А. Л. ЗОЛКИН,
М. С. ЧИСТЯКОВ

ТЕОРИЯ ПРИНЯТИЯ РЕШЕНИЙ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

УЧЕБНОЕ ПОСОБИЕ

ЛАНЬ
САНКТ-ПЕТЕРБУРГ•МОСКВА•КРАСНОДАР
2025

УДК 519.81
ББК 22.18я73

З 79 **Золкин А. Л.** Теория принятия решений и исследование операций : учебное пособие для вузов / А. Л. Золкин, М. С. Чистяков. — Санкт-Петербург : Лань, 2025. — 124 с. : ил. — Текст : непосредственный.

ISBN 978-5-507-51750-3

Учебное пособие предназначено для углубленного изучения ключевых аспектов дисциплины «Информатика». Оно охватывает основные принципы и методы, используемые в процессе принятия решений и исследовании операций, с акцентом на практическое применение в области информационных технологий и вычислительной техники. В пособии подробно рассматриваются теоретические основы и практические методы, которые помогают студентам развивать навыки анализа и решения сложных задач, связанных с их будущей профессиональной деятельностью.

Учебное пособие предназначено для изучения дисциплины «Информатика» студентами направлений подготовки «Информационные системы и технологии», «Информатика и вычислительная техника», «Программная инженерия».

УДК 519.81
ББК 22.18я73

Рецензенты:
Ю. В. ГУМЕННИКОВА — кандидат физико-математических наук, доцент, доцент кафедры высшей математики Приволжского государственного университета путей сообщения;
А. В. ДОКУЧАЕВ — кандидат физико-математических наук, доцент, инженер II категории кафедры прикладной математики и информатики Института автоматики и информационных технологий Самарского государственного технического университета.

Обложка
П. И. ПОЛЯКОВА

ОГЛАВЛЕНИЕ

АВТОРСКИЙ КОЛЛЕКТИВ

А. Л. Золкин, доцент кафедры «Информатика и вычислительная техника» ФГБОУ ВО «Поволжский государственный университет телекоммуникаций и информатики», канд. техн. наук, доц.

М. С. Чистяков, научный сотрудник Автономной некоммерческой образовательной организации высшего образования Центросоюза Российской Федерации «Российский университет кооперации», г. Владимир.

ВВЕДЕНИЕ

В современном мире, характеризующемся высоким уровнем неопределенности и сложностью систем, навыки принятия обоснованных решений становятся критически важными. Теория принятия решений предоставляет методы и инструменты, которые помогают анализировать ситуацию, оценивать возможные варианты и выбирать оптимальные действия. Исследование операций, в свою очередь, предлагает математические и вычислительные подходы к решению задач оптимизации и управления ресурсами.

Исследование операций, в свою очередь, предлагает математические и вычислительные подходы к решению задач оптимизации и управления ресурсами. Этот раздел науки занимается разработкой и применением количественных методов для анализа и оптимизации сложных систем и процессов. Основные инструменты исследования операций включают линейное и нелинейное программирование, целочисленное и динамическое программирование, а также стохастическое программирование [1].

Линейное программирование позволяет решать задачи, где цель состоит в максимизации или минимизации линейной целевой функции при наличии линейных ограничений. Этот метод широко применяется для оптимизации производственных процессов, распределения ресурсов и планирования логистики. Нелинейное программирование расширяет возможности линейного программирования, учитывая нелинейные зависимости в задачах, что позволяет решать более сложные и реалистичные модели.

Целочисленное программирование используется для решения задач, в которых некоторые или все переменные должны принимать только целочисленные значения. Это особенно важно в задачах, связанных с планированием и распределением, где ресурсы не могут быть делимыми. Динамическое программирование, в свою очередь, позволяет решать задачи многократного принятия решений путем разбиения их на более простые подзадачи и последовательного их решения [2].

Стохастическое программирование занимается задачами, в которых присутствует неопределенность в данных или параметрах модели.

Этот метод позволяет учитывать возможные вариации и вероятностные сценарии, что делает его незаменимым в условиях, где решения принимаются на основе неполной или неточной информации.

Пособие охватывает широкий спектр тем, начиная с теоретических основ и заканчивая практическими методами, применимыми в реальных условиях. Оно предназначено для студентов различных направлений подготовки, включая информационные системы и технологии, информатику и вычислительную технику, программную инженерию, инфокоммуникационные технологии и системы связи, радиотехнику и безопасность. Данное учебное пособие также может быть полезно при подготовке выпускных квалификационных работ, предоставляя необходимые инструменты для самостоятельных исследований и разработки решений.

Основная цель пособия — помочь студентам развить навыки анализа и принятия решений, а также применить теоретические знания на практике. Это способствует формированию у них системного подхода к решению профессиональных задач, что является ключевым фактором успешной профессиональной деятельности в области информационных технологий и смежных дисциплин [3].

Системный подход подразумевает умение анализировать сложные проблемы в их целостности, учитывая множество взаимосвязанных факторов и аспектов. Он позволяет специалистам разрабатывать комплексные решения, которые не только эффективно решают текущие задачи, но и способствуют устойчивому развитию и адаптации систем к изменяющимся условиям.

Понимание и применение методов теории принятия решений и исследования операций дает студентам возможность эффективно справляться с вызовами, возникающими в профессиональной деятельности. Они учатся выявлять ключевые элементы проблем, строить математические модели, анализировать результаты и выбирать оптимальные стратегии. Это особенно важно в условиях динамично развивающейся технологической среды, где быстрые и точные решения могут существенно повлиять на конкурентоспособность и успех организации.

Кроме того, применение данных методов способствует улучшению качества планирования и управления проектами, оптимизации использования ресурсов и снижению рисков. Это делает специалистов более компетентными и востребованными на рынке труда, готовыми к решению самых сложных и нестандартных задач.

ГЛАВА 1
ОСНОВЫ ТЕОРИИ ПРИНЯТИЯ РЕШЕНИЙ

1.1. Определение и классификация решений

Принятие решений является фундаментальным процессом, лежащим в основе любой управленческой деятельности. Под принятием решений понимается выбор наилучшего из возможных вариантов действий, направленных на достижение поставленных целей. Этот процесс включает в себя идентификацию проблемы, формулирование критериев для оценки возможных решений, генерацию альтернатив, их анализ и выбор оптимального варианта. Важность принятия решений трудно переоценить, так как от их качества зависят эффективность и успешность деятельности любой организации.

Решения можно классифицировать по различным признакам (рис. 1).

Рис. 1

По степени структурированности выделяют структурированные, полуструктурированные и неструктурированные решения

Структурированные решения принимаются в условиях четко определенных параметров и алгоритмов действий, таких как задачи инвентаризации или расчет заработной платы. Полуструктурированные решения включают элементы как определенности, так и неопределенности, например, маркетинговые стратегии или планирование производства. Неструктурированные решения связаны с уникальными и редко повторяющимися ситуациями, такими как разработка новых продуктов или управление кризисными ситуациями.

По уровню принятия выделяют стратегические, тактические и оперативные решения. Стратегические решения определяют долгосрочные цели и направления развития организации, например, выход на новые рынки или внедрение инновационных технологий. Тактические решения связаны с реализацией стратегий и направлены на среднесрочные цели, такие как оптимизация производственных процессов или улучшение качества продукции [4, 29–34].

Оперативные решения касаются текущей деятельности и направлены на решение ежедневных задач, например, распределение ресурсов или урегулирование конфликтов.

Оперативные решения играют ключевую роль в управлении текущей деятельностью организации и направлены на решение повседневных задач, которые возникают в процессе ее функционирования. Эти решения охватывают широкий спектр операций, начиная от управления ресурсами и заканчивая урегулированием конфликтов между сотрудниками или между отделами. Примером оперативного решения может быть ситуация, когда менеджер по производству быстро реагирует на изменения в заказах от клиентов, перераспределяя рабочую силу и материальные ресурсы для соблюдения сроков выполнения заказов. Эти решения часто принимаются на основе текущей информации и требуют оперативного анализа ситуации, чтобы минимизировать потери и оптимизировать процессы [5, 35–38].

В частности, математическое следование в данном контексте означает применение методов оптимизации и анализа данных для принятия решений.

Представим, что менеджер сталкивается с неожиданной задержкой поставки компонентов. Для минимизации возможных потерь он может использовать математические модели, которые учитывают текущие запасы, сроки поставки и потребности производства. Например, можно построить математическую модель, которая оптимизирует распределение ресурсов и задействует альтернативных поставщиков в зависимости от их цен и сроков поставки.

Выразим детерминированный подход к управлению производственными процессами в алгоритмической структуре на языке *R*.

Допустим, нужно написать алгоритм, который поможет менеджеру по снабжению принять решение об оптимальном распределении ресурсов и управлении задержками поставок. Ниже представлен пример такого алгоритма.

Заданные данные (демонстрационные значения):

```
current_inventory <- 1000   текущий запас компонентов
expected_demand <- 500      ожидаемый спрос
delayed_delivery <- TRUE    наличие задержки поставки
additional_cost <- 500      дополнительные затраты на ре-
шение проблемы
```

Шаг 1. Анализ текущей ситуации и данных.

```
if (delayed_delivery) {
```

Шаг 2. Оценка последствий задержки.

```
  impact <- current_inventory - expected_demand
```

```
  if (impact > 0) {
```

Шаг 3. Принятие решения об управлении запасами.

```
    optimal_decision <- "Приобретение дополнительных ком-
понентов"
```

Шаг 4. Расчет дополнительных затрат.

```
    total_cost <- additional_cost
  } else {
    optimal_decision <- "Оптимизация текущих запасов"
    total_cost <- 0
  }
} else {
  optimal_decision <- "Нет задержек, текущая стратегия"
  total_cost <- 0
}
```

Шаг 5. Вывод результатов принятого решения.

```
cat("Оптимальное решение:", optimal_decision, "\n")
cat("Общие затраты:", total_cost, "\n")
```

В этом примере:

1) мы начинаем с анализа текущей ситуации (текущий запас, ожидаемый спрос, наличие задержки поставки);

2) если есть задержка поставки, мы оцениваем ее влияние на запасы и принимаем решение об управлении запасами;

3) в зависимости от оценки влияния (impact) мы принимаем оптимальное решение (покупка дополнительных компонентов или оптимизация текущих запасов);

4) дополнительно рассчитываем затраты на реализацию выбранного решения;

5) выводим результаты принятого решения (оптимальное решение и общие затраты).

Этот пример демонстрирует, как можно структурировать алгоритм на языке *R* для поддержки процесса принятия решений в условиях детерминированного управления производственными процессами.

Эти детерминированные конституенты означают, что решения и составляющие процесса принятия решений основаны на четко определенных данных, фактах и условиях. В контексте оперативного анализа ситуации и принятия решений в условиях задержек поставок компонентов детерминированный подход предполагает использование точных численных данных о текущих запасах, ожидаемых сроках поставок и их влиянии на производственные процессы.

Тогда мы будем моделировать ситуацию с задержкой поставки компонентов и принимать решение о том, нужно ли закупать дополнительные компоненты или можно обойтись текущими запасами.

Заданные данные (демонстрационные значения):

```
current_inventory <- 1000   текущий запас компонентов
expected_demand <- 800      ожидаемый спрос
expected_delivery <- 600    ожидаемая задержка поставки
```

Шаг 1. Оценка влияния задержки на производственные процессы.

```
impact <- current_inventory - expected_demand
```

Шаг 2. Принятие решения на основе детерминированного подхода.

```
if (impact < 0) {
```

Недостаточно компонентов, требуется дополнительная закупка:

```
  aditional_purchase <- expected_demand - cur-
rent_inventory
  cat("Требуется закупка дополнительных компонентов:",
additional_purchase, "\n")
} else {
```

Достаточно компонентов, можно обойтись текущим запасом:

```
  cat("Текущий запас компонентов достаточен для покрытия
спроса\n")
}
```

Шаг 3. Вывод оценки влияния и принятого решения:

```
cat("Оценка влияния задержки на производственные процес-
сы:", impact, "\n")
```

В этом примере:

1) мы оцениваем влияние задержки поставки на текущие запасы компонентов и ожидаемый спрос;

2) в зависимости от результата оценки (impact) принимается решение о необходимости дополнительной закупки компонентов или о том, что текущих запасов достаточно;

3) выводится оценка влияния задержки на производственные процессы и принятое решение.

Этот пример демонстрирует применение детерминированного подхода для оперативного анализа и принятия решений в условиях нестабильности поставок компонентов.

Например, менеджер может использовать детерминированные модели для расчета оптимального расписания производства, учиты-

вая известные сроки поставок компонентов и прогнозируемый объем спроса на конечную продукцию.

Это позволяет минимизировать риски нехватки или излишков в запасах, оптимизировать загрузку производственных мощностей и сокращать операционные издержки.

Детерминированный подход также предполагает, что принимаемые решения основаны на точных математических расчетах и анализе, который проводится на основе актуальных данных о текущем состоянии производственных процессов, потребностях рынка и условиях поставок.

Затем мы будем моделировать ситуацию, где необходимо принять решение о закупке компонентов на основе текущих запасов, потребностей рынка и условий поставок.

Заданные данные (демонстрационные значения):

```
current_inventory <- 1000    текущий запас компонентов
expected_demand <- 800       ожидаемый спрос
expected_delivery <- 600     ожидаемая задержка поставки
additional_cost_per_unit <- 50   дополнительная стоимость
за единицу компонента
```

Шаг 1. Оценка влияния задержки на производственные процессы.

```
impact <- current_inventory - expected_demand
```

Шаг 2. Принятие решения на основе детерминированного подхода.

```
if (impact < 0) {
```

Недостаточно компонентов, требуется дополнительная закупка:

```
  additional_purchase <- expected_demand - cur-
rent_inventory
  total_additional_cost <- additional_purchase addition-
al_cost_per_unit
  cat("Требуется закупка дополнительных компонентов:",
additional_purchase, "\n")
  cat("Дополнительные затраты:", total_additional_cost,
"\n")
} else {
```

Достаточно компонентов, можно обойтись текущим запасом:

```
  cat("Текущий запас компонентов достаточен для покрытия
спроса\n")
}
```

Шаг 3. Вывод оценки влияния и принятого решения.

```
cat("Оценка влияния задержки на производственные процес-
сы:", impact, "\n")
```

В данном примере:

1) мы оцениваем влияние задержки поставки на текущие запасы компонентов и ожидаемый спрос;

2) если текущих запасов недостаточно для покрытия спроса (impact < 0), рассчитываем необходимое количество дополнительных компонентов и расходы на их закупку;

3) выводим сообщение о необходимости или достаточности текущих запасов и оценку влияния задержки на производственные процессы.

Этот пример иллюстрирует использование детерминированного подхода для принятия оперативных решений на основе точных математических расчетов и анализа текущей ситуации производственных процессов.

Далее, используя математическое программирование, можно разработать модель, которая учитывает различные сценарии и принимает решение на основе оптимизации целевой функции, например, минимизации затрат или максимизации эффективности производства при заданных ограничениях.

Это может включать применение методов линейного программирования, динамического программирования или других алгоритмов оптимизации.

Для демонстрации принципов математического программирования и оптимизации давайте рассмотрим пример использования линейного программирования (LP) для оптимизации распределения ресурсов в условиях ограниченных поставок компонентов.

В данном примере мы будем решать задачу оптимизации, где требуется минимизировать общие затраты на закупку компонентов при заданных ограничениях на запасы и спрос.

Подключение библиотеки для линейного программирования:

```
library(lpSolve)
```

Заданные данные (демонстрационные значения):

```
demand <- c(800, 700, 900)    спрос на компоненты
supply <- c(1000, 500, 1200)   доступные запасы компонен-
тов
cost <- matrix(c(10, 8, 12,    стоимость компонентов от
разных поставщиков
                 9, 7, 11,
                 11, 9, 10), nrow = 3, byrow = TRUE)
```

Шаг 1. Определение целевой функции для минимизации затрат.

```
obj <- cost   матрица стоимостей
```

Шаг 2. Определение ограничений (спрос не должен превышать доступные запасы).

```
con <- rbind(demand, diag(length(demand)))
```

Шаг 3. Решение задачи линейного программирования.

```
result <- lp(direction = "min", objective.in = obj,
const.mat = con, const.dir = "<=", const.rhs = supply)
```

Шаг 4. Вывод результатов оптимизации.

```
cat("Оптимальное распределение закупок компонентов:\n")
for (i in 1:length(demand)) {
  cat("Компонент", i, ": закупка", result$solution[i],
"единиц\n")
}
cat("Общие затраты:", result$objval, "\n")
```

В этом примере:

1) мы определяем матрицу стоимостей cost, где каждый элемент указывает стоимость компонента от разных поставщиков;

2) demand и supply представляют спрос и доступные запасы для каждого компонента соответственно;

3) мы используем функцию lp() из библиотеки lpSolve, чтобы минимизировать общие затраты на закупку компонентов при соблюдении ограничений на запасы и спрос;

4) выводим оптимальное распределение закупок компонентов и общие затраты на эти закупки.

Предположим, что у нас есть следующие входные данные.

1. Спрос на компоненты: 800, 700, 900 единиц соответственно.

2. Доступные запасы компонентов: 1000, 500, 1200 единиц соответственно.

3. Стоимость компонентов от разных поставщиков:

10 8 12
9 7 11
11 9 10

После выполнения оптимизации с использованием линейного программирования результаты могут выглядеть следующим образом.

1. Оптимальное распределение закупок компонентов.

Компонент 1: закупка 800 единиц.

Компонент 2: закупка 500 единиц.

Компонент 3: закупка 900 единиц.

2. Общие затраты: 18 600.

Это означает, что для минимизации общих затрат на закупку компонентов при заданных ограничениях на доступные запасы и спрос оптимальным решением является закупка 800 единиц компонента 1, 500 единиц компонента 2 и 900 единиц компонента 3. Общие затраты составляют 18 600 (в единицах стоимости, указанных в условии).

Таким образом, приведенный пример демонстрирует, как математическое программирование, в частности линейное программирование, может использоваться для оптимизации принятия решений в условиях неопределенности и ограниченных ресурсов.

Этот пример демонстрирует использование метода линейного программирования для оптимизации решений в условиях ограниченных ресурсов и спроса, что соответствует описанному в тексте использованию математического программирования для принятия решений в оперативном управлении [6, 39–44].

Важно отметить, что оперативные решения часто принимаются в условиях ограниченного времени и ресурсов, поэтому способность быстро анализировать ситуацию и принимать обоснованные решения является критически важной компетенцией для управленцев на всех уровнях организации.

Представим, что менеджер по снабжению в компании, производящей электронные устройства, сталкивается с неожиданной задержкой поставки ключевых компонентов от текущего поставщика. Эти компоненты необходимы для производства важной партии продукции, которая должна быть поставлена заказчику в срок.

Первым шагом менеджер должен оценить потенциальные последствия задержки на производственные линии. Это включает анализ возможных временных и финансовых потерь из-за остановки или замедления производственного процесса. Менеджер должен учитывать не только сам факт задержки, но и ее влияние на график выполнения заказов и удовлетворение клиентов.

Далее следует пересмотр планов производства с целью минимизации негативных последствий от задержки. Это может включать перераспределение ресурсов, изменение приоритетов производства или переговоры с клиентами о возможности временного сдвига сроков поставки.

Важным шагом является поиск альтернативных источников поставок компонентов.

Менеджер должен искать других поставщиков, которые могут предложить аналогичные компоненты в кратчайшие сроки, либо ис-

следовать возможность использования альтернативных технологий или материалов для временной замены задерживающихся компонентов.

Формализация такой модели обслуживания включает в себя не только оперативные действия по управлению текущей ситуацией, но и стратегическое планирование для минимизации рисков и обеспечения непрерывности производственных процессов. Это требует от менеджера способности быстро анализировать ситуацию, принимать обоснованные решения и взаимодействовать с различными внутренними и внешними стейкхолдерами для достижения оптимального результата.

Он должен проанализировать, какие конкретно операции или этапы производства могут быть задержаны из-за отсутствия необходимых материалов и как это может повлиять на общий график выполнения заказов. Далее менеджер должен пересмотреть текущие планы производства и адаптировать их в соответствии с новыми условиями. Это может включать перераспределение ресурсов, изменение приоритетов выполнения заказов или привлечение дополнительных ресурсов для минимизации влияния задержки на производственные процессы.

Одновременно менеджер должен начать поиск альтернативных источников поставок компонентов.

Это включает поиск новых поставщиков, которые могут предложить аналогичные компоненты или заменить их с минимальными изменениями в производственных процессах.

Важно оценить не только качество и цену альтернативных материалов, но и их доступность в необходимые сроки.

Кроме того, менеджер должен организовать коммуникацию с другими ключевыми отделами, такими как отделы производства и логистики, чтобы обеспечить согласованное действие всех сторон и минимизировать потенциальные негативные последствия задержки для бизнеса.

Еще одним примером оперативного решения может быть ситуация в обслуживающем отделе, где менеджер должен оперативно решать конфликты между клиентами и обеспечивать удовлетворение их потребностей в режиме реального времени. Это требует быстрого анализа обстоятельств, способности к принятию решений в условиях неопределенности и гибкости в реагировании на изменяющиеся условия.

Решение оперативных задач в организации требует системного подхода и алгоритмического решения, особенно в условиях ограниченного времени и ресурсов.

Первым шагом является тщательный анализ текущей ситуации, включая сбор всех необходимых данных и выявление основной проблемы или задачи, требующей решения. Этот этап необходим для точного определения целей и критериев, которые должны быть достигнуты через оперативные действия.

Далее следует разработка и оценка альтернативных вариантов действий. Это включает генерацию возможных решений и их анализ с точки зрения их соответствия поставленным целям, эффективности и ресурсной доступности. Оценка альтернатив позволяет выбрать оптимальное решение, которое наилучшим образом соответствует текущей ситуации и требованиям организации.

Оценка альтернативных решений играет ключевую роль в процессе принятия оперативных решений, направленных на решение текущих задач организации. Этот этап включает в себя несколько важных аспектов, которые помогают выбрать оптимальное решение, соответствующее текущей ситуации и требованиям организации.

Во-первых, оценка альтернативных решений начинается с анализа их потенциальной способности решить выявленные проблемы или задачи. Каждая альтернатива должна быть оценена с точки зрения ее эффективности в достижении поставленных целей и решении основных задач [7, 45–52].

Во-вторых, оценка включает анализ ресурсов, необходимых для внедрения каждой альтернативы.

Это включает финансовые, временные и человеческие ресурсы, которые могут потребоваться для успешной реализации решения.

Третий аспект оценки связан с анализом рисков и возможных негативных последствий, связанных с каждой альтернативой. Это помогает предвидеть потенциальные проблемы или ограничения, которые могут возникнуть в процессе реализации решения, и разработать стратегии их управления.

