных с краевыми условиями 1 – 4 рода.

5. Математические модели операций.

6. Схема Кранка-Николсона для решения дифференциальных уравнений в частных производных с краевыми условиями 1 – 4 рода.

7. Прямые и обратные задачи исследования операций.

8. Структуры связи между элементами сложных систем.

9. Метод релаксации с прогонкой по строке для решения дифференциальных уравнений в частных производных с краевыми условиями 1 – 4 рода.

10. Детерминированные задачи.

11. Выбор решения в условиях неопределённости.

12. Многокритериальные задачи исследования операций.

13. Линейное программирование.

14. Нелинейное программирование.

15. Целочисленное программирование.

16. Динамическое программирование.

17. Марковские случайные процессы.

18. Теория массового обслуживания.

19. Статистическое моделирование случайных процессов (метод Монте-Карло).

20. Игровые методы обоснования решений.

21. Методы анализа сложных систем.

22. Свойства сложных систем.

23. Математические модели сложных систем.

ПРИМЕРЫ ТИПОВЫХ ПРАКТИЧЕСКИХ ЗАДАНИЙ К ЭКЗАМЕНУ 3 СЕМЕСТР

1. Разработать математическую модель объекта по теме магистерской диссертации.

2. Выявить метод решения систем уравнений, наиболее подходящий для математической модели сложного технического объекта по теме магистерской диссертации.

3. Записать алгоритм решения системы уравнений математической модели сложного технического объекта по теме магистерской диссертации.

4. Записать фрагмент программы параллельных вычислений для решения задачи моделирования сложного технического объекта по теме магистерской диссертации.

ЗАКЛЮЧЕНИЕ

В учебном пособии рассмотрены математические, количественные методы для обоснования решений во всех областях целенаправленной человеческой деятельности. Большое внимание уделено как детерминированным, так и стохастическим математическим методам, позволяющих найти наилучшие (оптимальные) решения разнообразных практических задач. Приведены методические указания по лабораторным работам и курсовому проектированию, а также даны перечень вопросов к зачёту и экзамену. Всё это поможет студентам в освоении науки «Исследование операций».

СПИСОК ЛИТЕРАТУРЫ

1. Исследование операций. В 2-х т. / под ред. Дж. Моудера, С. Элмаграби. – Москва : Мир, 1981. – Т. 1. – 712 с.
2. Писарук, Н. Н. Исследование операций / Н. Н. Писарук. – Минск : БГУ, 2015. – 304 с.
3. Бухвалова, В. В. Введение в геометрическое программирование / В. В. Бухвалова, А. С. Рогульская. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 115 с.
4. Литовка, Ю. В. Методы конечномерной оптимизации. Мультимедиа : учебное пособие / Ю. В. Литовка, Д. С. Соловьёв, В. В. Конкина. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2015. – 80 с.
5. Гераськин, М. И. Линейное программирование : учебное пособие / М. И. Гераськин, Л. С. Клентак ; под общ. ред. Л. С. Клентак. – Самара : Изд-во СГАУ, 2014. – 104 с.
6. Шевченко, В. Н. Линейное и целочисленное линейное программирование / В. Н. Шевченко, Н. Ю. Золотых. – Нижний Новгород : Изд-во Нижегородского госуниверситета им. Н. И. Лобачевского, 2004. – 154 с.
7. Бояринов, А. И. Методы оптимизации в химической технологии / А. И. Бояринов, В. В. Кафаров. – Москва : Химия, 1975. – 576 с.
8. Алексеев, В. Е. Теория графов : учебное пособие / В. Е. Алексеев, Д. В. Захарова. – Нижний Новгород : Нижегородский госуниверситет, 2017. – 119 с.
9. Литовка, Ю. В. Методы оптимизации. Вариационное исчисление (web-формат) [Электронный ресурс. Мультимедиа] : учебное пособие / Ю. В. Литовка, Д. С. Соловьёв, В. В. Конкина. – Тамбов : Изд-во ФГБОУ ВО «ТГТУ», 2016. – 80 с.
10. Солнышкина, И. В. Теория систем массового обслуживания : учебное пособие / И. В. Солнышкина. – Комсомольск-на-Амуре : ФГБОУ ВПО «КнАГТУ», 2015. – 76 с.
11. Прокопенко, Н. Ю. Системы поддержки принятия решений [Электронный ресурс] : учебное пособие / Н. Ю. Прокопенко. – Нижний Новгород : ННГАСУ, 2017. – 188 с.

12. Кини, Р. Л. Принятие решений при многих критериях: предпочтения и замещения / Р. Л. Кини, Х. Райфа ; пер. с англ. ; под ред. И. Ф. Шахнова. – Москва : Радио и связь, 1981. – 560 с.

13. Подиновский, В. В. Парето-оптимальные решения многокритериальных задач / В. В. Подиновский, В. Д. Ногин. – Москва : Наука, 1982. – 254 с.

14. Литовка, Ю. В. Расчётно-логическая интеллектуальная система управления многокатодной гальванической ванной / Ю. В. Литовка, А. А. Банников // Труды МАИ. – 2022. – № 122. – С. 1 – 17.

15. Эльсгольц, Л. Э. Вариационное исчисление : учебник / Л. С. Эльсгольц. – Москва : URSS, 2008. – 205 с.

16. Андреева, Е. А. Вариационное исчисление и методы оптимизации : учебное пособие / Е. А. Андреева, В. М. Цирулева. – Москва : Высшая школа, 2006. – 583 с.

17. Сандерс, Дж. Технология CUDA в примерах: введение в программирование графических процессов / Дж. Сандерс, Э. Кэндрот ; пер. с англ. А. А. Слинкина. – Москва : ДМК Пресс, 2013. – 32 с.

18. Боресков, А. В. Основы работы с технологией CUDA / А. В. Боресков, А. А. Харламов. – Москва : ДМК Пресс, 2013. – 232 с.

19. Козлов, С. О. Использование технологии CUDA при разработке приложений для параллельных вычислительных устройств / С. О. Козлов, А. А. Медведев // Вестник Курганского государственного университета. – 2015. – № 4. – С. 106 – 112.

20. Петросян, Л. А. Теория игр : учебник / Л. А. Петросян, Н. А. Зенкевич, Е. В. Шевкопляс. – 2-е изд., перераб. и доп. – Санкт-Петербург : БХВ-Петербург, 2012. – 432 с.

ОГЛАВЛЕНИЕ

Учебное электронное издание

ЛИТОВКА Юрий Владимирович
МАЙСТРЕНКО Наталья Владимировна
ЕГОРОВ Сергей Яковлевич

МАТЕМАТИЧЕСКИЕ МЕТОДЫ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Учебное пособие

Редактирование Е. С. Мордасовой
Компьютерное макетирование М. А. Евсейчевой
Обложка, упаковка, тиражирование Е. С. Мордасовой

ISBN 978-5-8265-2569-2

9 785826 525692

Подписано к использованию 24.04.2023.
Тираж 50 шт. Заказ № 31

Издательский центр ФГБОУ ВО «ТГТУ»
392000, г. Тамбов, ул. Советская, д. 106, к. 14
Телефон: (4752) 63-81-08
E-mail: izdatelstvo@tstu.ru