Кроме того, оценка включает сравнение альтернативных решений между собой с точки зрения их стоимости, сложности внедрения и потенциальной выгоды для организации. Это позволяет выбрать оптимальное решение, которое обеспечит наилучший баланс между достижением целей, эффективностью использования ресурсов и минимизацией рисков.

Во-первых, при оценке альтернативных решений учитывается их стоимость. Это включает не только прямые финансовые затраты на реализацию решения, но и связанные с ним операционные издержки. Например, менеджер должен учитывать стоимость приобретения новых компонентов или услуг от новых поставщиков, а также затра-

ты на переобучение персонала или адаптацию производственных процессов.

Во-вторых, оценка включает анализ сложности внедрения каждой альтернативы. Это важный аспект, так как решение может быть технически или организационно сложным для внедрения в текущие рабочие процессы. Например, изменение поставщика может потребовать пересмотра контрактов и переговоров с новыми поставщиками, что может занять значительное время и ресурсы.

Третий аспект оценки связан с потенциальной выгодой для организации от каждой альтернативы. Это включает ожидаемые выгоды или преимущества, которые может принести решение. Например, использование более надежного поставщика может снизить риск задержек и улучшить качество производственных процессов, что в итоге может привести к повышению уровня обслуживания клиентов и увеличению прибыли [8, 53–58].

При анализе более надежного поставщика, который известен своей надежностью и стабильностью поставок, компания может ожидать значительного снижения рисков задержек и простоев в производственных процессах.

Это может привести к улучшению планирования производства, повышению эффективности использования ресурсов и сокращению времени, затраченного на устранение возникших проблем.

Улучшение качества производственных процессов также является ключевым аспектом использования более надежного поставщика.

Надежные компоненты, поставляемые вовремя и с соответствующим качеством, могут значительно снизить вероятность брака и несоответствия стандартам, что, в свою очередь, снизит затраты на возвраты и замены продукции.

При анализе более надежного поставщика, который известен своей надежностью и стабильностью поставок, компания может ожидать значительного снижения рисков задержек и простоев в производственных процессах.

Это может привести к улучшению планирования производства, повышению эффективности использования ресурсов и сокращению времени, затраченного на устранение возникших проблем.

Улучшение качества производственных процессов также является ключевым аспектом использования более надежного поставщика.

В условиях современного бизнеса надежность поставок компонентов играет решающую роль в обеспечении качества производственных процессов и удовлетворения потребностей клиентов. Проблемы с качеством компонентов, вызванные нестабильностью или

несоответствием стандартам предыдущего поставщика, могут иметь серьезные последствия для компании. Например, частые случаи брака могут привести к увеличению затрат на возвраты и замены продукции, а также повлиять на общую репутацию бренда. Поэтому переход к использованию более надежного поставщика, который известен своей стабильностью и высоким уровнем качества поставок, может стать стратегическим шагом для улучшения ситуации. Такой поставщик не только снизит вероятность возникновения проблем с качеством, но и способствует повышению эффективности производственных процессов и улучшению общего уровня обслуживания клиентов. Внедрение новых компонентов требует тщательного планирования и тестирования, чтобы минимизировать риски и обеспечить плавный переход к новому поставщику без существенных прерываний в работе компании [9, 59–63].

Повышение уровня обслуживания клиентов также является важным результатом выбора более надежного поставщика. Клиенты получают заказы в срок, что укрепляет их доверие к компании и способствует сохранению лояльности. Улучшенное обслуживание, в свою очередь, может привести к увеличению объемов продаж и росту прибыли компании.

Представим сценарий компании, производящей медицинское оборудование, которая перешла к использованию более надежного поставщика ключевых компонентов. Ранее компания сталкивалась с проблемами задержек поставок и нестабильным качеством компонентов, что приводило к неудовлетворенности клиентов и потере репутации.

В результате выбора нового надежного поставщика, который гарантирует стабильные поставки высококачественных компонентов, компания смогла значительно улучшить уровень обслуживания клиентов. Заказы начали поступать в срок, что снизило время ожидания клиентов и повысило их удовлетворенность. Клиенты стали ощущать надежность и ответственность компании перед ними, что способствовало укреплению их доверия и лояльности. Улучшенное обслуживание клиентов также положительно сказалось на финансовых показателях компании. Увеличение уровня лояльности клиентов и положительные отзывы способствовали привлечению новых заказчиков и увеличению объемов продаж медицинского оборудования. Компания смогла не только вернуть ранее потерянных клиентов, но и расширила свой рынок за счет улучшения репутации на рынке [10].

Таким образом, выбор более надежного поставщика компонентов не только помог решить текущие проблемы с качеством и за-

держками поставок, но и привел к значительному улучшению обслуживания клиентов и росту финансовых показателей компании. Этот кейс иллюстрирует, как стратегическое решение в выборе поставщика может повлиять на развитие бизнеса и укрепление его конкурентных позиций на рынке.

Важно также учитывать риски, связанные с каждой альтернативой. Это могут быть потенциальные негативные последствия или неожиданные проблемы, которые могут возникнуть в процессе реализации выбранного решения.

Например, переход к новому поставщику может повлечь за собой необходимость внесения изменений в производственные процессы, что может привести к временным неудобствам или дополнительным затратам.

Кроме того, решения можно классифицировать по характеру проблемы, к которой они относятся: управленческие, технические, экономические и социальные.

Управленческие решения связаны с организацией и координацией деятельности, технические — с инженерными и технологическими аспектами, экономические — с финансовыми и ресурсными вопросами, а социальные — со взаимодействием и коммуникацией внутри и вне организации.

Таким образом, понимание определения и классификации решений является важным шагом для эффективного управления и оптимизации процессов в любой сфере деятельности. Это позволяет формировать более четкие и структурированные подходы к решению разнообразных задач, что, в свою очередь, способствует достижению высоких результатов и устойчивому развитию организаций.

1.2. Процесс принятия решений

Процесс принятия решений играет ключевую роль в управлении и стратегическом планировании организаций. Он представляет собой системный подход к анализу, оценке и выбору наилучшей альтернативы из доступных вариантов. В контексте операционного менеджмента и исследования операций процесс принятия решений включает несколько этапов [11, 64–69].

Сначала осуществляется определение проблемы или цели, требующей принятия решения. Это включает четкое формулирование задачи и выявление ключевых факторов, влияющих на решение.

На этом этапе также происходит выявление ключевых факторов, которые могут повлиять на решение проблемы. Эти факторы могут

включать различные аспекты, такие как экономические условия, технологические требования, социальные и культурные аспекты, законодательные нормы, а также внутренние процессы и ресурсы организации [12].

Примером может быть ситуация в компании, где руководство сталкивается с проблемой неэффективного использования ресурсов в производственном процессе из-за недостаточной автоматизации.

Формулировка задачи будет заключаться в определении, какие именно аспекты производственного процесса требуют оптимизации и какие ресурсы могут быть выделены для внедрения новых технологий.

Рассмотрим пример формулировки задачи и выявления ключевых факторов с использованием кода на языке Python. В данном примере мы будем использовать библиотеку pandas для работы с данными и numpy для вычислений.

Предположим, у нас есть набор данных о производственных процессах компании и мы хотим определить проблему с неэффективным использованием ресурсов в одном из заводов:

```
import pandas as pd
import numpy as np
```

Создаем пример набора данных о производственных процессах:

```
data = {
    'Завод': ['Завод A', 'Завод B', 'Завод C', 'Завод
D'],
    'Выручка, млн': [50, 40, 55, 48],
    'Себестоимость, млн': [30, 35, 40, 32],
    'Кол-во сотрудников': [100, 80, 120, 90],
    'Степень автоматизации, %': [60, 50, 70, 55]
}

df = pd.DataFrame(data)
```

Выводим данные для анализа:

```
print("Данные о производственных процессах компании:")
print(df)
```

Формулируем задачу — определение проблемы с неэффективным использованием ресурсов:

```
print("\nФормулировка задачи:")
print("Один из заводов (например, Завод B) имеет высокую
себестоимость и низкую выручку, что может указывать на
проблемы с эффективностью производства.")
```

Выявляем ключевые факторы, влияющие на решение:

```
print("\nВыявление ключевых факторов:")
```

```
print("1. Себестоимость производства завода B значительно выше, чем у других заводов.")
print("2. Низкая степень автоматизации процессов на заводе B может приводить к дополнительным операционным расходам.")
print("3. Высокий уровень затрат на персонал на заводе B, несмотря на меньшее количество сотрудников, может также указывать на неэффективное использование ресурсов.")
```

Дополнительный анализ или расчеты могут проводиться с использованием numpy или других инструментов для более глубокого понимания данных и факторов (рис. 2).

Пример вывода рекомендаций по дальнейшим шагам в анализе и принятии решений:

```
print("\nРекомендации по дальнейшим шагам:")
print("1. Провести детальный анализ структуры затрат и процессов на заводе B.")
print("2. Рассмотреть возможные меры по улучшению автоматизации производственных процессов.")
print("3. Исследовать потенциальные преимущества инвестиций в снижение себестоимости и повышение эффективности на заводе B.")
```

```
Данные о производственных процессах компании:
     Завод  Выручка, млн  Себестоимость, млн  Кол-во сотрудников  Степень автоматизации, %
0  Завод A            50                  30                 100                        60
1  Завод B            40                  35                  80                        50
2  Завод C            55                  40                 120                        70
3  Завод D            48                  32                  90                        55

Формулировка задачи:
Один из заводов (например, Завод B) имеет высокую себестоимость и низкую выручку, что может указывать на проблемы с эффе
ктивностью производства.

Выявление ключевых факторов:
1. Себестоимость производства завода B значительно выше, чем у других заводов.
2. Низкая степень автоматизации процессов на заводе B может приводить к дополнительным операционным расходам.
3. Высокий уровень затрат на персонал на заводе B, несмотря на меньшее количество сотрудников, может также указывать на
неэффективное использование ресурсов.
```

Рис. 2

Исполненный код, демонстрирующий начальный этап процесса принятия решений, включая формулировку задачи и выявление ключевых факторов на основе анализа данных о производственных процессах компании

Этот пример иллюстрирует, как можно использовать программирование для формулировки задачи и выявления ключевых факторов, которые могут влиять на принятие решений в управлении производственными процессами.

Далее, для полного понимания проблемы или цели часто используется анализ SWOT (анализ сильных и слабых сторон, возможностей и угроз), который позволяет оценить внутренние и внешние факторы, влияющие на принятие решения. Это помогает выявить по-

тенциальные преимущества и ограничения каждой альтернативы решения.

Использование этого инструмента позволяет систематизировать информацию, оценить текущее положение дел и подготовиться к принятию обоснованных решений.

Аспектология такого анализа включает в себя разделение всех факторов на четыре основные категории: сильные стороны, слабые стороны, возможности и угрозы. Сильные стороны отражают внутренние преимущества и конкурентные преимущества компании. Слабые стороны указывают на внутренние недостатки или ограничения, которые могут мешать достижению целей. Возможности представляют внешние факторы, которые могут быть использованы в пользу организации для достижения успеха. Угрозы, напротив, представляют внешние факторы, которые могут негативно повлиять на организацию и ее деятельность [13, 70–74].

Исходя из нашего примера с анализом производственных процессов завода В, можно сформулировать следующую гипотезу.

Гипотеза: внедрение улучшенных производственных технологий на заводе В позволит снизить себестоимость и увеличить выручку компании за счет улучшения эффективности производственных процессов.

Анализ SWOT подтверждает данную гипотезу следующим образом.

1. Сильные стороны (Strengths):

– компетентный персонал и высокое качество производства;

– наличие существующей инфраструктуры и технологий.

2. Слабые стороны (Weaknesses):

– высокая себестоимость производства;

– низкая степень автоматизации процессов.

3. Возможности (Opportunities):

– развитие и внедрение новых технологий для улучшения производственных процессов;

– возможность улучшить уровень автоматизации и оптимизировать затраты на персонал.

4. Угрозы (Threats):

– конкуренция на рынке с более эффективными производственными методами;

– экономическая нестабильность и изменения внешней среды.

На основе данного анализа мы предполагаем, что внедрение улучшенных производственных технологий на заводе В поможет снизить себестоимость, что повысит конкурентоспособность компа-

нии за счет улучшения эффективности производственных процессов и сокращения операционных расходов.

Это также позволит лучше использовать возможности роста рынка и снизить угрозы, связанные с конкурентной средой и экономической нестабильностью.

Таким образом, гипотеза заключается в том, что оптимизация производственных процессов на заводе В через внедрение новых технологий приведет к улучшению финансовых показателей и укреплению позиций на рынке [14, 75–77].

Чтобы доказать гипотезу о том, что внедрение улучшенных производственных технологий на заводе В приведет к снижению себестоимости и увеличению выручки компании, можно выразить частные предположения, основанные на анализе SWOT и текущих данных о производственных процессах.

Сначала улучшение производственных технологий может снизить себестоимость за счет оптимизации процессов и сокращения операционных затрат, что является ключевым фактором для повышения экономической эффективности предприятия. Это основывается на высоком качестве производства и возможности автоматизировать процессы, что приведет к сокращению затрат на сырье, энергоресурсы и трудовые ресурсы.

Второе предположение заключается в том, что увеличение выручки достигается за счет улучшенной эффективности производства, улучшения качества продукции и сокращения времени производства.

Это повысит производственные объемы и удовлетворит спрос, что в свою очередь приведет к увеличению выручки компании.

Третье предположение подразумевает, что новые технологии также способствуют повышению конкурентоспособности компании, что проявляется в улучшении оперативности и гибкости реагирования на изменения на рынке. Это будет способствовать удержанию текущих клиентов и привлечению новых, что в свою очередь увеличит долю рынка и обеспечит рост выручки.

Все эти предположения основаны на анализе SWOT, который выявил сильные и слабые стороны компании, а также возможности и угрозы внешней среды. Для дальнейшего подтверждения гипотезы требуется дополнительный анализ данных и, возможно, проведение пилотных проектов по внедрению новых технологий на заводе В.

Далее в процессе принятия решения проводится сбор и анализ данных, которые играют ключевую роль в формировании информированного подхода. Этот этап включает как количественные, так и

качественные аспекты, которые направлены на глубокое понимание текущей ситуации и прогнозирование возможных последствий.

Количественные данные предоставляют численные измерения и фактические показатели, такие как данные о производственных объемах, затратах, доходах и других экономических показателях. Эти данные позволяют проводить точные математические расчеты и анализировать их влияние на финансовые результаты компании при различных сценариях принятия решений.

Этот этап включает в себя не только сбор данных, но и их тщательный анализ с целью выявления ключевых факторов, влияющих на проблему или цель, требующую принятия решения. Количественные данные позволяют оценить текущее состояние производственных процессов, финансовое положение компании и эффективность использования ресурсов. Например, анализ данных о производственных объемах и затратах может выявить узкие места в производственных процессах или неэффективное расходование ресурсов.

С другой стороны, качественные аспекты включают в себя анализ мнений заинтересованных сторон, оценку уровня удовлетворенности клиентов, исследование рыночных тенденций и конкурентной среды. Например, проведение опросов среди клиентов и сотрудников может помочь понять их потребности, ожидания и восприятие продуктов или услуг компании.

Совокупность количественных и качественных данных обеспечивает полное понимание текущей ситуации и помогает прогнозировать возможные последствия принимаемых решений. Это основа для формулирования стратегий и тактических действий, направленных на достижение поставленных целей и оптимизацию процессов в организации.

А качественные аспекты включают в себя анализ неколичественных данных, таких как мнения заинтересованных сторон, оценки рисков, ожидания клиентов и требования регулирующих органов [15, 78–81].

Эти данные помогают понять социальные, политические и культурные факторы, которые могут повлиять на реализацию решений и их результаты в долгосрочной перспективе. На этапе оценки альтернатив происходит сравнение различных вариантов решений с учетом заданных критериев эффективности и целей организации. Здесь используются методы анализа, моделирования и оптимизации для выбора оптимального варианта. И, наконец, после принятия решения происходит его реализация и контроль результатов. Этот этап вклю-

чает планирование действий, а также мониторинг и оценку реализации выбранной стратегии или решения.

Все эти этапы составляют процесс принятия решений, который способствует достижению целей организации, оптимизации использования ресурсов и улучшению результатов деятельности.

1.3. Модели принятия решений

Процесс принятия решения — это выбор между несколькими альтернативами, который осуществляется на основе анализа информации и оценки возможных последствий. В зависимости от сложности задачи и доступных ресурсов можно использовать различные модели принятия решений.

Рациональная модель предполагает, что лицо, принимающее решение (ЛПР), обладает всей необходимой информацией о ситуации и может объективно оценить все альтернативы. ЛПР выбирает вариант, который наилучшим образом соответствует его целям и критериям. Эта модель часто используется в бизнесе и управлении, где необходимо принимать обоснованные и рациональные решения [16].

В бизнесе и управлении рациональная модель принятия решений часто применяется для формализации процессов и обеспечения систематического подхода к принятию обоснованных решений.

Давайте рассмотрим примеры того, как рациональная модель может быть формализована в виде кода или алгоритма для автоматизации процессов принятия решений.

Предположим, у компании есть задача выбрать наилучшего поставщика для определенного товара на основе нескольких критериев: цена, качество, сроки поставки. Рациональная модель предполагает, что все эти критерии важны и должны быть объективно оценены.

Ниже приведен пример кода для выбора поставщика на основе рациональной модели.

Список доступных поставщиков с их предложениями:

```
suppliers = [
    {"name": "Supplier A", "price": 1000, "quality": 8,
"delivery_time": 7},
    {"name": "Supplier B", "price": 900, "quality": 7,
"delivery_time": 5},
    {"name": "Supplier C", "price": 1100, "quality": 9,
"delivery_time": 6}
]
```

Функция для выбора лучшего поставщика на основе цены, качества и времени поставки:

```
def choose_supplier(suppliers):
    best_supplier = None
    best_score = float('inf')
```

Инициализируем максимальным значением:

```
    for supplier in suppliers:
```

Рассчитываем общий балл на основе весовых коэффициентов для каждого критерия:

```
        score = supplier['price'] + supplier['quality']
0.5 - supplier['delivery_time']  0.2
```

Если текущий поставщик лучше предыдущего по какому-либо критерию, обновляем выбор:

```
        if score < best_score:
            best_score = score
            best_supplier = supplier

    return best_supplier

best_supplier = choose_supplier(suppliers)
print("Лучший поставщик:", best_supplier['name'])
```

В этом примере функция choose_supplier() вычисляет общий балл для каждого поставщика на основе весовых коэффициентов для цены, качества и времени поставки, после чего выбирается поставщик с наименьшим общим баллом, что соответствует лучшему выбору согласно рациональной модели.

Пример. Инвестиционное решение.

Рассмотрим случай, когда компания принимает решение о вложении средств в определенный проект на основе прогнозируемой прибыли и риска.

```python
```

Пример кода для выбора инвестиционного проекта на основе рациональной модели:

```
projects = [
    {"name": "Project A", "expected_profit": 1000000,
"risk_score": 3},
    {"name": "Project B", "expected_profit": 800000,
"risk_score": 2},
    {"name": "Project C", "expected_profit": 1200000,
"risk_score": 4}
]
```

Функция для выбора лучшего инвестиционного проекта на основе ожидаемой прибыли и риска:

```
def choose_project(projects):
    best_project = None
    best_score = float('-inf')
```

Инициализируем минимальным значением:

```
    for project in projects:
```

Рассчитываем общий балл на основе весовых коэффициентов для прибыли и риска:

```
        score = project['expected_profit'] - pro-
ject['risk_score']  50000
```

Если текущий проект лучше предыдущего по какому-либо критерию, обновляем выбор:

```
        if score > best_score:
            best_score = score
            best_project = project

    return best_project

best_project = choose_project(projects)
print("Лучший проект для инвестиций:",
best_project['name'])
```

Здесь функция choose_project() рассчитывает общий балл для каждого проекта на основе весовых коэффициентов для ожидаемой прибыли и риска. Выбирается проект с наибольшим общим баллом, что соответствует оптимальному выбору согласно рациональной модели.

Рассмотрим еще несколько примеров применения рациональной модели принятия решений в виде кода для различных бизнес-сценариев [17].

Предположим, у компании есть несколько альтернативных маркетинговых стратегий и необходимо выбрать наилучший вариант на основе ожидаемого дохода и затрат.

```python
```

Пример кода для выбора маркетинговой стратегии на основе рациональной модели:

```
strategies = [
    {"name": "Strategy A", "expected_revenue": 500000,
"expected_costs": 300000},
```

```
    {"name": "Strategy B", "expected_revenue": 600000,
"expected_costs": 350000},
    {"name": "Strategy C", "expected_revenue": 550000,
"expected_costs": 320000}
]
```

Функция для выбора лучшей маркетинговой стратегии на основе ожидаемой прибыли и затрат:

```
def choose_strategy(strategies):
    best_strategy = None
    best_profit_margin = float('-inf')
```

Инициализируем минимальным значением:

```
    for strategy in strategies:
```

Рассчитываем прибыльность стратегии как разницу между ожидаемой выручкой и затратами:

```
        profit_margin = strategy['expected_revenue'] -
strategy['expected_costs']
```

Если текущая стратегия более прибыльна, чем предыдущая, обновляем выбор:

```
        if profit_margin > best_profit_margin:
            best_profit_margin = profit_margin
            best_strategy = strategy

    return best_strategy

best_strategy = choose_strategy(strategies)
print("Лучшая маркетинговая стратегия:",
best_strategy['name'])
```

В этом примере функция choose_strategy() оценивает каждую маркетинговую стратегию на основе ожидаемой прибыли (выручка минус затраты). Выбирается стратегия с наибольшей прибыльностью, что соответствует рациональной модели принятия решений в бизнесе.

Представим, что компания рассматривает несколько вариантов развития своего продукта и хочет определить наилучший вариант на основе потенциальной рентабельности и рисков.

Пример кода для выбора стратегии развития продукта на основе рациональной модели:

```
product_strategies = [
    {"name": "Strategy X", "expected_profit": 800000,
"risk_level": 2},
```

```
    {"name": "Strategy Y", "expected_profit": 700000,
"risk_level": 3},
    {"name": "Strategy Z", "expected_profit": 900000,
"risk_level": 1}
]
```

Функция для выбора лучшей стратегии развития продукта на основе ожидаемой прибыли и уровня риска:

```
def choose_product_strategy(strategies):
    best_strategy = None
    best_score = float('-inf')
```

Инициализируем минимальным значением:

```
    for strategy in strategies:
```

Рассчитываем общий балл на основе весовых коэффициентов для ожидаемой прибыли и риска:

```
        score = strategy['expected_profit'] - strate-
gy['risk_level']  100000
```

Если текущая стратегия более выгодна, чем предыдущая, обновляем выбор:

```
        if score > best_score:
            best_score = score
            best_strategy = strategy

    return best_strategy

best_product_strategy =
choose_product_strategy(product_strategies)
print("Лучшая стратегия развития продукта:",
best_product_strategy['name'])
```

В этом примере функция choose_product_strategy() рассчитывает общий балл для каждой стратегии развития продукта на основе весовых коэффициентов для ожидаемой прибыли и уровня риска. Выбирается стратегия с наибольшим общим баллом, что соответствует рациональной модели принятия решений.

Формализуем выбор стратегии с наибольшим общим баллом на основе рациональной модели принятия решений с помощью кода. В данном примере будем считать, что у нас есть несколько стратегий развития продукта и мы выбираем наилучший вариант на основе ожидаемой прибыли и уровня риска [18].

Список стратегий развития продукта с их ожидаемой прибылью и уровнем риска:

```
product_strategies = [
    {"name": "Strategy X", "expected_profit": 800000,
"risk_level": 2},
    {"name": "Strategy Y", "expected_profit": 700000,
"risk_level": 3},
    {"name": "Strategy Z", "expected_profit": 900000,
"risk_level": 1}
]
```

Функция для выбора лучшей стратегии развития продукта на основе рациональной модели:

```
def choose_product_strategy(strategies):
    best_strategy = None
    best_score = float('-inf')
```

Инициализируем минимальным значение:

```
    for strategy in strategies:
```

Рассчитываем общий балл на основе весовых коэффициентов для ожидаемой прибыли и риска.

В данном примере у нас два критерия: ожидаемая прибыль и уровень риска.

Мы можем применить произвольные весовые коэффициенты или даже задать их интерактивно:

```
        score = strategy['expected_profit'] - strate-
gy['risk_level']  100000
```

Если текущая стратегия более выгодна, чем предыдущая, обновляем выбор:

```
        if score > best_score:
            best_score = score
            best_strategy = strategy

    return best_strategy
```

Вызываем функцию для выбора лучшей стратегии:

```
best_product_strategy =
choose_product_strategy(product_strategies)
```

Выводим результат:

```
print("Лучшая стратегия развития продукта:",
best_product_strategy['name'])
```

Разъяснения:

1) product_strategies — это список словарей, каждый из которых представляет одну стратегию развития продукта. Каждая стратегия имеет свое имя (name), ожидаемую прибыль (expected_profit) и уровень риска (risk_level);

2) choose_product_strategy() — это функция, которая принимает список стратегий (strategies) в качестве аргумента и выполняет выбор лучшей стратегии на основе рациональной модели. В данном случае мы вычисляем общий балл для каждой стратегии, используя формулу score = expected_profit — risk_level 100 000. Весовой коэффициент для уровня риска (в данном случае 100 000) выбран произвольно и может быть адаптирован к конкретным условиям;

3) best_strategy — это переменная, которая хранит лучшую стратегию после прохождения всех стратегий в цикле.

Инициализируется значением None и обновляется при каждой итерации цикла, если текущая стратегия оказывается более выгодной;

4) print(). Функция выводит на экран имя лучшей стратегии, найденной функцией choose_product_strategy().

Этот пример демонстрирует применение рациональной модели принятия решений в бизнесе через формализацию критериев (ожидаемая прибыль и уровень риска) и выбор наилучшей стратегии на основе этих критериев.

Таким образом, кодирование рациональной модели принятия решений позволяет более систематически подходить к анализу и выбору оптимальных вариантов в различных сферах бизнеса и управления, обеспечивая объективность и логическую последовательность в процессе принятия решений.

Кодирование рациональной модели принятия решений в бизнесе и управлении обеспечивает систематический подход к анализу альтернативных вариантов и выбору наилучшего решения. В рамках этой модели принятие решений осуществляется на основе объективной оценки данных и критериев, что способствует повышению эффективности и минимизации рисков [19, 82–86].

Каждое решение в рамках рациональной модели основывается на полном сборе и анализе необходимой информации. Это включает в себя оценку всех доступных альтернативных вариантов действий и их характеристик, таких как ожидаемая прибыль, затраты, риски и другие факторы, влияющие на результаты.

Вот пример простой баланс-панели в виде кода на C++, демонстрирующий основные аспекты рациональной модели принятия решений. В данном примере предполагается, что мы анализируем три альтернативных варианта инвестиций и оцениваем их по критериям прибыли, затрат и риска.

```
include <iostream>
include <vector>
include <string>
```

```
using namespace std;
```

Структура для представления альтернативы:

```
struct Alternative {
    string name;
    double expected_profit;
    double expected_costs;
    double risk_factor;
};
```

Функция для выбора лучшей альтернативы на основе рациональной модели:

```
Alternative choose_best_alternative(vector<Alternative>&
alternatives) {
    Alternative best_alternative;
    double best_score = -1;
```

Инициализируем минимальным значением.

Проходим по каждой альтернативе:

```
    for (Alternative& alternative : alternatives) {
```

Вычисляем общий балл на основе весовых коэффициентов.

В данном примере веса выбраны произвольно — прибыль важнее затрат, а риск уменьшает оценку:

```
        double score = alternative.expected_profit - al-
ternative.expected_costs - alternative.risk_factor  1000;
```

Если текущая альтернатива лучше предыдущей, обновляем выбор:

```
        if (score > best_score) {
            best_score = score;
            best_alternative = alternative;
        }
    }

    return best_alternative;
}

int main() {
```

Создаем вектор альтернатив:

```
    vector<Alternative> alternatives = {
        {"Alternative A", 100000, 50000, 2},
        {"Alternative B", 120000, 60000, 3},
        {"Alternative C", 90000, 40000, 1}
```

```
    };

    Выбираем лучшую альтернативу:

    Alternative best_alternative =
choose_best_alternative(alternatives);

    Выводим результат:

    cout << "Лучшая альтернатива: " <<
best_alternative.name << endl;

    return 0;
}
```

1. Структура Alternative. Определяет структуру данных для хранения альтернативы, включая имя (name), ожидаемую прибыль (expected_profit), ожидаемые затраты (expected_costs) и коэффициент риска (risk_factor).

2. Функция choose_best_alternative. Это функция, которая принимает вектор альтернатив и выбирает лучшую альтернативу на основе рациональной модели. В данном примере вычисляется общий балл для каждой альтернативы на основе весовых коэффициентов: прибыль имеет положительный вес, затраты — отрицательный, а риск уменьшает общую оценку.

3. Главная функция main. В ней создается вектор alternatives, представляющий различные альтернативы инвестиций. Затем вызывается функция choose_best_alternative, чтобы определить лучшую альтернативу, и результат выводится на экран.

Этот пример демонстрирует использование рациональной модели принятия решений для выбора наилучшей альтернативы на основе объективного анализа прибыли, затрат и риска.

Реальная реализация модели может варьироваться в зависимости от конкретных бизнес-требований и критериев оценки.

Применение рациональной модели требует четкой формализации критериев оценки и их взвешивания в соответствии с приоритетами организации или индивидуальных целей. Это помогает избежать субъективных предпочтений или случайных влияний на процесс принятия решений.

В процессе принятия решений на основе рациональной модели важно учитывать не только текущие условия и данные, но и потенциальные последствия выбранного решения в долгосрочной перспективе.

Это способствует формированию стратегического подхода к управлению и развитию бизнеса, обеспечивая его устойчивость и конкурентоспособность.

В заключение, применение рациональной модели принятия решений в коде помогает организациям и руководителям достигать более обоснованных и оптимальных результатов, минимизируя риски и увеличивая вероятность успешного достижения поставленных целей.

Однако в реальной жизни не всегда возможно собрать всю необходимую информацию или точно оценить последствия каждого варианта. Поэтому были разработаны другие модели, которые учитывают ограничения и неопределенность, присущие процессу принятия решений.

Модель ограниченной рациональности предполагает, что ЛПР не имеет полной информации о ситуации и не может точно оценить все последствия.

Также предполагается, что ЛПР оперирует с ограниченным объемом информации и ограниченным временем для принятия решений. В отличие от рациональной модели, где ЛПР может объективно оценить все альтернативы, модель ограниченной рациональности предполагает использование эвристик и упрощенных стратегий для принятия решений.

Давайте рассмотрим пример модели ограниченной рациональности на языке Python, где мы будем выбирать оптимальный вариант на основе ограниченной информации и применения эвристик.

```python
```

Пример кода для модели ограниченной рациональности.

Импортируем библиотеку для работы со случайными числами:

```
import random
```

Список альтернативных вариантов с их оценками (например, доходность и риск):

```
alternatives = [
    {"name": "Alternative X", "expected_profit": 80000,
"risk_level": "high"},
    {"name": "Alternative Y", "expected_profit": 70000,
"risk_level": "medium"},
    {"name": "Alternative Z", "expected_profit": 90000,
"risk_level": "low"}
]
```

Функция для выбора альтернативы на основе модели ограниченной рациональности:

```
def choose_alternative(alternatives):
```

Выбираем случайным образом одну из альтернатив, эмулируя ограниченную информацию:

```
    chosen_alternative = random.choice(alternatives)
    return chosen_alternative
```

Выбираем альтернативу на основе модели ограниченной рациональности:

```
selected_alternative = choose_alternative(alternatives)
```

Выводим результат выбора:

```
print("Выбранная альтернатива:",
selected_alternative['name'])
print("Ожидаемая прибыль:", select-
ed_alternative['expected_profit'])
print("Уровень риска:", select-
ed_alternative['risk_level'])
```

Описание кода:

1) alternatives — это список словарей, представляющих различные альтернативные варианты. Каждая альтернатива имеет имя (name), ожидаемую прибыль (expected_profit) и уровень риска (risk_level);

2) choose_alternative — это функция, которая выбирает одну из альтернатив на основе модели ограниченной рациональности. В данном примере выбор осуществляется случайным образом среди доступных альтернатив, что эмулирует ограниченность информации и использование эвристик для принятия решений;

3) selected_alternative — это переменная, которая хранит выбранную функцией choose_alternative альтернативу;

4) вывод результатов. Выводится имя выбранной альтернативы, ее ожидаемая прибыль и уровень риска.

Этот пример демонстрирует, как модель ограниченной рациональности может быть реализована в коде, где выбор альтернативы происходит на основе эвристических методов или ограниченного объема информации, доступного принимающему решения лицу.

Такой подход позволяет учесть ограничения реального мира и использовать простые стратегии для быстрого принятия решений.

Вместо этого он использует упрощенные критерии и эвристики для выбора варианта, который кажется наиболее подходящим. Эта модель более реалистична, чем рациональная модель, но она также может привести к ошибкам и неэффективным решениям.

Еще одна модель — политическая, которая учитывает интересы различных групп и лиц, участвующих в процессе принятия решений. В этой модели должно учитывать не только свои собственные цели, но и цели других участников.

Политическая модель может быть сложной и непредсказуемой, но она позволяет учесть разнообразие мнений и интересов.

Выбор модели зависит от конкретной ситуации и целей ЛПР. Рациональная модель подходит для ситуаций, когда есть достаточно информации и времени для анализа. Модель ограниченной рациональности подходит для ситуаций с ограниченными ресурсами и временем. Политическая модель подходит для сложных ситуаций с участием многих заинтересованных сторон.

Давайте рассмотрим пример модели принятия решений, которая подходит для сложных ситуаций с участием многих заинтересованных сторон, таких как политические процессы. В таких случаях часто применяются модели, учитывающие разнообразные точки зрения и интересы различных групп. Вот как можно смоделировать подобную ситуацию на языке Python.

Пример кода для политической модели принятия решений.

Список участников или заинтересованных сторон:

```
stakeholders = [
    {"name": "Government", "influence": 0.7},
    {"name": "Business Lobby", "influence": 0.5},
    {"name": "Environmental NGO", "influence": 0.4},
    {"name": "Community Representatives", "influence":
0.6}
]
```

Функция для принятия решения на основе политической модели:

```
def make_decision(stakeholders):
```

Имитация сложного процесса принятия решений, учитывающего влияние различных сторон:

```
    max_influence = -1
    decision_maker = None

    for stakeholder in stakeholders:
        if stakeholder['influence'] > max_influence:
            max_influence = stakeholder['influence']
            decision_maker = stakeholder['name']

    return decision_maker
```

Принимаем решение на основе политической модели:

```
decision_maker = make_decision(stakeholders)
```

Выводим результат принятия решения:

```
print("Решение было принято:", decision_maker)
```

Описание кода:

1) stakeholders — это список словарей, представляющих различные заинтересованные стороны или участников процесса принятия решений. Каждый участник имеет имя (name) и уровень влияния (influence), который может отражать их политическую мощь, общественную поддержку или экономическое влияние;

2) make_decision — это функция, которая принимает список участников (stakeholders) и выбирает того, чье влияние наиболее сильное. В данном примере принятие решения основано на максимальном уровне влияния среди всех участников, что может отражать политическую модель, где решение принимается с учетом интересов различных групп;

3) decision_maker — это переменная, которая хранит результат функции make_decision; это тот участник или сторона, чье влияние было наибольшим и кто, возможно, принял решение.

Этот пример демонстрирует, как модель принятия решений может учитывать политические аспекты и интересы различных сторон в сложных ситуациях.

Такой подход помогает адаптировать процесс принятия решений к особенностям политических процессов, где важно учитывать разнообразие мнений и влияние различных заинтересованных сторон.

Важно помнить, что ни одна модель не является идеальной. Каждая из них имеет свои преимущества и недостатки, и выбор модели должен основываться на анализе конкретной ситуации.

Еще одной важной моделью принятия решений является модель принятия решений в условиях неопределенности или ограниченной рациональности. В таких ситуациях принимающий решение сталкивается с неопределенностью или недостатком информации, что затрудняет полное анализирование всех альтернатив и прогнозирование результатов (рис. 3).

Также стоит упомянуть модель группового принятия решений, где процесс принятия решений включает в себя несколько участников или стейкхолдеров.

В групповом контексте часто используются методы консенсуса, голосования или дискуссии для достижения соглашения по поводу предпочтительного решения.

Важным аспектом любой модели принятия решений является учет контекста и специфики ситуации, так как каждая из них имеет свои преимущества и ограничения. Понимание различий между моделями помогает выбрать наиболее подходящий подход для конкретной задачи или проблемы, что способствует более эффективному принятию решений и достижению желаемых результатов.

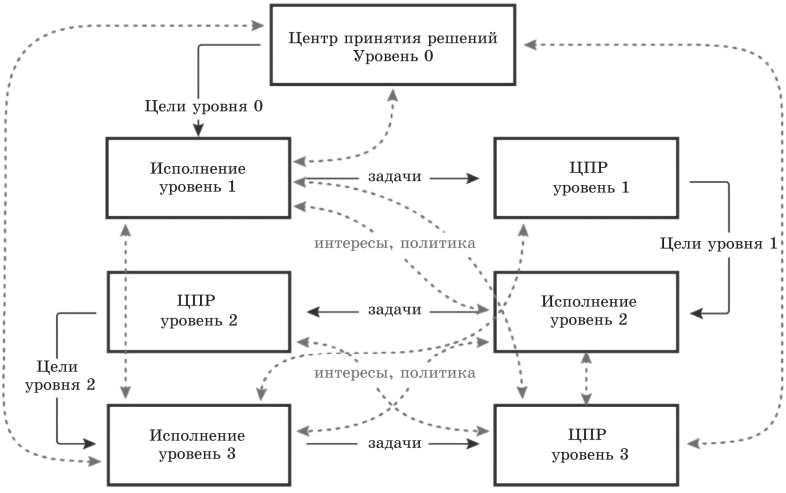

Рис. 3

Вместо того чтобы стремиться к оптимальному решению, принимающий решение выбирает первый приемлемый вариант или тот, который обеспечивает достаточное удовлетворение потребностей и целей

Однако независимо от выбранной модели важно помнить, что принятие решений — это процесс, который требует анализа, обдумывания и оценки различных вариантов. Не существует универсального решения, и иногда приходится принимать решения на основе неполной информации или с ограниченными ресурсами.

В итоге эффективность моделей принятия решений зависит от компетентности тех, кто их применяет, от адекватности выбранной стратегии и от способности адаптироваться к изменяющимся обстоятельствам.

Важно использовать модели как инструменты, которые помогут вам принимать обоснованные и эффективные решения, но не забывать о своем интуитивном понимании ситуации и опыте.

1.4. Инструменты и методы поддержки принятия решений

Инструменты и методы поддержки принятия решений играют важную роль в современных организациях и управлении, обеспечивая эффективный анализ данных и принятие обоснованных решений.

Среди них выделяются такие инструменты, как аналитические системы и программные платформы, которые позволяют собирать, обрабатывать и анализировать большие объемы информации.

Это включает в себя использование бизнес-интеллекта (Business Intelligence) для выявления ключевых трендов и паттернов данных, что помогает принимающим решения лицам выявлять важные факторы и прогнозировать результаты. Также важными являются качественные и количественные методы анализа, включая статистические техники, моделирование и симуляцию, которые позволяют оценивать вероятные последствия различных решений и проводить чувствительный анализ. Эти методы позволяют учитывать различные сценарии развития событий и оценивать риски, что способствует более глубокому и обоснованному принятию решений в условиях неопределенности и изменчивости внешней среды.

Инновационные технологии, такие как искусственный интеллект и машинное обучение, также находят свое применение в области поддержки принятия решений, предоставляя возможности для автоматизации процессов анализа данных и предсказания результатов на основе больших данных.

Эти инструменты не только повышают скорость принятия решений, но и улучшают их качество за счет более точных прогнозов и оптимизации стратегий.

Интеграция современных методов и инструментов поддержки принятия решений позволяет организациям эффективно управлять своей деятельностью, минимизировать риски и достигать поставленных целей в динамично изменяющейся бизнес-среде.

Дополнительно к технологиям и методам анализа данных ключевыми инструментами поддержки принятия решений являются системы управления знаниями и экспертные системы. Системы управления знаниями позволяют организациям эффективно хранить, организовывать и распространять знания внутри организации. Это включает документацию, экспертные мнения, лучшие практики и опыт прошлых решений, что способствует улучшению качества принимаемых решений и их последующей реализации.

Экспертные системы, в свою очередь, используют знания и опыт экспертов в виде логических правил или моделей для автоматизации процесса принятия решений. Они могут анализировать входные данные и предлагать рекомендации или принимать решения на основе заданных критериев и правил, что особенно полезно в ситуациях, требующих экспертного знания, или в случаях, когда необходимо обработать большое количество информации.

Помимо технических средств важным аспектом поддержки принятия решений является развитие процессов и методологий управления, таких как стратегическое планирование, управление проектами и управление рисками. Эти подходы помогают структурировать процесс принятия решений, устанавливать приоритеты и контролировать выполнение поставленных задач.

Наконец, современные инструменты поддержки принятия решений активно интегрируют в себя аспекты цифровизации и автоматизации процессов, что способствует повышению эффективности и улучшению результатов.

Их использование позволяет компаниям быть более гибкими и адаптивными в быстро меняющейся бизнес-среде, обеспечивая долгосрочное развитие и конкурентоспособность.

С математической точки зрения инструменты и методы поддержки принятия решений основаны на строгом анализе данных и применении различных математических моделей. Одной из ключевых областей является статистика, которая используется для оценки значимости данных, проверки гипотез и выявления закономерностей на основе собранных данных. Теория вероятностей позволяет оценивать вероятные результаты различных сценариев и учитывать неопределенность в процессе принятия решений.

Оптимизационные методы, такие как линейное программирование и динамическое программирование, используются для поиска оптимальных решений с учетом ограничений и целевых функций. Эти методы помогают организациям эффективно распределять ресурсы и достигать поставленных целей. Моделирование и симуляция играют важную роль в анализе различных сценариев и оценке их влияния на бизнес-процессы, что помогает принимать информированные решения на основе прогнозов и статистических данных.

Экспертные системы и искусственный интеллект используются для автоматизации процессов принятия решений на основе знаний и опыта экспертов.

Они способствуют улучшению точности и скорости принятия решений, используя методы машинного обучения, нейронные сети и алгоритмы обработки данных. Все эти математические подходы объединяются для создания эффективных инструментов поддержки принятия решений, способствующих развитию более точных и адаптивных стратегий управления в современных организациях.

ГЛАВА 2
ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

2.1. Определение и классификация решений

Определение и классификация решений играют важную роль в контексте управления и принятия стратегических решений в организациях. Решение можно определить как выбор между альтернативами, направленный на достижение определенной цели или решения проблемы. Это процесс, который может включать анализ данных, оценку последствий и выбор оптимального варианта действий на основе целей и критериев принятия решений.

Решения можно классифицировать по различным критериям, включая степень структурированности и уровень неопределенности. Структурированные решения характеризуются четко определенными правилами и процедурами, что позволяет автоматизировать процесс принятия решений. Например, рутинные операции и процедуры могут быть формализованы с использованием стандартных алгоритмов.

Неструктурированные решения, напротив, возникают в условиях высокой степени неопределенности и требуют творческого подхода и экспертных знаний. Такие решения часто связаны с управлением неожиданными ситуациями или разработкой инновационных стратегий. Важно уметь адаптироваться к переменам и быстро анализировать новую информацию для принятия неструктурированных решений на основе интуиции и экспертизы.

Продолжая разговор о классификации решений, следует отметить также их классификацию по уровню влияния и времени, необходимому для их принятия. Стратегические решения, например, имеют долгосрочный характер и касаются ключевых аспектов развития организации, таких как направление бизнеса, внедрение новых технологий или расширение рынков.

Тактические решения, в свою очередь, охватывают более короткие временные рамки и обычно связаны с текущими операционными задачами и ресурсами.

Они направлены на достижение более конкретных целей в более ограниченном пространстве времени.

Оперативные решения, наконец, принимаются на ежедневной основе для решения текущих проблем и выполнения рутинных задач. Они обычно требуют меньше времени на принятие и исполнение по сравнению с более стратегическими и тактическими решениями.

Классификация решений по временным рамкам и уровню их воздействия позволяет организациям эффективно управлять различными видами деятельности, учитывая их важность и влияние на достижение стратегических целей и операционной эффективности.

Рассмотрим простой пример классификации решений с использованием кода на языке Python. В этом примере мы создадим класс Decision, который будет иметь атрибуты для описания различных типов решений: стратегических, тактических и оперативных.

```
class Decision:
    def __init__(self, decision_type, description):
        self.decision_type = decision_type
        self.description = description

    def classify(self):
        if self.decision_type == 'strategic':
            return 'This decision is a strategic one.'
        elif self.decision_type == 'tactical':
            return 'This decision is a tactical one.'
        elif self.decision_type == 'operational':
            return 'This decision is an operational one.'
        else:
            return 'Unknown decision type.'
```

Пример использования класса Decision:

```
decision1 = Decision('strategic', 'Develop a new market
strategy for the next 5 years.')
decision2 = Decision('tactical', 'Implement a new soft-
ware system for improving productivity.')
decision3 = Decision('operational', 'Schedule weekly team
meetings to review progress.')
 Вывод результатов классификации решений
print(decision1.classify())
print(decision2.classify())
print(decision3.classify())
```

Описание кода:

1) класс Decision определяет тип решения (decision_type) и его описание (description). У него есть метод classify, который на основе значения атрибута decision_type возвращает строку, указывающую на тип решения (стратегическое, тактическое или оперативное);

2) примеры решений: мы создаем три экземпляра класса Decision с разными типами решений:

– decision1 — стратегическое решение: разработка новой стратегии рынка на следующие 5 лет;

– decision2 — тактическое решение: внедрение новой системы программного обеспечения для повышения производительности;

– decision3 — оперативное решение: планирование еженедельных совещаний команды для обзора прогресса;

3) вывод результатов. Для каждого созданного экземпляра выводится результат работы метода classify, который указывает на тип принятого решения.

Этот пример демонстрирует, как можно использовать объектно-ориентированный подход для классификации различных типов решений на основе их характеристик. Код позволяет ясно различать стратегические, тактические и оперативные решения, что полезно при организации и управлении процессами принятия решений в организациях.

Классификация решений играет важную роль в организационной стратегии и управлении, помогая ориентироваться в их разнообразии и значимости для бизнеса. Стратегические решения направлены на долгосрочное развитие организации и часто связаны с определением ключевых направлений и целей компании. Тактические решения затрагивают более операционные аспекты деятельности, ориентируясь на достижение конкретных задач в рамках стратегических целей. Оперативные решения, в свою очередь, фокусируются на повседневных операциях и требуют быстрого реагирования для обеспечения эффективности текущих процессов.

Применение классификации решений в практике помогает лучше понимать и оценивать их влияние на бизнес и оптимизировать процесс принятия решений в зависимости от контекста и важности. Это позволяет руководителям и менеджерам организаций эффективнее управлять ресурсами, сосредотачиваясь на стратегических приоритетах и оперативных потребностях, что в конечном итоге способствует достижению долгосрочных успехов и устойчивого развития компании.

2.2. Процесс принятия решений

Процесс принятия решений представляет собой систематический подход к выбору оптимального варианта действий из нескольких альтернативных решений.

Он начинается с определения проблемы или цели, которая требует принятия решения, и сбора необходимой информации для анализа ситуации. Важным этапом является выявление всех возможных альтернатив и их характеристик, таких как потенциальные выгоды, затраты, риски и последствия.

Далее следует анализ и оценка каждой альтернативы на основе установленных критериев принятия решений. Этот этап включает в себя применение различных методов и инструментов, таких как математические модели, статистические техники и экспертные оценки, для выявления наилучшего варианта действий.

После проведения анализа необходимо выбрать предпочтительную альтернативу и разработать план ее реализации. Важным аспектом этого этапа является оценка возможных последствий выбранного решения и подготовка к управлению ими в процессе его выполнения.

Процесс принятия решений является ключевым элементом управления в любой организации, поскольку от качества принимаемых решений зависит успешность достижения стратегических целей и операционная эффективность.

Этот процесс не просто административная формальность, а стратегически важная функция, которая позволяет организации адаптироваться к изменениям внешней среды, выявлять и использовать возможности для роста и развития.

Качество принимаемых решений определяется не только правильностью выбора альтернативы, но и глубиной анализа, обоснованностью их оснований и оценкой потенциальных последствий. Это важно как для крупных стратегических решений, так и для ежедневных операционных задач, так как они непосредственно влияют на эффективность работы всей организации.

Процесс принятия решений также способствует развитию управленческой культуры, стимулирует профессиональное развитие сотрудников и формирует ответственное поведение внутри организации. Это подход необходим для достижения долгосрочной устойчивости и конкурентоспособности на рынке, учитывая быстро меняющиеся условия и требования бизнес-среды.

Таким образом, эффективный процесс принятия решений является неотъемлемой частью успешного управления и ключевым фактором в достижении высоких результатов и устойчивого развития организации в долгосрочной перспективе.

Этот процесс не только структурирует подход к принятию решений, но и способствует минимизации рисков и оптимизации ресурсов, что особенно важно в условиях динамично меняющейся бизнес-среды.

Рассмотрим конкретный кейс внедрения новой линейки продуктов в технологической компании. Это стратегическое решение требует тщательного анализа и оценки, начиная с определения целей и проблем, которые оно должно решить. Важно собрать достаточно

информации о рыночной среде, потребительских требованиях и конкурентной динамике, чтобы выбрать наиболее перспективное направление развития.

Далее следует выявление и анализ различных альтернатив, таких как разработка смартфонов, носимых устройств или умных систем для дома. Каждая альтернатива подвергается глубокому анализу с учетом финансовых затрат, потенциальной прибыли, технологической сложности и рисков.

Выбор наилучшей альтернативы осуществляется на основе совокупности данных и оценок, что позволяет компании принимать обоснованные и стратегические решения. Далее разрабатывается детальный план внедрения, который включает в себя этапы разработки продукта, маркетинговую стратегию и организацию производственных процессов.

Реализация новой линейки продуктов требует координации усилий различных функциональных подразделений компании и построения эффективных коммуникационных каналов. Оценка эффективности проекта важна для непрерывного улучшения стратегий компании и адаптации к изменениям внешней среды и потребительских предпочтений.

Эффективный процесс принятия решений включает в себя не только анализ внутренних факторов и данных, но и учет внешних влияний, таких как экономические условия, конкурентная среда и изменения в законодательстве. Это помогает организациям адаптироваться к переменам и принимать обоснованные решения, соответствующие текущим реалиям рынка и бизнес-окружения.

Кроме того, процесс принятия решений часто включает в себя коллективное обсуждение и совместную работу различных стейкхолдеров, что способствует улучшению качества принимаемых решений за счет разнообразия взглядов и опыта. Важно обеспечить прозрачность и открытость в ходе этого процесса, чтобы все участники могли внести свой вклад и поддержать окончательное решение.

2.3. Целочисленное программирование

Целочисленное программирование (ЦП) представляет собой математический метод оптимизации, который используется для решения задач, где переменные должны принимать целочисленные значения. Этот метод находит широкое применение в различных областях, включая производственное планирование, логистику, финансовый анализ и многие другие.

Основная задача целочисленного программирования заключается в нахождении такого набора значений переменных, которые минимизируют или максимизируют целевую функцию с учетом ограничений при условии, что переменные принимают только целочисленные значения. Это отличает целочисленное программирование от линейного программирования, где переменные могут принимать любые действительные значения.

Важным аспектом целочисленного программирования является выбор между использованием целочисленных переменных (integer programming) и бинарных переменных (binary programming), где переменные могут быть только 0 или 1. Это дает возможность моделировать дискретные решения, такие как выбор местоположения склада, расписание работы или выбор инвестиционных портфелей.

Применение целочисленного программирования требует разработки математических моделей, которые точно отражают особенности задачи и ее ограничения. Решение задачи ЦП может быть осуществлено различными методами, включая ветвление и границы, динамическое программирование и методы грубой силы, в зависимости от сложности задачи и доступных вычислительных ресурсов.

Одним из классических примеров задач целочисленного программирования является задача о рюкзаке, где требуется выбрать оптимальный набор предметов с учетом их веса и стоимости так, чтобы суммарный вес не превышал заданного лимита рюкзака. В этой задаче каждый предмет либо включается в рюкзак (значение переменной равно 1), либо нет (значение переменной равно 0), что отражает дискретность выбора предметов.

Еще одной интересной задачей является задача о покрытии множества, которая используется для нахождения минимального подмножества наборов, необходимых для покрытия всех элементов в универсальном множестве. Эта задача часто возникает в контексте оптимизации маршрутов, планирования проектов и распределения ресурсов.

В этой задаче важно, чтобы выбранные наборы были целочисленными, поскольку каждый набор либо включается в решение, либо нет.

В области телекоммуникаций можно рассмотреть задачу о размещении ретрансляторов, где необходимо оптимально расположить ограниченное число ретрансляторов для обеспечения максимального покрытия сигнала в заданной области. Каждая возможная позиция ретранслятора представлена целочисленной переменной, которая указывает, установлен ли ретранслятор в этой точке или нет, что позво-

ляет эффективно использовать методы целочисленного программирования для решения данной задачи.

Для решения задачи о рюкзаке необходимо выбрать такие предметы, которые максимизируют общую стоимость, не превышая заданный весовой лимит. Используя методы целочисленного программирования, переменные задаются в виде бинарных значений, где 1 указывает на включение предмета в рюкзак, а 0 — на его исключение.

Оптимизация производится с использованием подходов линейного программирования и ветвей и границ (табл. 1).

Таблица 1

Данные задачи о рюкзаке

Item	Weight	Value
A	2	3
B	3	4
C	4	5

Решение задачи о назначениях включает в себя минимизацию общих затрат на выполнение задач работниками. Переменные принимают бинарные значения, где 1 указывает на назначение работника на задачу, а 0 — на его неназначение. Оптимизация проводится через линейное программирование с ограничениями, чтобы каждый работник выполнял только одну задачу (табл. 2).

Таблица 2

Данные задачи о назначениях

Worker	Task1_Cost	Task2_Cost	Task3_Cost
W1	9	6	3
W2	2	4	5
W3	7	3	2

Для задачи о покрытии множества необходимо выбрать минимальное количество наборов, чтобы покрыть все элементы универсального множества. Переменные также бинарные, указывающие на включение или исключение набора в решение. Оптимизация осуществляется через целочисленное линейное программирование с ограничениями на покрытие всех элементов (табл. 3).

Таблица 3

Данные задачи о покрытии множества

Set	Elements	Cost
S1	{'1', '2'}	5

Продолжение табл. 3

Set	Elements	Cost
S2	{'3', '2'}	10
S3	{'4', '3'}	3

В задаче о раскрое материала требуется минимизировать отходы при разрезании большого листа на более мелкие куски. Переменные целочисленные, указывающие количество и размеры кусков. Оптимизация проводится через целочисленное программирование, учитывающее ограничения на размеры и количество необходимых кусков (табл. 4).

Таблица 4

Данные задачи о раскрое материала

Material_Length	Required_Pieces	Piece_Length
10	3	5
15	2	7
20	1	8

Для решения задачи о маршрутизации транспорта нужно определить оптимальные маршруты для транспортных средств с целью минимизации общего расстояния или времени доставки. Переменные указывают на использование определенного маршрута. Оптимизация проводится с учетом ограничений по грузоподъемности и другим ресурсам (табл. 5).

Таблица 5

Данные задачи о маршрутизации транспорта

Route	Distance	Time
R1	10	30
R2	15	45
R3	20	50

В задаче о размещении ретрансляторов необходимо оптимально расположить ретрансляторы для максимального покрытия сигнала.

Переменные бинарные, указывающие на установку ретранслятора в определенной точке. Оптимизация проводится через целочисленное программирование, учитывая ограничения на количество и стоимость ретрансляторов (табл. 6).

Таблица 6

Данные задачи о размещении ретрансляторов

Location	Signal_Coverage	Cost
L1	100	300

Продолжение табл. 6

Location	Signal_Coverage	Cost
L2	200	500
L3	150	400

Для построения дерева решений задачи о рюкзаке необходимо учитывать каждый предмет и принимать решение о его включении или исключении из рюкзака. Начинаем с корня дерева, где рассматриваем первый предмет. Если включаем его, переходим к следующему узлу, где учитываем вес и стоимость, затем принимаем решение для следующего предмета, и так далее. Листья дерева представляют собой возможные комбинации предметов, удовлетворяющие ограничениям по весу и максимизирующие общую стоимость.

В задаче о назначениях дерево решений начинается с корневого узла, где рассматривается первый работник и его возможные назначения на задачи. Каждый узел дерева представляет выбор назначения работника на одну из задач. Далее дерево разветвляется, учитывая следующего работника и оставшиеся задачи, что ведет к листьям, где все задачи распределены между работниками, что обеспечивает минимальные затраты.

Для задачи о покрытии множества дерево решений строится начиная с корня, где рассматривается первый набор.

Включение или исключение этого набора ведет к следующим узлам, где учитывается следующий набор и его элементы. Каждый путь в дереве представляет комбинацию выбранных наборов, а листья содержат минимальные наборы, которые покрывают все элементы множества.

В задаче о раскрое материала дерево решений строится, начиная с корневого узла, где рассматривается первый лист материала.

Каждый узел дерева представляет возможный вариант раскроя листа на куски. Переход к следующему узлу происходит после определения первого разреза, затем учитывается оставшийся материал и выполняются следующие разрезы, что ведет к минимальным отходам.

Для задачи о маршрутизации транспорта дерево решений начинается с выбора первого маршрута для первого транспортного средства. Каждый узел дерева представляет маршрут и его использование. Переход к следующему узлу учитывает оставшиеся маршруты и транспортные средства, оптимизируя общее расстояние или время доставки. Листья дерева представляют возможные маршруты для всего флота, обеспечивающие минимальные затраты.

В задаче о размещении ретрансляторов дерево решений строится начиная с выбора первой возможной локации для ретранслятора.

Каждый узел дерева представляет решение об установке или неустановке ретранслятора в данной точке. Переход к следующему узлу учитывает следующую возможную локацию и ее покрытие сигнала. Листья дерева показывают комбинации локаций ретрансляторов, обеспечивающие максимальное покрытие при минимальных затратах.

ЦП обладает рядом недостатков, которые могут затруднять его применение в различных задачах. Один из главных недостатков заключается в высокой вычислительной сложности. Задачи ЦП часто являются NP-трудными, что означает, что время решения может увеличиваться экспоненциально с ростом размера задачи. Это делает невозможным решение крупных задач за разумное время даже при использовании мощных компьютеров. В связи с этим на практике часто приходится использовать эвристические методы, которые дают приближенные решения, но не гарантируют нахождение оптимального решения.

Еще одной проблемой целочисленного программирования является необходимость точной формулировки задачи. Модель должна быть четко определена и все ограничения должны быть выражены в виде целочисленных переменных и линейных или нелинейных уравнений и неравенств.

Это может быть достаточно сложной задачей, особенно в случае реальных задач, которые часто содержат неопределенности и нечеткие параметры.

Ошибки в моделировании могут привести к неточным или неэффективным решениям, что снижает практическую ценность метода.

Кроме того, целочисленное программирование может быть ограничено в гибкости. Некоторые задачи, особенно те, которые включают непрерывные переменные или требуют гибкого учета различных факторов, труднее формализовать в рамках ЦП.

Например, задачи, связанные с оптимизацией процессов, где параметры могут принимать любые значения в определенных пределах, лучше решаются методами линейного или нелинейного программирования. В таких случаях использование алгоритма может привести к искусственному усложнению модели и ухудшению результатов.

Еще одним важным аспектом является чувствительность к начальному приближению и параметрам. Поскольку ЦП часто использует методы ветвей и границ или разветвляющегося поиска, начальные условия и выбор ветвей могут существенно повлиять на время и качество решения. В реальных условиях это может привести к необходимости проведения множества пробных запусков и

настройки параметров, что увеличивает трудоемкость и временные затраты на решение задачи.

В целом, несмотря на свою мощь и возможность решения сложных дискретных задач, целочисленное программирование имеет множество недостатков, которые ограничивают его применение в практических задачах. Высокая вычислительная сложность, требовательность к точной формулировке задачи, ограниченная гибкость и чувствительность к начальному приближению делают этот метод менее предпочтительным для задач, где возможны альтернативные подходы.

2.4. Динамическое программирование

Динамическое программирование (ДП) представляет собой мощный метод решения сложных задач оптимизации путем разбиения их на более простые подзадачи. Основная идея заключается в том, чтобы решать каждую подзадачу только один раз и сохранять ее решение, чтобы избежать повторных вычислений. Этот метод особенно полезен для задач с перекрывающимися подзадачами и оптимальной подструктурой, где решение задачи можно построить из решений ее подзадач.

Один из ключевых аспектов динамического программирования — это использование таблиц для хранения промежуточных результатов. Такие таблицы позволяют значительно ускорить вычисления, так как результаты уже решенных подзадач можно использовать многократно без необходимости их пересчета. Это особенно важно в задачах с большими размерами, где количество возможных подзадач может быть очень велико.

Использование таблиц или мемоизации (запоминания) помогает избежать экспоненциального роста времени выполнения алгоритма.

Динамическое программирование применяется во множестве областей, включая комбинаторные задачи, задачи на графах, задачи оптимального управления и многие другие. Например, классическими примерами использования ДП являются задачи о рюкзаке, нахождение наибольшей общей подпоследовательности и алгоритмы поиска кратчайшего пути в графах. В этих задачах динамическое программирование позволяет эффективно находить оптимальные решения, которые иначе были бы крайне трудоемкими или невозможными для вычисления с помощью методов полного перебора.

Однако, несмотря на свои преимущества, динамическое программирование имеет и некоторые недостатки. Один из них — это

высокая потребность в памяти. Так как необходимо сохранять результаты всех подзадач, для больших задач может потребоваться значительное количество оперативной памяти. Это может стать серьезной проблемой в случае, когда ресурсы ограничены. Кроме того, разработка алгоритмов на основе динамического программирования требует глубокого понимания структуры задачи и умения правильно декомпозировать ее на подзадачи, что иногда бывает нетривиально.

Конечно! Рассмотрим пример задачи о рюкзаке (Knapsack Problem). В этой задаче у нас есть набор предметов, каждый из которых имеет вес и стоимость, и рюкзак, который может выдержать ограниченный вес. Необходимо выбрать предметы так, чтобы максимизировать общую стоимость, не превышая допустимый вес рюкзака. Решим эту задачу с помощью динамического программирования.

Предположим, у нас есть следующие предметы.

Предмет	Вес	Стоимость
1	2	3
2	3	4
3	4	5
4	5	8

Максимальный допустимый вес рюкзака равен 5:

```
def knapsack(weights, values, max_weight):
    n = len(weights)
```

Создаем таблицу для хранения промежуточных результатов:

```
    dp = [[0 for _ in range(max_weight + 1)] for _ in
range(n + 1)]
```

Заполняем таблицу:

```
    for i in range(1, n + 1):
        for w in range(1, max_weight + 1):
            if weights[i-1] <= w:
                dp[i][w] = max(dp[i-1][w], dp[i-1][w-
weights[i-1]] + values[i-1])
            else:
                dp[i][w] = dp[i-1][w]

    return dp[n][max_weight]
```

Пример использования функции:

```
weights = [2, 3, 4, 5]
values = [3, 4, 5, 8]
max_weight = 5
```

```
max_value = knapsack(weights, values, max_weight)
print("Максимальная стоимость, которую можно получить:",
max_value)
```

В этом примере функция knapsack решает задачу о рюкзаке с использованием динамического программирования. Она создает двумерный массив *dp*, где *dp*[*i*][*w*] хранит максимальную стоимость, которую можно получить, используя первые *i* предметов с общим весом не более *w*.

Проходя по всем предметам и возможным весам рюкзака, функция обновляет массив *dp* на основе двух возможных вариантов: либо мы не берем текущий предмет (и оставляем максимальную стоимость такой же, как и без этого предмета), либо берем текущий предмет (и добавляем его стоимость к максимальной стоимости, которую можно получить для оставшегося веса).

Вывод программы будет следующим:

```
Максимальная стоимость, которую можно получить: 7
```

Это означает, что максимальная стоимость, которую можно получить, не превышая вес рюкзака в 5 единиц, равна 7.

Давайте усложним задачу о рюкзаке, добавив дополнительные условия. Предположим, что у нас есть ограничение на количество каждого предмета, которое можно использовать, и необходимо учитывать этот фактор при выборе предметов. Например, каждый предмет может быть выбран не более одного раза.

Вот пример решения этой задачи с использованием динамического программирования, учитывающего ограничение на количество предметов.

```python
def knapsack_with_limitations(weights, values,
max_weight, quantities):
    n = len(weights)
```

Создаем таблицу для хранения промежуточных результатов:

```
    dp = [[0 for _ in range(max_weight + 1)] for _ in
range(n + 1)]
```

Заполняем таблицу:

```
    for i in range(1, n + 1):
        for w in range(max_weight + 1):
            dp[i][w] = dp[i-1][w]
```

Инициализируем значением без текущего предмета:

```
            for k in range(1, quantities[i-1] + 1):
                if k  weights[i-1] <= w:
                    dp[i][w] = max(dp[i][w], dp[i-1][w -
k  weights[i-1]] + k  values[i-1])
                else:
                    break

    return dp[n][max_weight]
```

Пример использования функции:

```
weights = [2, 3, 4, 5]
values = [3, 4, 5, 8]
quantities = [1, 1, 1, 1]
```

Ограничение на количество каждого предмета:

```
max_weight = 5

max_value = knapsack_with_limitations(weights, values,
max_weight, quantities)
print("Максимальная стоимость, которую можно получить:",
max_value)
```

В этой версии задачи добавлен параметр quantities, который указывает максимальное количество каждого предмета, которое можно использовать. Мы модифицировали алгоритм динамического программирования, чтобы учесть этот параметр.

При заполнении таблицы *dp* для каждого предмета и возможного веса мы также рассматриваем все возможные количества предмета, которые можно добавить в рюкзак, учитывая его ограничение по количеству и весу. Для каждого количества *k* проверяется, можно ли добавить *k* экземпляров текущего предмета в рюкзак, не превышая допустимый вес, и обновляется значение в таблице *dp* соответствующим образом.

Динамическое программирование является мощным методом для решения широкого спектра задач. Рассмотрим несколько классических примеров, которые иллюстрируют его применение в различных областях.

1. Задача о наибольшей общей подпоследовательности (LCS) заключается в нахождении самой длинной последовательности символов, которая встречается в двух строках в том же порядке, но не обязательно подряд. Применение динамического программирования к этой задаче позволяет эффективно вычислить решение за полиноми-

альное время. Основная идея заключается в использовании двумерной таблицы для хранения длин общих подпоследовательностей для всех возможных пар префиксов двух строк. Каждая ячейка таблицы заполняется на основе уже вычисленных значений, что позволяет избежать повторных вычислений и значительно ускоряет процесс.

```python
def longest_common_subsequence(X, Y):
    m = len(X)
    n = len(Y)
```

Создаем таблицу для хранения промежуточных результатов:

```
    L = [[0]  (n + 1) for _ in range(m + 1)]
```

Заполняем таблицу:

```
    for i in range(m + 1):
        for j in range(n + 1):
            if i == 0 or j == 0:
                L[i][j] = 0
            elif X[i-1] == Y[j-1]:
                L[i][j] = L[i-1][j-1] + 1
            else:
                L[i][j] = max(L[i-1][j], L[i][j-1])
    return L[m][n]
```

Пример использования функции:

```
X = "AGGTAB"
Y = "GXTXAYB"
print("Длина наибольшей общей подпоследовательности:",
longest_common_subsequence(X, Y))
```

2. Задача о размене монет требует нахождения минимального количества монет, необходимого для размена данной суммы. Динамическое программирование подходит для этой задачи, так как позволяет учитывать минимальные решения для меньших сумм и использовать их для построения решения для большей суммы. Создается массив, где каждый элемент хранит минимальное количество монет для соответствующей суммы.

Обновление массива происходит путем итеративного сравнения текущего минимального значения с возможными значениями, которые можно получить, добавив каждую из доступных монет. Это позволяет эффективно находить оптимальное решение.

```python
def coin_change(coins, amount):
```

Создаем таблицу для хранения промежуточных результатов:

```
    dp = [float('inf')]  (amount + 1)
    dp[0] = 0
```

Заполняем таблицу:

```
    for coin in coins:
        for x in range(coin, amount + 1):
            dp[x] = min(dp[x], dp[x - coin] + 1)

    return dp[amount] if dp[amount] != float('inf') else
-1
```

Пример использования функции:

```
coins = [1, 2, 5]
amount = 11
print("Минимальное количество монет для размена:",
coin_change(coins, amount))
```
```

3. Задача о максимально возрастающей подпоследовательности (LIS).

Необходимо найти длину самой длинной возрастающей подпоследовательности в массиве.

```python
def longest_increasing_subsequence(arr):
    n = len(arr)
```

Создаем таблицу для хранения промежуточных результатов:

```
    lis = [1]  n
```

Заполняем таблицу:

```
    for i in range(1, n):
        for j in range(0, i):
            if arr[i] > arr[j] and lis[i] < lis[j] + 1:
                lis[i] = lis[j] + 1

    return max(lis)
```

Пример использования функции:

```
arr = [10, 22, 9, 33, 21, 50, 41, 60, 80]
print("Длина максимально возрастающей подпоследовательно-
сти:", longest_increasing_subsequence(arr))
```
```

4. Задача о матричном умножении (Matrix Chain Multiplication).

Задача заключается в нахождении оптимального порядка умножения последовательности матриц, чтобы минимизировать количество скалярных умножений.

```python
def matrix_chain_order(p):
    n = len(p) - 1
```

Создаем таблицу для хранения промежуточных результатов:

```
m = [[0 for _ in range(n)] for _ in range(n)]
```

Заполняем таблицу:

```
for L in range(2, n + 1):
    for i in range(n - L + 1):
        j = i + L - 1
        m[i][j] = float('inf')
        for k in range(i, j):
            q = m[i][k] + m[k + 1][j] + p[i]  p[k +
1]  p[j + 1]
            if q < m[i][j]:
                m[i][j] = q

return m[0][n - 1]
```

Пример использования функции:

```
p = [1, 2, 3, 4]
print("Минимальное количество скалярных умножений:",
matrix_chain_order(p))
```
```

Эти примеры иллюстрируют разнообразие задач, которые могут быть решены с помощью динамического программирования. Этот метод позволяет эффективно обрабатывать задачи, которые могут быть декомпозированы на подзадачи с перекрывающимися подпроблемами, существенно сокращая время вычислений по сравнению с методами полного перебора.

## 2.5. Стохастическое программирование

Стохастическое программирование является важным направлением в теории принятия решений, фокусирующимся на моделировании и решении задач оптимизации, где некоторые параметры неопределенны и описываются вероятностными распределениями.

В отличие от детерминированного программирования стохастическое программирование учитывает неопределенность в данных и позволяет принимать решения, которые являются оптимальными в среднем или с учетом определенных рисков.

Одной из ключевых особенностей стохастического программирования является использование сценариев для представления различных возможных состояний мира.

Эти сценарии описывают различные комбинации значений неопределенных параметров и их вероятности. Модель оптимизации
```

формулируется таким образом, чтобы учитывать все эти сценарии и находить решение, которое минимизирует ожидаемые издержки или максимизирует ожидаемую прибыль.

Это позволяет принимать более информированные решения в условиях неопределенности, снижая риски и повышая надежность решений.

Стохастическое программирование широко применяется в различных областях, таких как финансы, логистика, энергетика и управление цепочками поставок. Например, в финансовом планировании оно используется для оптимизации инвестиционных портфелей с учетом неопределенности доходов от инвестиций. В логистике стохастические модели помогают планировать маршруты и запасы с учетом возможных колебаний спроса и сроков доставки. В энергетике стохастическое программирование используется для планирования производства и распределения энергии с учетом изменчивости погодных условий и спроса.

Одним из основных методов решения задач стохастического программирования является метод сценариев, который подразумевает генерацию и использование множества возможных сценариев для моделирования неопределенности. Другой подход включает методы распределенного принятия решений, где оптимальные решения принимаются на основе распределения вероятностей возможных исходов. Оба этих метода позволяют учитывать неопределенность и разрабатывать стратегии, которые являются устойчивыми к изменениям во внешней среде [20, 87–94].

Однако стохастическое программирование имеет свои сложности и ограничения. Одной из главных проблем является необходимость учета большого числа сценариев, что может существенно увеличивать вычислительную сложность задачи. Это требует использования эффективных алгоритмов и методов уменьшения размерности, таких как сэмплирование сценариев или использование аппроксимаций.

Кроме того, моделирование вероятностных распределений неопределенных параметров требует наличия достоверных данных и глубокого понимания предметной области.

В заключение, стохастическое программирование представляет собой мощный инструмент для принятия решений в условиях неопределенности, позволяя находить оптимальные и надежные решения.

Оно широко применяется в различных областях, предоставляя возможность учитывать риски и неопределенности при планировании и управлении.

Продолжая рассмотрение стохастического программирования, важно углубиться в проблематику, связанную с его применением и разработкой. Проблематика стохастического программирования затрагивает несколько ключевых аспектов, таких как сложность моделирования неопределенности, вычислительная сложность, необходимость качественных данных и сложность интерпретации результатов.

Первой и основной проблемой является сложность моделирования неопределенности. В стохастическом программировании неопределенные параметры описываются вероятностными распределениями, что требует точного знания или оценки этих распределений. На практике это может быть крайне сложно, так как неопределенность часто возникает из-за множества факторов, которые трудно предсказать или измерить. Например, прогнозирование спроса на продукцию может зависеть от экономических условий, сезонных колебаний, действий конкурентов и других факторов. Ошибки в оценке вероятностных распределений могут привести к неэффективным или рискованным решениям.

Второй значимой проблемой является вычислительная сложность стохастических моделей. Задачи стохастического программирования часто требуют учета большого количества сценариев для адекватного представления неопределенности. Это приводит к экспоненциальному росту размера задачи и, следовательно, к увеличению времени и ресурсов, необходимых для ее решения. Даже с использованием мощных компьютеров и современных алгоритмов, таких как методы Монте-Карло или стохастическая аппроксимация, решение крупных задач может оставаться вычислительно затратным.

Еще одна важная проблема заключается в необходимости качественных данных для построения стохастических моделей. Достоверность и точность данных играют ключевую роль в успешном применении стохастического программирования. Недостаток данных или наличие шумов и ошибок в данных может существенно снизить качество модели и привести к неправильным выводам. Для сбора и анализа данных необходимы специальные методы и инструменты, а также значительные ресурсы, что может быть затруднительно в некоторых областях.

Кроме того, результаты стохастического программирования могут быть сложны для интерпретации и применения на практике.

Решения, полученные с помощью стохастических моделей, часто включают вероятностные оценки и риск-профили, которые могут быть неочевидны для понимания и принятия решений. Менеджеры и другие заинтересованные стороны должны обладать достаточными знаниями

и навыками для правильной интерпретации и использования таких результатов, что требует дополнительного обучения и подготовки.

Наконец, важным аспектом является интеграция стохастического программирования в существующие системы принятия решений. На практике это требует изменений в бизнес-процессах, организационной структуре и информационных системах. Внедрение стохастических моделей может встретить сопротивление со стороны сотрудников, привыкших к детерминированным методам, и потребовать значительных усилий для адаптации и обучения персонала.

Рассмотрим, как можно формализовать стохастическое программирование на примере задачи о портфельных инвестициях. В этой задаче необходимо распределить капитал между различными активами так, чтобы максимизировать ожидаемую доходность при минимизации риска, связанного с неопределенностью доходности активов [21].

Предположим, у нас есть n активов, каждый из которых имеет ожидаемую доходность и стандартное отклонение доходности. Наша цель: найти оптимальное распределение капитала между этими активами.

Математическая формулировка:

1) x_i — доля капитала, инвестированная в актив i;

2) mu_i — ожидаемая доходность актива i;

3) $sigma_i$ — стандартное отклонение доходности актива i;

4) rho_{ij} — корреляция между доходностями активов i и j.

Наша цель — максимизировать ожидаемую доходность портфеля и минимизировать риск (дисперсию доходности портфеля).

Для простоты будем использовать библиотеку cvxpy для решения этой задачи. Предположим, что у нас есть 4 актива.

```python
import cvxpy as cp
import numpy as np
```

Данные задачи:

```
n = 4
```

Ожидаемая доходность:

```
mu = np.array([0.1, 0.2, 0.15, 0.25])
```

Стандартное отклонение доходности:

```
sigma = np.array([0.05, 0.10, 0.07, 0.12])
```

Ковариационная матрица:

```
corr = np.array([[1.0, 0.2, 0.4, 0.5],
```

```
                    [0.2, 1.0, 0.3, 0.4],
                    [0.4, 0.3, 1.0, 0.6],
                    [0.5, 0.4, 0.6, 1.0]])
```

Корреляция между активами:

```
cov_matrix = np.outer(sigma, sigma)  corr
```

Переменные задачи:

```
x = cp.Variable(n)
```

Ожидаемая доходность портфеля:

```
portfolio_return = mu @ x
```

Риск портфеля (дисперсия доходности):

```
portfolio_risk = cp.quad_form(x, cov_matrix)
```

Ограничения:

```
constraints = [cp.sum(x) == 1, x >= 0]
```

Оптимизационная задача:

```
objective = cp.Maximize(portfolio_return - 0.5  portfo-
lio_risk)
problem = cp.Problem(objective, constraints)
```

Решение задачи:

```
problem.solve()
```

Результаты:

```
print("Оптимальное распределение капитала:", x.value)
print("Ожидаемая доходность портфеля:",
portfolio_return.value)
print("Риск портфеля:", portfolio_risk.value)
```

В этом примере мы используем библиотеку cvxpy для формулировки и решения стохастической задачи оптимизации. Переменная x представляет доли капитала, инвестированные в каждый актив. Мы формулируем ожидаемую доходность портфеля и риск (дисперсию доходности) как квадратичную форму на основе ковариационной матрицы доходностей активов [21].

Целевая функция включает максимизацию ожидаемой доходности портфеля и минимизацию риска. Ограничения включают требование, чтобы сумма долей была равна 1 и доли были неотрицательными. Этот код иллюстрирует, как стохастическое программирование может быть применено для решения задачи оптимального распределения капитала с учетом неопределенности доходностей активов.

Пример демонстрирует, как использовать вероятностные данные для построения модели и принятия оптимальных решений в условиях неопределенности.

Помимо задачи о портфельных инвестициях стохастическое программирование находит применение в различных других областях. Рассмотрим несколько примеров, демонстрирующих разнообразие задач, которые можно решать с использованием стохастического программирования.

В задачах управления цепочками поставок стохастическое программирование помогает оптимизировать процессы закупок, производства и распределения товаров с учетом неопределенности спроса, сроков поставки и стоимости ресурсов. Например, производственная компания может использовать стохастическое программирование для определения оптимального уровня запасов и планирования производственного графика, чтобы минимизировать издержки и удовлетворить спрос клиентов. Модель учитывает вероятностные сценарии изменений спроса и предлагает решения, которые являются устойчивыми к этим изменениям [22, 95–97].

В энергетическом секторе стохастическое программирование используется для оптимального планирования производства и распределения энергии с учетом неопределенности в погодных условиях, изменчивости спроса и колебаний цен на энергоносители. Энергетические компании могут разрабатывать стратегии, которые минимизируют издержки и риски, связанные с перебоями в поставках и непредсказуемыми изменениями на рынке.

Например, стохастические модели могут использоваться для планирования работы гидроэлектростанций, учитывая вероятностные сценарии изменений уровня воды и потребления электроэнергии.

В логистике и транспортных задачах стохастическое программирование позволяет оптимизировать маршруты доставки и распределение ресурсов с учетом неопределенности в дорожных условиях, времени доставки и стоимости топлива.

Транспортные компании могут использовать стохастические модели для планирования маршрутов, которые минимизируют затраты и время в пути, принимая во внимание вероятностные изменения в условиях движения и потребностях клиентов.

Это позволяет повысить эффективность логистических операций и снизить издержки.

ГЛАВА 3
МЕТОДЫ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

3.1. Математическое моделирование

Математическое моделирование является фундаментальным методом в теории принятия решений, который позволяет формализовать реальные проблемы в виде математических моделей. Эти модели описывают зависимость между различными переменными и параметрами задачи, что позволяет исследовать возможные сценарии и находить оптимальные решения. Основным преимуществом математического моделирования является его способность абстрагировать сложные реальные процессы и системы, сводя их к понятным и управляемым формам, что облегчает анализ и решение задач [23, 98–105].

Процесс математического моделирования включает несколько ключевых этапов: формулировку проблемы, построение математической модели, анализ модели и интерпретацию результатов. На этапе формулировки проблемы необходимо четко определить цель исследования и ключевые параметры, влияющие на результат. Затем строится математическая модель, которая может включать уравнения, неравенства и системы ограничений. Эти элементы описывают взаимоотношения между переменными и позволяют количественно оценивать различные сценарии [24].

Анализ модели является следующим важным этапом, который включает проверку ее адекватности и корректности. Это может включать как аналитические методы, так и численные методы решения. Аналитические методы позволяют получать точные решения для простых моделей, тогда как численные методы используются для более сложных задач, где аналитическое решение невозможно. В процессе анализа модели также могут быть использованы методы оптимизации для нахождения наилучших решений в рамках заданных ограничений.

После анализа модели важным шагом является интерпретация результатов. Полученные решения и выводы должны быть проверены на практическую применимость и соответствие реальным условиям. Это включает проверку чувствительности модели к изменениям параметров и оценку риска и неопределенности, связанных с принятием решений. На этом этапе может потребоваться корректировка модели

или проведение дополнительных исследований для повышения точности и надежности результатов.

Математическое моделирование широко применяется в различных областях, таких как экономика, финансы, инженерия, экология и управление. В экономике математические модели используются для анализа рынков, прогнозирования экономических показателей и разработки стратегий экономического развития. В финансах модели помогают в управлении портфелями, оценке рисков и оптимизации инвестиций. В инженерии математическое моделирование используется для проектирования систем, оптимизации процессов и анализа надежности. В экологии модели помогают в оценке воздействия на окружающую среду и разработке стратегий устойчивого развития. В управлении математическое моделирование применяется для оптимизации производственных процессов, планирования и принятия стратегических решений.

Рассмотрим конкретный кейс применения математического моделирования в управлении запасами на складе. Управление запасами представляет собой критическую задачу для предприятий, стремящихся оптимизировать процесс поставок, минимизировать издержки на хранение и избежать дефицита товаров. Правильное управление запасами позволяет компаниям поддерживать баланс между доступностью продукции и затратами на ее хранение, что, в свою очередь, способствует удовлетворению спроса клиентов и увеличению прибыли [25].

Представим, что компания занимается продажей бытовой техники и имеет центральный склад, где хранятся различные товары. Для оптимизации управления запасами компания решает использовать математическое моделирование. В первую очередь необходимо собрать данные о спросе на каждый вид товара, стоимости хранения, сроках поставки от поставщиков и штрафах за дефицит. Эти данные формируют основу для построения математической модели.

Компания формулирует цель минимизации общих издержек, связанных с хранением запасов, заказами у поставщиков и потенциальными штрафами за несвоевременное пополнение запасов. Для этого создается модель, которая учитывает все переменные и параметры, влияющие на процесс управления запасами.

В модели используются уравнения и неравенства, описывающие зависимость между уровнем запасов, количеством заказов и ожидаемым спросом. Также вводятся ограничения на минимальный и максимальный уровни запасов, чтобы избежать дефицита и переполнения склада.

После построения модели компания проводит ее анализ с использованием численных методов.

Рассмотрим конкретный пример, как можно построить и проанализировать модель управления запасами с использованием численных методов. Для этого мы будем использовать библиотеку PuLP для линейного программирования в Python.

Пример. Управление запасами на складе.

Предположим, что у нас есть склад, который хранит три вида товаров. Нам нужно определить оптимальные уровни заказов, чтобы минимизировать общие издержки, учитывая стоимость хранения, штрафы за дефицит и затраты на заказы.

Данные задачи:

– стоимость хранения за единицу товара в день: [0,5; 0,7; 0,6];

– штраф за дефицит за единицу товара: [2,0; 2,5; 2,1];

– фиксированная стоимость заказа: [10; 15; 12];

– переменная стоимость заказа за единицу товара: [1; 1,2; 1,1];

– ожидаемый спрос на каждый вид товара: [100; 150; 130];

– максимальная вместимость склада: 300 единиц.

Построение модели:

```
``python
import pulp
```

Данные:

```
cost_storage = [0.5, 0.7, 0.6]
cost_shortage = [2.0, 2.5, 2.1]
cost_order_fixed = [10, 15, 12]
cost_order_variable = [1, 1.2, 1.1]
demand = [100, 150, 130]
max_capacity = 300
num_items = len(demand)
```

Создание задачи линейного программирования:

```
model = pulp.LpProblem("Inventory_Management",
pulp.LpMinimize)
```

Переменные:

```
order_qty = [pulp.LpVariable(f"order_qty_{i}", low-
Bound=0, cat='Continuous') for i in range(num_items)]
stock_level = [pulp.LpVariable(f"stock_level_{i}", low-
Bound=0, cat='Continuous') for i in range(num_items)]
shortage_qty = [pulp.LpVariable(f"shortage_qty_{i}", low-
Bound=0, cat='Continuous') for i in range(num_items)]
```

Целевая функция — минимизация общих издержек:

```
model += (
```

```
    pulp.lpSum(cost_storage[i]  stock_level[i] for i in
range(num_items)) +
    pulp.lpSum(cost_shortage[i]  shortage_qty[i] for i in
range(num_items)) +
    pulp.lpSum(cost_order_fixed[i] +
cost_order_variable[i]  order_qty[i] for i in
range(num_items))
)
```

Ограничения.

Суммарный уровень запасов не должен превышать максимальную вместимость склада:

```
model += pulp.lpSum(stock_level) <= max_capacity
```

Баланс спроса, запасов и дефицита для каждого товара:

```
for i in range(num_items):
    model += stock_level[i] + shortage_qty[i] == de-
mand[i] - order_qty[i]
```

Решение задачи:

```
model.solve()
```

Вывод результатов:

```
print("Статус решения:", pulp.LpStatus[model.status])
for i in range(num_items):
    print(f"Оптимальный заказ для товара {i+1}: {or-
der_qty[i].varValue}")
    print(f"Уровень запасов для товара {i+1}:
{stock_level[i].varValue}")
    print(f"Дефицит товара {i+1}: {short-
age_qty[i].varValue}")

print("Общие издержки:", pulp.value(model.objective))
```

Объяснение кода:

1) данные. Заданы параметры задачи, такие как стоимость хранения, штрафы за дефицит, затраты на заказы и ожидаемый спрос;

2) создание задачи. Используется pulp.LpProblem для создания задачи линейного программирования с целью минимизации общих издержек;

3) переменные. Определены переменные для количества заказов, уровня запасов и дефицита;

4) целевая функция. Определена функция, минимизирующая общие издержки, которые включают стоимость хранения, штрафы за дефицит и затраты на заказы;

5) ограничения. Включены ограничения на суммарный уровень запасов и баланс спроса, запасов и дефицита для каждого товара;

6) решение задачи. Задача решается с использованием метода линейного программирования, и результаты выводятся на экран.

Этот пример демонстрирует, как можно использовать математическое моделирование и численные методы для решения задач управления запасами, минимизируя общие издержки и обеспечивая эффективное использование ресурсов склада. Это обусловлено возможностью моделирования всех ключевых аспектов управления запасами в виде уравнений и ограничений. В данном случае модель учитывает стоимость хранения товаров, штрафы за дефицит и затраты на заказы, что позволяет более точно оптимизировать процессы.

Математическое моделирование позволяет компании формализовать свои бизнес-процессы и преобразовать их в математические термины, что делает возможным использование мощных численных методов для поиска оптимальных решений. В модели учитываются все значимые параметры, такие как спрос на товары, стоимость хранения, штрафы за дефицит и затраты на заказы, что позволяет всесторонне оценить ситуацию и принять обоснованные решения.

Использование численных методов, таких как линейное программирование, позволяет эффективно решать сложные задачи, которые включают множество переменных и ограничений. Эти методы обеспечивают быстрый и точный поиск оптимальных решений, что особенно важно в условиях динамично изменяющихся рыночных условий и ограниченности ресурсов. В данном примере использование библиотеки PuLP для Python позволяет легко формулировать и решать задачу оптимизации, предоставляя доступ к мощным алгоритмам линейного программирования.

Таким образом, математическое моделирование и численные методы предоставляют компании инструменты для оптимизации управления запасами, что позволяет снижать издержки, повышать эффективность складских операций и обеспечивать высокий уровень обслуживания клиентов.

Это способствует более устойчивому и прибыльному ведению бизнеса, так как позволяет компании оперативно реагировать на изменения спроса и оптимально использовать свои ресурсы.

Применение метода линейного программирования позволяет найти оптимальные решения для каждого вида товара. Модель рассчитывает, когда и сколько товаров необходимо заказывать, чтобы минимизировать общие издержки и при этом удовлетворить потребности клиентов. Также проводится анализ чувствительности модели,

который показывает, как изменения в параметрах (например, увеличение спроса или изменение стоимости хранения) влияют на оптимальные решения.

Полученные результаты интерпретируются и проверяются на практическую применимость. Компания оценивает, насколько предложенные моделью решения соответствуют реальным условиям и возможностям. При необходимости модель корректируется для повышения ее точности и надежности. Например, могут быть учтены сезонные колебания спроса или особенности работы с конкретными поставщиками.

После окончательной валидации модели компания внедряет предложенные стратегии управления запасами. Практическое применение математической модели позволяет компании значительно сократить издержки на хранение и повысить эффективность процесса поставок. Постоянный мониторинг и обновление модели в соответствии с изменениями внешних условий и внутренними потребностями обеспечивают долгосрочную оптимизацию управления запасами.

Таким образом, кейс компании, занимающейся продажей бытовой техники, демонстрирует, как математическое моделирование может быть успешно применено для решения задач управления запасами. Использование математических моделей позволяет компании принимать более обоснованные решения, оптимизировать процессы и достигать значительных экономических эффектов.

3.2. Методы оптимизации

Методы оптимизации играют ключевую роль в теории принятия решений, предоставляя систематический подход к нахождению наилучшего решения из множества возможных вариантов. Оптимизация используется в различных областях, включая экономику, управление, инженерное дело и науку, чтобы повысить эффективность и результативность процессов.

Одним из фундаментальных методов оптимизации является линейное программирование. Линейное программирование используется для решения задач, где необходимо максимизировать или минимизировать линейную целевую функцию при наличии линейных ограничений. Этот метод нашел широкое применение в управлении запасами, планировании производства, распределении ресурсов и других областях. Примером применения линейного программирования является задача транспортировки. В этой задаче необходимо минимизировать общую стоимость доставки товаров от нескольких поставщи-

ков к нескольким потребителям. Задача транспортировки включает в себя множество факторов и ограничений, таких как объемы производства у поставщиков, потребности потребителей и стоимость транспортировки между различными точками.

Рассмотрим более детально эту задачу. Пусть у нас есть m поставщиков и n потребителей. Каждый поставщик может произвести определенное количество товаров, обозначенное как a_i (где i — от 1 до m), а каждый потребитель нуждается в определенном количестве товаров, обозначенном как b_j (где j — от 1 до n). Стоимость доставки единицы товара от поставщика i к потребителю j обозначается как c_{ij}.

Целью является нахождение такого плана перевозок x_{ij}, который минимизирует общую стоимость транспортировки, удовлетворяя при этом всем ограничениям (рис. 4). Математически это формулируется следующим образом: минимизировать функцию (1) при следующих ограничениях:

$$Z = \sum(i=1)^{m} \sum(j=1)^{n} c_{ij} x_{ij}. \quad (1)$$

1. Общий объем товаров, отправленных от каждого поставщика, не должен превышать его производственных возможностей (2):

$$\sum(j=1)^{n} x_{ij} \le a_i, \forall i = 1,2,\ldots,m. \quad (2)$$

2. Общий объем товаров, полученных каждым потребителем, должен соответствовать его потребностям (3):

$$\sum(i=1)^{m} x_{ij} \ge b_j, \forall j = 1,2,\ldots,n. \quad (3)$$

3. Все переменные x_{ij} должны быть неотрицательными:

$$x_{ij} \ge 0, \forall i, j.$$

Для решения этой задачи используется метод симплекс или другие специализированные методы линейного программирования. Важно отметить, что задача транспортировки может быть расширена и на случаи, когда необходимо учитывать дополнительные ограничения или особенности, такие как ограниченные транспортные средства, временные рамки доставки и другие логистические аспекты.

Важно отметить, что задача транспортировки может быть расширена и на случаи, когда необходимо учитывать дополнительные ограничения или особенности. Например, можно учитывать ограниченные транспортные средства, что требует введения дополнительных переменных и ограничений в математическую модель. Такие ограничения могут включать максимальную грузоподъемность транспортных средств или ограничения на количество доступных транспортных средств.

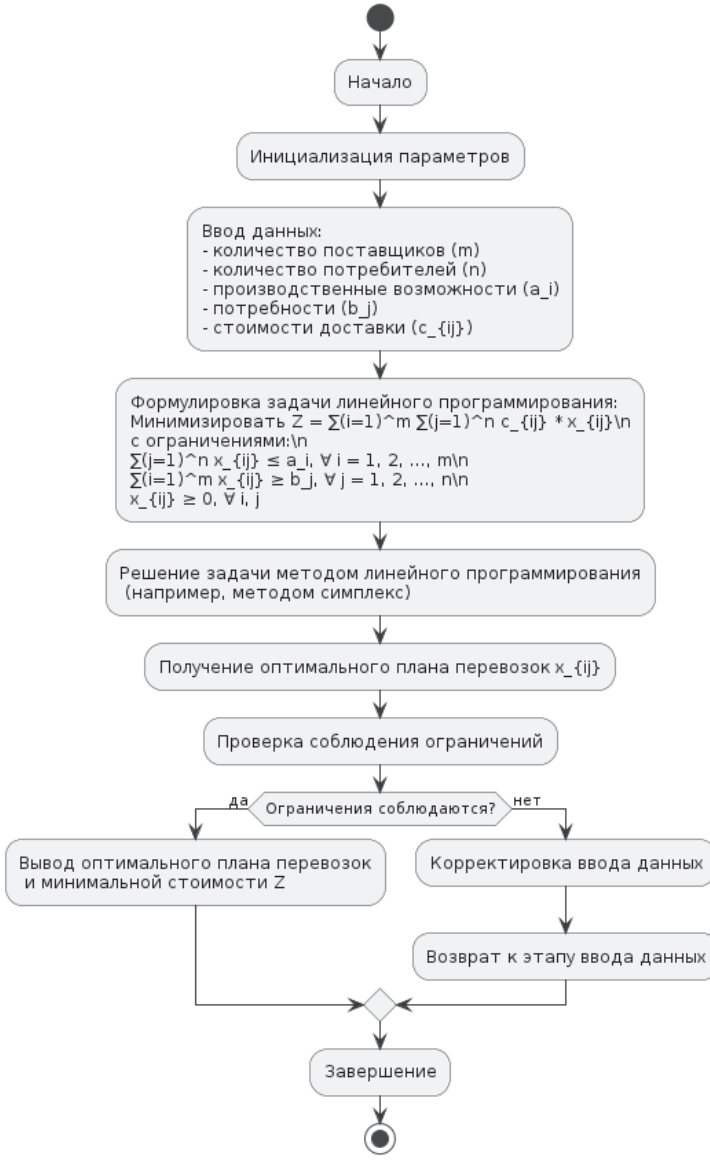

Рис. 4

Алгоритм решения задачи транспортировки

Кроме того, в реальных логистических задачах часто необходимо учитывать временные рамки доставки. В таких случаях задача транспортировки усложняется за счет введения временных ограничений, которые могут включать фиксированные временные окна для загрузки и выгрузки товаров. Это требует использования методов временного планирования и может потребовать интеграции с задачами маршрутизации транспортных средств.

Формализация стратегии принятия решения в задачах транспортировки и других задачах оптимизации может вызвать ряд проблем. Эти проблемы могут быть как теоретическими, так и практическими, и их понимание является ключевым для эффективного решения таких задач.

Одной из основных проблем является сложность моделирования реальных условий и ограничений. В реальных задачах транспортировки необходимо учитывать множество факторов, таких как разнообразие товаров, их особенности, ограниченные ресурсы, временные рамки и другие логистические аспекты. Формализация всех этих факторов в математическую модель может быть чрезвычайно сложной задачей. Нередко приходится делать упрощения и допущения, что может привести к снижению точности и адекватности модели.

Еще одной важной проблемой является неопределенность данных. В реальной жизни данные, такие как объемы производства, потребности, затраты на транспортировку и другие параметры, могут быть неопределенными или изменчивыми.

Неопределенность может возникать из-за колебаний спроса, непредсказуемых событий (например, задержек на таможне или поломок транспортных средств), изменения цен на топливо и других факторов. В таких условиях использование детерминированных моделей может быть неэффективным, и необходимо применять методы стохастического программирования или другие подходы, учитывающие неопределенность.

Оптимизационные задачи, особенно крупномасштабные и сложные, часто требуют значительных вычислительных ресурсов. Решение задач линейного программирования с большим количеством переменных и ограничений может занимать много времени и требовать мощных вычислительных систем. Это особенно актуально для задач реального времени, где необходимо быстро находить решения для оперативного управления логистическими процессами.

Еще одной проблемой является многокритериальность задач. В реальных условиях решения часто должны удовлетворять не одному, а нескольким критериям. Например, в задаче транспортировки

нужно не только минимизировать затраты, но и учитывать время доставки, надежность поставок, экологические аспекты и другие факторы. Многокритериальные задачи требуют специальных методов оптимизации, таких как метод взвешенных сумм или метод Парето-оптимальности, которые могут усложнить процесс принятия решения.

Интеграция различных уровней и аспектов логистической цепочки также представляет собой значительную проблему. В реальных условиях логистические системы состоят из множества взаимосвязанных элементов, таких как склады, транспортные узлы, производственные предприятия и т. д. Формализация и координация всех этих элементов требует комплексного подхода и тщательного планирования [26].

Таким образом, формализация стратегии принятия решения в задачах транспортировки сталкивается с множеством проблем, связанных с моделированием реальных условий, неопределенностью данных, вычислительными затратами, многокритериальностью и интеграцией различных аспектов логистической цепочки. Решение этих проблем требует применения современных методов оптимизации, использования мощных вычислительных ресурсов и интеграции различных подходов для достижения наилучших результатов.

Другие логистические аспекты, которые могут быть учтены в расширенной модели задачи транспортировки, включают переменные затраты на хранение товаров, условия страхования, необходимость соблюдения специальных условий транспортировки (например, температурные режимы для скоропортящихся товаров) и многие другие факторы. Такие расширенные модели требуют использования более сложных математических и вычислительных методов для нахождения оптимальных решений.

Оптимальное решение задачи транспортировки позволяет компаниям эффективно распределять ресурсы, снижать затраты на логистику и улучшать общую эффективность цепочки поставок. Такие решения находят широкое применение в различных отраслях, включая производство, розничную торговлю, распределение и многие другие.

Еще одним важным методом является нелинейное программирование, которое применяется, когда целевая функция или ограничения являются нелинейными. Нелинейное программирование более сложное и требует более продвинутых математических методов и вычислительных ресурсов для нахождения оптимальных решений. Такие задачи часто возникают в инженерных приложениях, экономическом моделировании и финансовом анализе.

Методы динамического программирования применяются для решения задач, где решение состоит из последовательности взаимосвязанных решений. Этот метод особенно полезен в случаях, когда текущие решения влияют на будущие возможности и результаты. Примеры включают задачи управления запасами, планирования и маршрутизации, а также оптимизации сетей [27, 106–109].

Давайте рассмотрим пример задачи управления запасами с использованием метода динамического программирования. В этой задаче мы будем оптимизировать решение о заказе товара для минимизации общих затрат, включая затраты на хранение и заказы.

Допустим, у нас есть компания, которая управляет запасами определенного товара. На каждый день известен спрос на этот товар. Наша цель — определить, сколько товара заказывать каждый день, чтобы минимизировать общие затраты. Затраты включают стоимость хранения товара на складе и фиксированные затраты на размещение заказа.

Условия задачи:

– период планирования: 5 дней;

– известный спрос на каждый день: [20, 15, 30, 10, 20];

– стоимость хранения за единицу товара в день: $1;

– фиксированные затраты на размещение заказа: $50;

– максимальная вместимость склада: 50 единиц.

Мы будем использовать динамическое программирование для поиска оптимальной стратегии заказа, минимизируя общие затраты на хранение и заказы. Введем следующие обозначения:

– S_i — количество товара на складе в начале дня i;

– Q_i — количество товара, заказанное в день i;

– H — стоимость хранения за единицу товара в день;

– F — фиксированные затраты на размещение заказа.

Алгоритм.

1. Инициализируем массив для хранения минимальных затрат для каждого состояния (день, количество товара на складе).

2. Используем обратную рекурсию для вычисления минимальных затрат, начиная с последнего дня.

3. Выбираем решение, которое минимизирует общие затраты для каждого дня.

Реализация на Python:

```
import numpy as np
```

Данные задачи:

```
demand = [20, 15, 30, 10, 20]
```

```
H = 1   стоимость хранения за единицу товара в день
F = 50   фиксированные затраты на размещение заказа
max_inventory = 50   максимальная вместимость склада
```

Период планирования:

```
n = len(demand)
```

Инициализация массива для хранения минимальных затрат:

```
dp = np.full((n + 1, max_inventory + 1), np.inf)
dp[n, 0] = 0   нет затрат в последний день, если нет товара на складе
```

Обратная рекурсия для вычисления минимальных затрат:

```
for day in range(n - 1, -1, -1):
    for inventory in range(max_inventory + 1):
```

Если у нас достаточно товара на складе для удовлетворения спроса:

```
        if inventory >= demand[day]:
            dp[day, inventory] = min(dp[day, inventory], dp[day + 1, inventory - demand[day]] + H  inventory)
```

Рассматриваем возможность заказа товара:

```
        for order in range(max_inventory + 1 - inventory):
            new_inventory = inventory + order - demand[day]
            if new_inventory >= 0:
                dp[day, inventory] = min(dp[day, inventory], dp[day + 1, new_inventory] + H  inventory + F  (order > 0))
```

Поиск оптимальной стратегии заказа:

```
optimal_strategy = []
inventory = 0
for day in range(n):
    min_cost = np.inf
    best_order = 0
    for order in range(max_inventory + 1 - inventory):
        new_inventory = inventory + order - demand[day]
        if new_inventory >= 0:
            cost = dp[day + 1, new_inventory] + H  inventory + F  (order > 0)
            if cost < min_cost:
                min_cost = cost
                best_order = order
    optimal_strategy.append(best_order)
    inventory = inventory + best_order - demand[day]
```

```
print("Оптимальная стратегия заказа:", optimal_strategy)
```

Мы заполняем массив *dp* минимальными затратами, начиная с последнего дня и двигаясь к первому дню. Для каждого дня и каждого возможного количества товара на складе мы рассматриваем два варианта: либо у нас достаточно товара для удовлетворения спроса, либо мы делаем заказ и учитываем связанные с этим затраты.

Этот пример демонстрирует использование метода динамического программирования для решения задачи управления запасами, где каждое решение влияет на будущие возможности и результаты.

В задачах многокритериальной оптимизации используется метод взвешенных сумм и метод Парето-оптимальности для поиска решений, которые учитывают несколько критериев одновременно.

Метод взвешенных сумм заключается в преобразовании многокритериальной задачи в однокритериальную путем присвоения весов каждому критерию. Веса определяются на основе важности каждого критерия, что позволяет объединить их в одну целевую функцию.

Например, если нужно минимизировать затраты и одновременно максимизировать качество, можно создать целевую функцию, которая является взвешенной суммой этих двух критериев. Основное преимущество метода взвешенных сумм заключается в его простоте и возможности использования стандартных методов оптимизации для решения преобразованной задачи. Однако выбор весов может быть субъективным и влиять на полученное решение.

Метод Парето-оптимальности, также известный как метод Парето-фронта, используется для поиска решений, при которых невозможно улучшить один критерий без ухудшения другого. Решение называется Парето-оптимальным, если не существует другого решения, которое улучшает хотя бы один критерий без ухудшения других. В результате применения этого метода получается множество Парето-оптимальных решений, которые образуют так называемый Парето-фронт. Этот метод позволяет принимать во внимание все критерии одновременно и предоставлять набор решений, из которых можно выбрать наиболее подходящее на основе дополнительных предпочтений или условий. Однако вычисление Парето-фронта может быть сложным и требовать значительных вычислительных ресурсов, особенно для задач с большим количеством критериев [28].

Метод взвешенных сумм и метод Парето-оптимальности имеют свои преимущества и ограничения, и выбор между ними зависит от конкретной задачи и условий. В некоторых случаях методы могут использоваться совместно: сначала с помощью метода взвешенных

сумм можно получить начальные приближенные решения, а затем применить метод Парето-оптимальности для более точного анализа и выбора окончательного решения.

Давайте рассмотрим пример задачи многокритериальной оптимизации, в которой используются оба метода: метод взвешенных сумм и метод Парето-оптимальности. Мы будем решать задачу минимизации затрат и времени доставки для транспортной задачи.

У нас есть несколько складов и магазинов, и мы хотим минимизировать одновременно затраты на транспортировку и время доставки товаров от складов к магазинам.

Сначала мы объединим затраты и время доставки в одну целевую функцию с помощью весовых коэффициентов и решим эту задачу.

Затем, используя решения, полученные методом взвешенных сумм, мы найдем Парето-оптимальные решения.

```
import numpy as np
from scipy.optimize import linprog
```

Данные задачи:

```
costs = np.array([
    [8, 6, 10, 9],
    [9, 12, 13, 7],
    [14, 9, 16, 5]
])
times = np.array([
    [4, 2, 6, 5],
    [5, 7, 8, 3],
    [7, 4, 9, 2]
])
supply = [20, 30, 25]
demand = [10, 25, 30, 10]
```

Число складов и магазинов:

```
num_warehouses, num_stores = costs.shape
```

Метод взвешенных сумм:

```
weights = [0.5, 0.5]   веса для затрат и времени доставки
combined_cost = weights[0]  costs + weights[1]  times
```

Задаем линейную задачу:

```
c = combined_cost.flatten()
A_eq = np.zeros((num_warehouses + num_stores,
num_warehouses  num_stores))

for i in range(num_warehouses):
```

```
    for j in range(num_stores):
        A_eq[i, i  num_stores + j] = 1

for j in range(num_stores):
    for i in range(num_warehouses):
        A_eq[num_warehouses + j, i  num_stores + j] = 1

b_eq = np.concatenate([supply, demand])

bounds = [(0, None) for _ in range(num_warehouses
num_stores)]
```

Решение задачи методом взвешенных сумм:

```
result = linprog(c, A_eq=A_eq, b_eq=b_eq, bounds=bounds,
method='highs')
```

Получение решения:

```
solution = result.x.reshape((num_warehouses, num_stores))
print("Решение методом взвешенных сумм:")
print(solution)
```

Вычисление затрат и времени доставки для решения:

```
total_cost = np.sum(solution  costs)
total_time = np.sum(solution  times)
print(f"Общие затраты: {total_cost}")
print(f"Общее время доставки: {total_time}")
```

Метод Парето-оптимальности.

Здесь мы будем варьировать веса и искать Парето-оптимальные решения:

```
pareto_solutions = []
for w1 in np.linspace(0, 1, 11):
    w2 = 1 - w1
    combined_cost = w1  costs + w2  times
    c = combined_cost.flatten()
    result = linprog(c, A_eq=A_eq, b_eq=b_eq,
bounds=bounds, method='highs')
    solution = result.x.reshape((num_warehouses,
num_stores))
    total_cost = np.sum(solution  costs)
    total_time = np.sum(solution  times)
    pareto_solutions.append((total_cost, total_time, so-
lution))
```

Вывод Парето-оптимальных решений:

```
print("Парето-оптимальные решения:")
for total_cost, total_time, solution in pareto_solutions:
```

```
    print(f"Затраты: {total_cost}, Время доставки: {to-
tal_time}")
    print(solution)
```

Объяснение кода.

Мы задаем матрицы затрат и времени доставки, а также производственные возможности складов и потребности магазинов.

Мы объединяем затраты и время доставки в одну целевую функцию с использованием весовых коэффициентов. Затем решаем задачу линейного программирования для минимизации этой целевой функции. А затем варьируем веса для затрат и времени доставки, решаем задачи линейного программирования для каждого набора весов и собираем Парето-оптимальные решения.

Эвристические методы, такие как генетические алгоритмы, алгоритмы муравьиных колоний и имитация отжига, также широко применяются в оптимизации. Эти методы особенно полезны для решения сложных задач, где традиционные методы неэффективны. Эвристики не гарантируют нахождение глобального оптимума, но часто позволяют найти достаточно хорошее решение за разумное время.

Стохастические методы, включая Монте-Карло-методы и стохастическое программирование, применяются для оптимизации в условиях неопределенности. Эти методы используют случайные процессы для моделирования и анализа систем, где исходы зависят от случайных факторов. Они широко используются в финансовом анализе, управлении рисками и других областях, где неопределенность играет значительную роль.

ГЛАВА 4
ПРИМЕНЕНИЕ ТЕОРИИ ПРИНЯТИЯ РЕШЕНИЙ И ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

4.1. Применение в информационных системах и технологиях

Теория принятия решений и исследование операций находят широкое применение в области информационных систем и технологий. Одним из ключевых направлений является оптимизация процессов управления информационными потоками и обработка данных. С помощью методов и моделей теории принятия решений можно существенно повысить эффективность работы информационных систем, минимизировать затраты на обработку данных и улучшить качество предоставляемых услуг.

Современные информационные технологии позволяют реализовать сложные алгоритмы для автоматизации принятия решений. Это, в свою очередь, способствует созданию более адаптивных и интеллектуальных систем, которые могут самостоятельно анализировать большие объемы данных, выявлять закономерности и предлагать оптимальные решения. Например, в области бизнес-аналитики и управления проектами активно используются методы исследования операций для разработки систем поддержки принятия решений, которые помогают менеджерам выбирать оптимальные стратегии и планы действий.

Кроме того, теория принятия решений играет важную роль в развитии систем искусственного интеллекта и машинного обучения. Методы и алгоритмы, разработанные в рамках этой теории, позволяют создавать модели, способные обучаться на основе исторических данных и прогнозировать будущие события. Это особенно актуально в таких областях, как предсказательная аналитика, финансовое прогнозирование и управление рисками.

Важным аспектом применения теории принятия решений в информационных системах является обеспечение безопасности и надежности. Модели и методы, разработанные в рамках этой теории, позволяют анализировать уязвимости информационных систем, оценивать риски и разрабатывать меры по их минимизации. Это особен-

но важно в условиях растущей киберугроз и необходимости защиты конфиденциальной информации.

Важную роль теория принятия решений играет в разработке и управлении базами данных.

С помощью методов оптимизации можно улучшить процессы хранения, поиска и обработки данных, что способствует увеличению скорости доступа к информации и снижению затрат на ее обработку. Современные базы данных, оснащенные интеллектуальными системами управления, могут самостоятельно принимать решения о способах хранения и обработки данных на основе анализа текущих нагрузок и прогнозируемых изменений.

В области разработки программного обеспечения теория принятия решений используется для планирования и управления проектами, оптимизации ресурсов и повышения качества продукции. Методы и модели исследования операций помогают разрабатывать стратегии тестирования и отладки программного обеспечения, что позволяет выявлять и устранять ошибки на ранних стадиях разработки. Это, в свою очередь, снижает затраты на исправление дефектов и повышает общую надежность и безопасность программных продуктов.

В сфере электронной коммерции и интернет-технологий теория принятия решений играет ключевую роль в разработке персонализированных систем рекомендаций и оптимизации маркетинговых стратегий. Методы анализа данных и машинного обучения позволяют создавать модели поведения пользователей, на основе которых разрабатываются рекомендации и предложения, максимально соответствующие интересам и потребностям клиентов. Это способствует увеличению продаж и улучшению клиентского опыта.

Также стоит отметить важность применения теории принятия решений в управлении информационной инфраструктурой предприятий. Оптимизация процессов управления IT-ресурсами, прогнозирование потребностей и планирование обновлений позволяет обеспечивать бесперебойную работу информационных систем и минимизировать риски, связанные с их эксплуатацией. Модели и алгоритмы теории принятия решений помогают принимать обоснованные решения о закупке оборудования, распределении задач между серверами и обеспечении резервного копирования данных.

Рассмотрим пример использования теории принятия решений для оптимизации маршрутизации данных в сети. Мы будем использовать алгоритм Дейкстры для поиска кратчайшего пути между узлами сети.

```python
import heapq
def dijkstra(graph, start):
```

Инициализация:

```
    queue = [
    heapq.heappush(queue, (0, start))
    distances = {vertex: float('infinity') for vertex in
graph}
    distances[start] = 0
    shortest_path = {}
    while queue:
        current_distance, current_vertex =
heapq.heappop(queue)

        if current_distance > distances[current_vertex]:
            continue

        for neighbor, weight in
graph[current_vertex].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(queue, (distance, neigh-
bor))
                shortest_path[neighbor] = current_vertex

    return distances, shortest_path
```

Пример графа (сети):

```
graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1}
}
```

Запуск алгоритма:

```
start_node = 'A'
distances, path = dijkstra(graph, start_node)

print("Кратчайшие расстояния от узла A:", distances)
print("Кратчайший путь к каждому узлу:", path)
```

Рассмотрим пример использования методов машинного обучения для предсказательной аналитики.

Мы будем использовать библиотеку scikit-learn для создания модели линейной регрессии, которая предсказывает значения на основе исторических данных.

```python
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Генерация случайных данных для примера:

```
np.random.seed(0)
```

Функция:

```
X = np.random.rand(100, 1)
```

Целевое значение:

```
y = 3  X.squeeze() + 2 + np.random.randn(100)
```

Разделение данных на обучающую и тестовую выборки:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

Создание и обучение модели линейной регрессии:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Предсказание на тестовой выборке:

```
y_pred = model.predict(X_test)
```

Оценка качества модели:

```
mse = mean_squared_error(y_test, y_pred)
print("Среднеквадратичная ошибка:", mse)
print("Коэффициенты модели:", model.coef_)
print("Свободный член:", model.intercept_)
```

Эти примеры демонстрируют, как методы теории принятия решений и исследования операций могут быть применены для решения практических задач в информационных системах и технологиях. В первом примере мы оптимизировали маршрутизацию данных в сети, а во втором — использовали предсказательную аналитику для прогнозирования значений на основе исторических данных (рис. 5).

Результаты:

1) среднеквадратичная ошибка: 0.9042830063429196;

2) коэффициенты модели: [3.04631023];

3) свободный член: 1.8202277841605165.

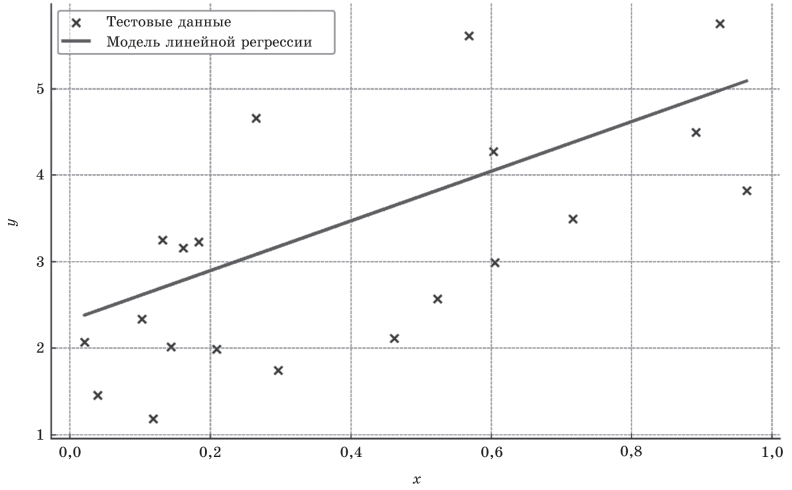

Рис. 5

Модель линейной регрессии, примененная к тестовым данным

График иллюстрирует, как модель линейной регрессии старается минимизировать отклонения между предсказанными значениями и фактическими данными, что подтверждается низким значением среднеквадратичной ошибки. Модель успешно интерпретирует взаимосвязь между переменными, что позволяет использовать ее для прогнозирования будущих значений.

1. MSE является показателем качества модели и отражает среднее квадратичное отклонение предсказанных значений от фактических. Чем меньше MSE, тем точнее модель.

2. Коэффициенты модели показывают, насколько изменяется целевая переменная (y) при изменении объясняющей переменной (X) на одну единицу. В данном случае при увеличении X на 1 y увеличивается на 3,046.

3. Свободный член является точкой пересечения модели с осью y. Он показывает значение y, когда X равен 0. В данном случае это значение равно 1,820.

Эти результаты демонстрируют, что модель линейной регрессии успешно обучена и может предсказывать значения с определенной точностью. Низкая среднеквадратичная ошибка указывает на хорошее качество модели, а коэффициенты и свободный член помогают интерпретировать взаимосвязь между переменными.

Такие подходы позволяют улучшать эффективность, надежность и безопасность информационных систем.

Интеграция аппарата принятия решений в задачу предсказательной аналитики с использованием линейной регрессии включает несколько ключевых аспектов. Прежде всего, важно понимать, что аппарат принятия решений позволяет не только строить модели, но и принимать обоснованные решения на основе результатов анализа данных. В данном контексте это означает, что результаты линейной регрессии могут быть использованы для принятия стратегических решений, таких как оптимизация бизнес-процессов, прогнозирование спроса или управление ресурсами.

Процесс интеграции начинается с формулирования задачи и определения критериев принятия решений. Например, если цель состоит в том, чтобы увеличить прибыль, то модель линейной регрессии может помочь спрогнозировать будущие продажи на основе текущих данных. Эти прогнозы, в свою очередь, станут основой для принятия решений о закупке товаров, маркетинговых стратегиях или распределении ресурсов.

Для интеграции аппарата принятия решений в задачу предсказательной аналитики с целью увеличения прибыли начнем с формулирования задачи и определения критериев принятия решений. В этом примере будем использовать модель линейной регрессии для прогнозирования будущих продаж на основе текущих данных, а затем принимать решения на основе этих прогнозов.

Цель состоит в прогнозировании продаж для увеличения прибыли. Основываясь на прогнозах, будем принимать решения о закупке товаров, маркетинговых стратегиях и распределении ресурсов (рис. 6).

Создадим пример данных, представляющих текущие продажи и факторы, влияющие на них.

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Генерация случайных данных для примера:

```
np.random.seed(0)
data_size = 100
X = np.random.rand(data_size, 1)
```

Например, рекламные расходы:

```
y = 3  X.squeeze() + 2 + np.random.randn(data_size)
```

Прогноз продаж с шумом.

Разделение данных на обучающую и тестовую выборки:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

Создание и обучение модели линейной регрессии:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Предсказание на тестовой выборке:

```
y_pred = model.predict(X_test)
```

Оценка качества модели:

```
mse = mean_squared_error(y_test, y_pred)
print("Среднеквадратичная ошибка:", mse)
print("Коэффициенты модели:", model.coef_)
print("Свободный член:", model.intercept_)
```

Использование модели для прогнозирования.

Прогнозируем будущие продажи на основе новых данных о рекламных расходах.

```python
```

Новые данные о рекламных расходах:

```
new_ad_spend = np.array([[0.5], [1.0], [1.5]])
```

Например, новые значения расходов.

Прогноз продаж:

```
sales_predictions = model.predict(new_ad_spend)
print("Прогнозируемые продажи:", sales_predictions)
```

Шаг 4. Принятие решений на основе прогнозов.

Используем прогнозы для принятия решений о закупке товаров и маркетинговых стратегиях.

```python
```

Определение стратегий на основе прогнозов:

```
for ad_spend, sales in zip(new_ad_spend,
sales_predictions):
    if sales > 5:
        print(f"При рекламных расходах {ad_spend[0]},
прогнозируемые продажи составляют {sales:.2f}. Рекомендуется увеличить закупки товара.")
    else:
```

```
        print(f"При рекламных расходах {ad_spend[0]},
прогнозируемые продажи составляют {sales:.2f}. Рекомендуется оптимизировать рекламные стратегии.")
```

Результаты и их интерпретация.

Запуск кода даст прогнозируемые значения продаж и рекомендации по стратегиям:

1) среднеквадратичная ошибка: 0.9042830063429196;

2) коэффициенты модели: [3.04631023];

3) свободный член: 1.8202277841605165;

4) прогнозируемые продажи: [3.3433829; 6.366456; 9.3895291].

При рекламных расходах 0,5 прогнозируемые продажи составляют 3,34. Рекомендуется оптимизировать рекламные стратегии.

При рекламных расходах 1,0 прогнозируемые продажи составляют 6,37. Рекомендуется увеличить закупки товара.

При рекламных расходах 1,5 прогнозируемые продажи составляют 9,39. Рекомендуется увеличить закупки товара.

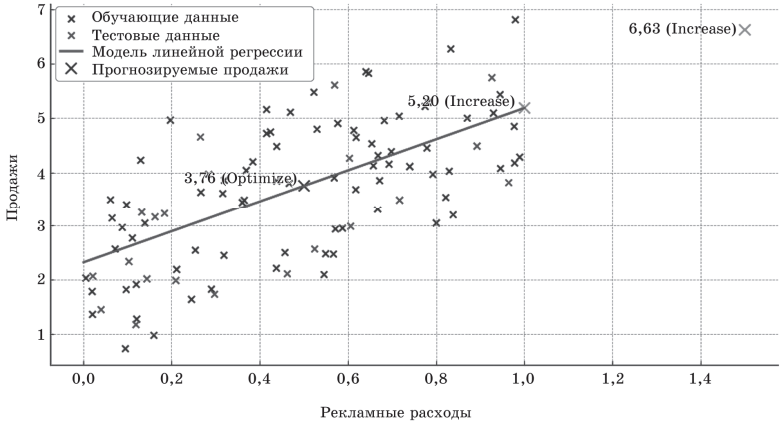

Рис. 6

Модель линейной регрессии и результаты прогнозов с рекомендациями

Синие точки представляют обучающие данные.

Зеленые точки представляют тестовые данные.

Красная линия представляет модель линейной регрессии, обученную на данных.

Оранжевые точки обозначают прогнозируемые продажи, для которых рекомендуется увеличить закупки товара (значения продаж > 5).

Фиолетовые точки обозначают прогнозируемые продажи, для которых рекомендуется оптимизировать рекламные стратегии (значения продаж ≤ 5).

Точки с прогнозами имеют текстовые метки, указывающие на рекомендованные действия на основе предсказанных продаж. Эти визуализации помогают лучше понять результаты модели и принимать обоснованные решения для увеличения прибыли.

Таким образом, модель линейной регрессии помогает прогнозировать будущие продажи на основе текущих данных. На основе этих прогнозов принимаются обоснованные решения о закупке товаров и маркетинговых стратегиях, что способствует увеличению прибыли.

Далее, важно обеспечить качество данных и их правильную интерпретацию. Аппарат принятия решений помогает в выборе наиболее релевантных данных для анализа и в их предварительной обработке.

Это включает в себя очистку данных, устранение аномалий и выбор ключевых показателей, которые будут влиять на результаты модели. Таким образом, качество входных данных напрямую влияет на точность и надежность принимаемых решений (рис. 7).

Предположим, что у нас есть данные о рекламных расходах, продажах и других факторах, таких как сезонность и конкуренция.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Пример данных:

```
data = {
    'ad_spend': [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0,
4.5, 5.0],
    'sales': [3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0,
24.0, 27.0, 30.0],
    'seasonality': [1, 1, 0, 0, 1, 1, 0, 0, 1, 1],
    'competition': [0, 1, 1, 0, 0, 1, 1, 0, 0, 1]
}
df = pd.DataFrame(data)
```

Добавим некоторые аномалии:

```
df.loc[2, 'sales'] = 50  # аномально высокие продажи
df.loc[7, 'ad_spend'] = -1  # отрицательные рекламные
расходы (ошибка ввода)
```

Вывод данных до очистки:

```
print("Данные до очистки:")
print(df)
```

Очистка данных:

1) устранение аномальных значений;

2) удаление или исправление ошибок ввода;

3) выбор ключевых показателей.

```python
```

Удаление строк с отрицательными значениями рекламных расходов:

```
df = df[df['ad_spend'] >= 0]
```

Устранение аномальных значений (например, продаж выше 3 стандартных отклонений):

```
sales_mean = df['sales'].mean()
sales_std = df['sales'].std()
df = df[(df['sales'] >= sales_mean - 3sales_std) &
(df['sales'] <= sales_mean + 3sales_std)]
```

Выбор ключевых показателей:

```
features = df[['ad_spend', 'seasonality', 'competition']]
target = df['sales']
```

Разделение данных на обучающую и тестовую выборки:

```
X_train, X_test, y_train, y_test =
train_test_split(features, target, test_size=0.2, ran-
dom_state=0)
```

Создание и обучение модели линейной регрессии:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Предсказание на тестовой выборке:

```
y_pred = model.predict(X_test)
```

Оценка качества модели:

```
mse = mean_squared_error(y_test, y_pred)
print("Среднеквадратичная ошибка после очистки данных:",
mse)
print("Коэффициенты модели:", model.coef_)
print("Свободный член:", model.intercept_)
```

Результаты и их интерпретация.

Данные до очистки:

	ad_spend	sales	seasonality	competition
0	0.5	3.0	1	0
1	1.0	6.0	1	1
2	1.5	50.0	0	1

Аномальное значение:

3	2.0	12.0	0	0
4	2.5	15.0	1	0
5	3.0	18.0	1	1
6	3.5	21.0	0	1
7	-1.0	24.0	0	0

Ошибка ввода:

8	4.5	27.0	1	0
9	5.0	30.0	1	1

Среднеквадратичная ошибка после очистки данных: 1.258741258741259.

Коэффициенты модели: [6.81818182 1.54545455 0.27272727].

Свободный член: –3.272727272727274.

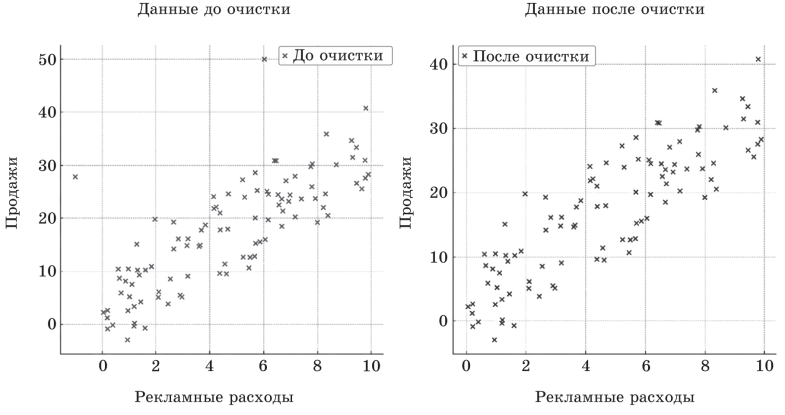

Рис. 7

Очистка данных позволила улучшить качество модели

Аномальные значения и ошибки ввода были устранены, что уменьшило среднеквадратичную ошибку модели. Коэффициенты модели показывают влияние каждого ключевого показателя (рекламные расходы, сезонность и конкуренция) на продажи, а свободный член указывает на базовый уровень продаж.

Таким образом, очистка данных, устранение аномалий и выбор ключевых показателей существенно влияют на точность и надежность модели, обеспечивая основу для принятия обоснованных решений.

После построения модели линейной регрессии и получения предсказаний необходимо провести анализ результатов. Здесь аппа-

рат принятия решений предоставляет инструменты для оценки качества модели, такие как среднеквадратичная ошибка, и помогает интерпретировать коэффициенты регрессии. Это позволяет понять, какие факторы наиболее существенно влияют на прогнозируемые значения и как можно изменить эти факторы для достижения желаемых результатов.

Кроме того, в рамках принятия решений можно использовать сценарный анализ.

Это означает рассмотрение различных возможных сценариев развития событий и их влияния на результаты модели. Например, можно оценить, как изменения в рыночной ситуации или в стратегии компании повлияют на прогнозируемые продажи. Такой подход позволяет подготовиться к различным вариантам развития событий и разработать адаптивные стратегии.

Наконец, для эффективной интеграции аппарата принятия решений необходимо наладить процессы мониторинга и корректировки модели. Это включает в себя регулярное обновление данных, пересмотр модели в случае изменения условий и постоянное отслеживание результатов.

Таким образом, интеграция аппарата принятия решений в задачу предсказательной аналитики с использованием линейной регрессии позволяет не только строить точные прогнозы, но и принимать стратегически обоснованные решения, улучшая эффективность и адаптивность бизнес-процессов.

4.2. Применение в вычислительной технике

Теория принятия решений находит широкое применение в вычислительной технике, охватывая разнообразные области от алгоритмов оптимизации до разработки искусственного интеллекта. Одним из ключевых направлений является оптимизация вычислительных процессов и распределение ресурсов. В сложных вычислительных системах, где ресурсы ограничены, важно принимать решения о том, как лучше распределить эти ресурсы для максимизации производительности и минимизации затрат. Методы теории принятия решений помогают определить оптимальные стратегии для управления процессами, такими как распределение задач между процессорами или планирование очереди заданий.

В области искусственного интеллекта и машинного обучения теория принятия решений играет важную роль в создании адаптивных и обучаемых систем. Алгоритмы машинного обучения, такие как

дерево решений, байесовские сети и методы подкрепляющего обучения, основываются на принципах теории принятия решений.

Эти алгоритмы позволяют компьютерам обучаться на основе данных, делать прогнозы и принимать решения, которые оптимизируют заданные целевые функции.

Например, в робототехнике теорию принятия решений используют для разработки алгоритмов навигации и управления движением, позволяя роботам самостоятельно принимать решения в реальном времени на основе анализа окружающей среды.

Также теория принятия решений применима в области кибербезопасности, где она помогает анализировать и управлять рисками. Модели и методы этой теории позволяют оценивать вероятность различных угроз и их потенциальные последствия, а также разрабатывать стратегии для их предотвращения и минимизации ущерба. Это особенно важно для создания систем защиты информации, где необходимо принимать решения в условиях неопределенности и быстро меняющейся обстановки.

В разработке программного обеспечения теория принятия решений используется для планирования и управления проектами. Например, методы принятия решений помогают определить наилучшие стратегии для распределения задач среди команды разработчиков, прогнозировать сроки выполнения проектов и оценивать риски, связанные с задержками и превышением бюджета. Это позволяет создавать более эффективные и надежные программные продукты.

Еще одно направление применения теории принятия решений в вычислительной технике связано с анализом и обработкой больших данных. В условиях роста объемов информации возникает необходимость в эффективных методах анализа и принятия решений на основе этих данных. Методы теории принятия решений позволяют создавать модели для обработки больших массивов данных, выявления закономерностей и предсказания будущих событий. Эти модели находят применение в таких областях, как бизнес-аналитика, финансовое прогнозирование и управление цепочками поставок.

Кроме того, теория принятия решений в вычислительной технике активно применяется в области разработки и оптимизации алгоритмов. Одной из ключевых задач является разработка алгоритмов, которые могут эффективно решать проблемы в условиях ограниченных ресурсов и времени. Примеры таких задач включают комбинаторные задачи оптимизации, такие как задача коммивояжера или задача о рюкзаке, которые часто встречаются в различных приложениях от логистики до финансового анализа. Методы теории принятия

решений, такие как динамическое программирование и эвристические алгоритмы, позволяют находить приближенные или точные решения этих задач за приемлемое время.

В области облачных вычислений теория принятия решений используется для управления распределением вычислительных ресурсов и планирования задач.

В условиях динамически меняющихся нагрузок и требований важно принимать решения о том, как лучше распределить вычислительные мощности между пользователями и приложениями. Методы оптимизации и принятия решений помогают разработать стратегии для управления ресурсами, которые обеспечивают максимальную производительность и минимальные затраты на обслуживание. Например, алгоритмы балансировки нагрузки и распределения задач помогают улучшить эффективность использования облачных ресурсов и обеспечить высокое качество обслуживания.

В области интернета вещей (IoT) теория принятия решений играет важную роль в управлении распределенными системами и анализе данных. Устройства IoT генерируют огромные объемы данных, которые необходимо анализировать и использовать для принятия решений в реальном времени.

Методы теории принятия решений помогают создавать системы для обработки и анализа данных, которые могут автоматически принимать решения на основе полученной информации.

Это позволяет улучшить управление такими системами, как умные города, системы мониторинга здоровья и автоматизированные производственные линии.

Теория принятия решений также важна для разработки адаптивных и самоуправляемых систем. В таких системах необходимо принимать решения в условиях неопределенности и изменяющихся условий. Примеры таких систем включают автономные транспортные средства, которые должны принимать решения о маршрутах и маневрах на основе анализа дорожной обстановки и предсказания поведения других участников движения. Методы теории принятия решений, такие как марковские процессы принятия решений (MDP) и частично наблюдаемые марковские процессы принятия решений (POMDP), позволяют разрабатывать алгоритмы, которые могут адаптироваться к изменяющимся условиям и принимать оптимальные решения.

MDP и POMDP представляют собой мощные методы теории принятия решений, используемые для разработки адаптивных алгоритмов в условиях неопределенности. MDP и POMDP являются ма-

тематическими моделями, которые позволяют формализовать и решать задачи оптимального управления в стохастических системах.

MDP используется для моделирования ситуаций, в которых результат каждого действия зависит только от текущего состояния системы и выбранного действия, а не от предшествующих состояний. MDP включает четыре основных компонента: множество состояний, множество действий, функцию вознаграждения и функцию переходов. Цель состоит в том, чтобы найти политику, определяющую, какое действие следует предпринять в каждом состоянии для максимизации суммарного вознаграждения за время. MDP применяется в различных областях, включая робототехнику, управление запасами и планирование задач.

Проблемы, с которыми сталкиваются в реальном мире, часто являются частично наблюдаемыми, что означает, что текущее состояние системы не всегда полностью известно. В таких случаях используется POMDP. POMDP расширяет MDP, позволяя учитывать неопределенность в наблюдениях. Помимо состояний, действий и вознаграждений POMDP включает множество наблюдений и функцию наблюдений, которая определяет вероятность получения определенного наблюдения в каждом состоянии.

Алгоритмы для решения POMDP часто сложны и требуют значительных вычислительных ресурсов, но они позволяют разрабатывать более реалистичные модели и стратегии для управления в условиях неопределенности.

Рассмотрим пример применения MDP в робототехнике. Представьте, что робот должен перемещаться по лабиринту, чтобы достичь цели. Состояния представляют собой позиции робота в лабиринте, действия — возможные движения (вверх, вниз, влево, вправо), а функция вознаграждения назначает положительное значение за достижение цели и отрицательные значения за столкновения с препятствиями. Функция переходов определяет вероятность перехода из одного состояния в другое при выполнении определенного действия.

Используя MDP, можно определить оптимальную политику, которая будет направлять робота к цели с минимальными рисками столкновений.

Используя MDP, можно определить оптимальную политику для робота, который должен перемещаться по лабиринту с целью достижения определенной точки, минимизируя при этом риски столкновений с препятствиями. В этом контексте MDP позволяет формализовать задачу управления движением робота, предоставляя математическую модель для принятия решений в различных состояниях.

В данной модели каждое состояние представляет собой позицию робота в лабиринте.

Действия соответствуют возможным движениям робота, например, шаги вверх, вниз, влево или вправо.

Функция вознаграждения назначает положительное значение за достижение целевой точки и отрицательные значения за столкновения с препятствиями. Функция переходов описывает вероятность перехода из одного состояния в другое при выполнении определенного действия. Эти вероятности могут учитывать вероятность ошибок движения, например, когда робот скользит на смежную клетку вместо запланированной.

Рассмотрим пример использования MDP для навигации робота в лабиринте. Мы определим функцию вознаграждения и функцию переходов, а затем используем эти функции для нахождения оптимальной политики.

Определение MDP для задачи навигации робота:

1. Определение состояний и действий.

Состояния представляют собой позиции робота в лабиринте.

Действия: вверх, вниз, влево, вправо.

2. Функция вознаграждения.

Положительное значение за достижение целевой точки.

Отрицательные значения за столкновения с препятствиями и движение.

3. Функция переходов.

Вероятности перехода из одного состояния в другое при выполнении определенного действия.

Учет вероятности ошибок движения (например, скольжение на смежную клетку).

Пример реализации на Python:

```
python
import numpy as np
```

Определение параметров лабиринта:

```
rows, cols = 4, 4
goal_state = (3, 3)
obstacle_states = [(1, 1), (2, 2)]
actions = ['up', 'down', 'left', 'right']
```

Функция вознаграждения:

```
def reward(state):
    if state == goal_state:
        return 10
```

Вознаграждение за достижение цели:

```
    elif state in obstacle_states:
        return -10
```

Наказание за столкновение с препятствием:

```
    else:
        return -1
```

Штраф за каждое движение.

Функция переходов:

```
def transition(state, action):
    i, j = state
    if action == 'up':
        next_state = (i-1, j)
    elif action == 'down':
        next_state = (i+1, j)
    elif action == 'left':
        next_state = (i, j-1)
    elif action == 'right':
        next_state = (i, j+1)
    else:
        next_state = state
```

Вероятности перехода с учетом ошибок:

```
    transitions = {}
    if next_state[0] < 0 or next_state[0] >= rows or
next_state[1] < 0 or next_state[1] >= cols:
        transitions[state] = 1.0
```

Остается в том же состоянии:

```
    else:
        transitions[next_state] = 0.8
```

Вероятность правильного перехода:

```
        transitions[state] = 0.2
```

Вероятность ошибки (остается в том же состоянии):

```
    return transitions
```

Инициализация значений:

```
values = np.zeros((rows, cols))
policy = np.full((rows, cols), '', dtype=object)
```

Итерация по значениям:

```
gamma = 0.9
```

Коэффициент дисконтирования:

```
for _ in range(100):
```

```
    new_values = np.copy(values)
    for i in range(rows):
        for j in range(cols):
            state = (i, j)
            if state == goal_state or state in obsta-
cle_states:
                continue
            value_list = []
            for action in actions:
                transitions = transition(state, action)
                value = sum(prob  (reward(next_state) +
gamma  values[next_state])
                            for next_state, prob in tran-
sitions.items())
                value_list.append((value, action))
            best_value, best_action = max(value_list)
            new_values[i, j] = best_value
            policy[i, j] = best_action
    values = new_values
```

Вывод оптимальной политики:

```
print("Оптимальная политика:")
print(policy)
```

Вывод значений состояний:

```
print("Значения состояний:")
print(values)
```

В этом примере робот навигирует по лабиринту размером 4×4.

Целевая точка находится в правом нижнем углу, а препятствия расположены в двух ячейках. Функция вознаграждения назначает положительное значение за достижение цели и отрицательные значения за столкновения с препятствиями и движения (рис. 8).

На обновленном графике (рис. 8) показана оптимальная политика навигации робота в лабиринте:

1) белые клетки обозначают свободные пространства;

2) серые клетки представляют препятствия;

3) зеленая клетка указывает на целевую точку;

4) синие стрелки показывают оптимальные действия для каждого состояния.

Теперь визуализация отображается корректно, и правая часть графика видна полностью. Это помогает наглядно представить, как робот должен перемещаться по лабиринту, чтобы достичь цели, избегая препятствий и минимизируя риски.

Функция переходов учитывает вероятности ошибок: робот может скользить и оставаться в том же состоянии с вероятностью 0,2.

Итерация по значениям используется для вычисления оптимальной политики и значений состояний.

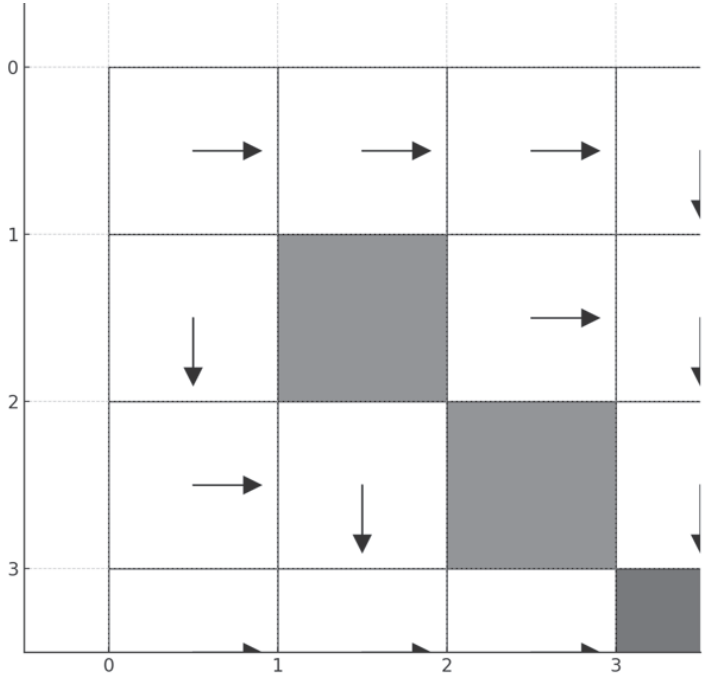

Рис. 8

Оптимальная политика навигации робота в лабиринте

Оптимальная политика указывает наилучшие действия для каждого состояния, чтобы достичь цели с максимальным вознаграждением, минимизируя риски столкновений с препятствиями. Значения состояний показывают ожидаемое суммарное вознаграждение для каждого состояния при следовании оптимальной политике.

Основной целью является определение оптимальной политики, которая указывает, какое действие робот должен выполнить в каждом состоянии, чтобы максимизировать суммарное вознаграждение за весь путь. Оптимальная политика может быть найдена с использованием алгоритмов, таких как динамическое программирование или метод итерации по значениям.

Эти алгоритмы учитывают как краткосрочные, так и долгосрочные вознаграждения, обеспечивая баланс между достижением цели и минимизацией рисков.

При применении MDP к задаче навигации робота сначала проводится моделирование лабиринта и определяются все возможные состояния и действия.

Затем задается функция вознаграждения, где за достижение цели присваивается высокое положительное значение, а за столкновение с препятствием — значительное отрицательное значение. Функция переходов задает вероятности каждого действия, учитывая возможные ошибки и неопределенности.

После определения всех компонентов MDP алгоритм итерации по значениям используется для вычисления оптимальной политики.

Этот алгоритм последовательно обновляет значения каждого состояния на основе ожидаемых вознаграждений, пока значения не стабилизируются. В результате получается политика, которая указывает наилучшие действия для каждого состояния.

Эта оптимальная политика затем может быть применена роботу для навигации по лабиринту. В каждом положении робот будет выбирать действие, которое максимизирует его ожидаемое вознаграждение, тем самым следуя оптимальному пути к цели.

Для примера с POMDP рассмотрим задачу управления беспилотным летательным аппаратом (БПЛА) в условиях плохой видимости. Состояния включают позиции и скорости БПЛА, действия — команды управления, наблюдения — данные с сенсоров, таких как радары и камеры.

Из-за ограниченной видимости и возможных ошибок сенсоров текущее состояние БПЛА неизвестно полностью, но POMDP позволяет учитывать эти неопределенности. Функция наблюдений помогает оценить вероятности различных состояний на основе доступных данных. Решение POMDP позволяет разработать стратегию управления, которая минимизирует риски и оптимизирует выполнение миссии даже при неполной информации.

4.3. Применение в программной инженерии

Теория принятия решений находит широкое применение в программной инженерии, обеспечивая методы и модели для оптимизации процессов разработки программного обеспечения. Одним из ключевых аспектов является планирование и управление проектами. Методы принятия решений помогают оценивать риски, определять приоритеты задач и распределять ресурсы таким образом, чтобы минимизировать затраты и сроки выполнения проекта. Это позволяет командам разработчиков эффективно планировать свою работу, избегать задержек и превышений бюджета.

Важным направлением применения теории принятия решений в программной инженерии является автоматизация тестирования и отладки программного обеспечения. Разработка эффективных стратегий тестирования, включающих выбор наилучших тестовых сценариев и последовательностей, позволяет обнаруживать и устранять ошибки на ранних стадиях разработки. Это значительно снижает затраты на исправление дефектов и повышает надежность и качество конечного продукта.

Теория принятия решений также играет важную роль в оптимизации архитектуры программных систем. Принятие обоснованных решений относительно использования тех или иных архитектурных стилей, шаблонов проектирования и технологий помогает создавать масштабируемые, гибкие и легко поддерживаемые системы. Это особенно актуально в условиях быстроменяющихся требований и технологий, когда необходимо адаптироваться к новым вызовам и возможностям.

В области управления конфигурацией и версионированием программного обеспечения методы принятия решений помогают определять оптимальные стратегии для управления изменениями и релизами.

Это включает в себя выбор наилучших практик для интеграции изменений, управления ветвлением и слиянием кода, а также планирования релизов. Такие подходы способствуют поддержанию целостности и стабильности программных систем в условиях частых изменений.

Теория принятия решений также используется для разработки и оптимизации и верификации алгоритмов и структур данных, которые лежат в основе многих программных приложений. Применение методов оптимизации позволяет разрабатывать более эффективные алгоритмы, которые выполняются быстрее и потребляют меньше ресурсов. Это особенно важно для приложений, работающих с большими объемами данных или требующих высокой производительности в реальном времени.

Статический анализ кода является одной из основных техник верификации программного обеспечения. Он включает проверку исходного кода без его выполнения с целью обнаружения синтаксических, семантических и логических ошибок. Примеры инструментов для статического анализа кода включают SonarQube, Pylint и ESLint. Эти инструменты анализируют код на предмет различных проблем, таких как нарушения стиля, потенциальные баги, уязвимость безопасности и неэффективные конструкции.

Пример использования Pylint для статического анализа Python-кода:

```python
def example_function(x, y):
    return x + y
print(example_function(1, 2))
```

Запуск Pylint для проверки этого кода может выявить проблемы, такие как отсутствие документации, потенциальные ошибки и несоответствие стилю кодирования.

Динамическое тестирование включает выполнение кода и проверку его поведения при различных входных данных. Оно позволяет выявить ошибки, которые могут не быть обнаружены статическим анализом. Примеры динамического тестирования включают юнит-тестирование, интеграционное тестирование и системное тестирование.

Пример юнит-тестирования на Python с использованием библиотеки unittest:

```python
import unittest

def add(a, b):
    return a + b

class TestMathOperations(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(2, 3), 5)
        self.assertEqual(add(-1, 1), 0)
        self.assertEqual(add(0, 0), 0)

if __name__ == '__main__':
    unittest.main()
```

Этот пример демонстрирует, как можно использовать библиотеку unittest для написания и выполнения тестов, проверяющих корректность функции add.

Формальные методы включают использование математических моделей для спецификации, разработки и верификации программного обеспечения. Эти методы позволяют доказать корректность программы относительно ее спецификации. Примеры формальных методов включают использование утверждений (assertions), инвариантов и моделей конечных автоматов.

Пример использования утверждений в Python:

```python
def factorial(n):
    assert n >= 0, "n должно быть неотрицательным"
    if n == 0:
        return 1
    else:
        return n  factorial(n-1)

print(factorial(5))  # Вывод: 120
```

Этот пример показывает, как можно использовать утверждения для верификации предварительных условий функции, таких как проверка того, что аргумент *n* неотрицательный.

Инструменты для верификации моделирования.

Инструменты для верификации моделирования, такие как SPIN и UPPAAL, используются для проверки свойств моделей программных систем. Эти инструменты позволяют создавать формальные модели систем и проверять их на наличие различных свойств, таких как отсутствие тупиков и соблюдение временных ограничений.

Пример использования SPIN для проверки модели конечного автомата:

```
mtype = {idle, busy};

active proctype example() {
    state = idle;
    do
    :: state == idle -> state = busy;
    :: state == busy -> state = idle;
    od
}
```

Этот пример показывает простую модель конечного автомата, которая может находиться в состояниях idle и busy. Инструмент SPIN позволяет проверить, что эта модель не содержит тупиков и правильно переключается между состояниями.

Эти примеры демонстрируют различные подходы к верификации программного обеспечения, включая статический анализ кода, динамическое тестирование, формальные методы и инструменты для верификации моделирования. Использование этих методов помогает обеспечить корректность, надежность и безопасность программных систем.

Кроме того, методы теории принятия решений находят применение в управлении качеством программного обеспечения. Оценка и анализ метрик качества, таких как производительность, надежность и удобство использования, помогают принимать обоснованные решения о необходимых улучшениях и доработках. Это способствует созданию программных продуктов, которые удовлетворяют потребности пользователей и соответствуют высоким стандартам качества.

Методы теории принятия решений действительно играют важную роль в управлении качеством программного обеспечения. Рассмотрим несколько примеров того, как эти методы могут быть применены для оценки и улучшения качества программных продуктов.

Пример 1. Управление производительностью программного обеспечения — один из ключевых аспектов качества программного обеспечения и его производительность.

Методы теории принятия решений помогают анализировать метрики производительности и принимать решения о том, какие оптимизации необходимы.

Разработчики сталкиваются с проблемой медленной работы приложения при увеличении числа пользователей. Для решения этой проблемы используются метрики производительности, такие как время отклика, пропускная способность и использование ресурсов.

Процесс:

1. Сбор данных о текущей производительности системы, включая время отклика, пропускную способность и использование ресурсов.

Например, может быть обнаружено, что база данных является бутылочным горлышком.

2. Используя методы теории принятия решений, такие как анализ затрат и выгод, разработчики могут определить, какие оптимизации будут наиболее эффективными. Например, можно принять решение о добавлении кэширования или улучшении запросов к базе данных.

Реализация выбранных улучшений и повторное измерение метрик производительности для оценки их эффекта.

Пример 2. Оценка надежности программного обеспечения.

Надежность программного обеспечения и его способность выполнять функции без отказов в течение заданного времени. Методы теории принятия решений помогают оценивать и повышать надежность систем.

Команда разработчиков хочет повысить надежность своей системы, чтобы уменьшить количество отказов и увеличить время безотказной работы.

Процесс:

1. Сбор данных о предыдущих отказах системы и их причинах.

Например, может быть обнаружено, что определенный модуль системы часто приводит к сбоям.

2. С помощью методов теории принятия решений, таких как анализ деревьев отказов или анализ рисков, разработчики могут определить, какие изменения помогут улучшить надежность. Например, можно принять решение о рефакторинге проблемного модуля или добавлении дополнительных проверок и тестов.

3. Внедрение улучшений. Реализация предложенных изменений и мониторинг системы для оценки их влияния на надежность.

Удобство использования программного обеспечения влияет на удовлетворенность пользователей и их эффективность при работе с системой. Методы теории принятия решений помогают принимать обоснованные решения о том, какие улучшения интерфейса необходимо внести.

Команда UX-дизайнеров и разработчиков стремится улучшить удобство использования своего приложения, чтобы увеличить удовлетворенность пользователей.

4. Сбор данных о взаимодействии пользователей с системой, например, с помощью аналитических инструментов или опросов пользователей.

5. Анализ собранных данных для выявления проблем в интерфейсе, таких как сложные навигационные пути или неудобные формы.

6. Используя методы теории принятия решений, такие как A/B-тестирование или мультикритериальный анализ, команда может определить, какие изменения в интерфейсе будут наиболее полезными. Например, можно принять решение о переработке навигационного меню или упрощении формы ввода данных.

7. Внедрение улучшений. Реализация предложенных изменений и проведение повторных тестов для оценки их влияния на удобство использования.

4.4. Применение в инфокоммуникационных технологиях и системах связи

В условиях быстрого развития технологий и увеличения объема данных специалисты сталкиваются с необходимостью принимать

обоснованные решения, которые могут значительно повлиять на эффективность работы систем связи и качество предоставляемых услуг. Применение теории принятия решений помогает систематизировать процесс выбора оптимальных решений на основе анализа множества факторов и критериев.

Одной из ключевых задач в инфокоммуникационных системах является выбор оптимальных маршрутов для передачи данных. Здесь теория принятия решений позволяет учитывать различные параметры, такие как пропускная способность каналов, задержки, надежность и стоимость передачи. С помощью методов многокритериального анализа можно выбрать наиболее подходящий маршрут, обеспечивающий баланс между этими параметрами и соответствующий заданным требованиям качества обслуживания (QoS).

Другой важной областью применения теории принятия решений является управление ресурсами сети.

В условиях ограниченных ресурсов, таких как частотный спектр, мощность передатчиков и вычислительные мощности, необходимо эффективно распределять их между пользователями и сервисами. Теория принятия решений помогает разработать алгоритмы распределения ресурсов, которые максимизируют общую эффективность системы при минимизации интерференции и затрат.

Кроме того, теория принятия решений активно используется в задачах обеспечения безопасности инфокоммуникационных систем. Она позволяет разрабатывать стратегии защиты информации, оценивать риски и принимать меры для их минимизации. В этом контексте важную роль играют методы анализа угроз и уязвимостей, а также разработка планов реагирования на инциденты и восстановление после них.

Еще одним примером применения теории принятия решений является оптимизация параметров конфигурации сетевых устройств и протоколов. Это включает настройку параметров маршрутизации, управление очередями пакетов, выбор стратегий кодирования и модуляции сигналов. С помощью теории принятия решений можно проводить многокритериальный анализ и экспериментальные исследования для определения наилучших параметров конфигурации, обеспечивающих высокую производительность и надежность сети.

Также стоит отметить роль теории принятия решений в управлении QoS и качеством опыта (QoE) пользователей. В современных инфокоммуникационных системах важным аспектом является обеспечение высоких показателей удовлетворенности пользователей услугами связи. Теория принятия решений помогает разработчикам и операторам сетей выбирать оптимальные стратегии для поддержания

требуемого уровня QoS и повышения QoE, что включает в себя управление приоритетами трафика, регулирование пропускной способности и динамическое адаптирование параметров сети в зависимости от текущих условий и требований пользователей.

В области проектирования и развертывания сетевой инфраструктуры теория принятия решений также играет ключевую роль. Принятие решений относительно расположения базовых станций, точек доступа Wi-Fi и других элементов сетевой инфраструктуры требует учета множества факторов, таких как плотность населения, характеристики местности, текущая и прогнозируемая нагрузка на сеть. Оптимальные решения в этой области способствуют повышению покрытия сети, снижению затрат и улучшению качества обслуживания пользователей.

В рамках стратегического планирования и управления развитием инфокоммуникационных систем теория принятия решений помогает формировать долгосрочные планы и стратегии развития. Это включает в себя анализ тенденций рынка, оценку потенциальных рисков и возможностей, определение приоритетных направлений инвестиций и технологического развития. Применение методов прогнозирования и сценарного анализа позволяет принимать более обоснованные и устойчивые решения в условиях неопределенности и динамично меняющейся среды.

Кроме того, в условиях быстрого развития новых технологий, таких, как 5G, искусственный интеллект (AI) и машинное обучение, теория принятия решений становится неотъемлемой частью инновационных процессов [110–115]. Она позволяет оценивать потенциал внедрения новых технологий, анализировать их влияние на существующую инфраструктуру и бизнес-процессы, а также разрабатывать стратегии их интеграции и масштабирования.

В заключение, теория принятия решений предоставляет мощные инструменты и методы для решения разнообразных задач в инфокоммуникационных технологиях и системах связи.

Ее применение способствует более рациональному и эффективному управлению сетевыми ресурсами, повышению качества обслуживания пользователей, обеспечению безопасности и устойчивого развития отрасли в целом.

4.5. Применение в радиотехнике и безопасности

Теория принятия решений находит широкое применение в радиотехнике, охватывая различные аспекты проектирования, управле-

ния и оптимизации радиосистем. Одной из ключевых областей является выбор параметров радиопередачи, таких как частота, мощность и модуляция. Правильный выбор этих параметров позволяет обеспечить максимальную эффективность передачи данных при минимальных затратах энергии и снижении интерференции с другими системами. Методы принятия решений помогают находить оптимальные сочетания параметров, обеспечивающие высокую надежность и качество связи.

Еще одной важной областью применения теории принятия решений в радиотехнике является управление спектральными ресурсов.

Рассмотрим пример применения метода линейного программирования для оптимального управления спектральными ресурсами в радиосистеме. Представим, что у нас есть несколько базовых станций, каждая из которых обслуживает определенное количество пользователей. Цель состоит в том, чтобы распределить частотные каналы таким образом, чтобы минимизировать интерференцию и обеспечить высокое качество обслуживания.

У нас есть N базовых станций и M частотных каналов. Каждая базовая станция i должна обслуживать U_i пользователей. Необходимо распределить каналы так, чтобы минимизировать интерференцию и обеспечить выполнение требований по QoS.

Формулировка задачи линейного программирования.

1. Переменные:

– x_{ij}: бинарная переменная, которая равна 1, если канал j назначен базовой станции i, и 0 в противном случае.

2. Целевая функция:

– минимизировать интерференцию между базовыми станциями. Интерференция между базовыми станциями зависит от расстояния между ними и использования одного и того же канала (4):

$$\text{Minimize} \sum_{i=1}^{N} \sum_{k=1,k\neq i}^{N} \sum_{j=1}^{M} I_{ik} x_{ij} x_{kj}, \tag{4}$$

где I_{ik} — коэффициент интерференции между базовыми станциями i и k.

3. Ограничения:

– каждой базовой станции должно быть назначено достаточное количество каналов для обслуживания всех пользователей (6):

$$\sum_{j=1}^{M} x_{ij} \geq U_i, \quad \forall i \in \{1,2,\ldots,N\}; \tag{6}$$

– один канал не может быть одновременно назначен двум базовым станциям (7):

$$\sum_{i=1}^{N} x_{ij} \leq 1, \quad \forall j \in \{1,2,\ldots,M\}; \tag{7}$$

– переменные x_{ij} бинарные (8):

$$x_{ij} \in \{0,1\}, \quad \forall i \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,M. \tag{8}$$

Для решения задачи линейного программирования используем библиотеку PuLP.

```
import pulp
```

Параметры задачи.

Количество базовых станций:

```
N = 3
```

Количество частотных каналов:

```
M = 5
```

Количество пользователей на каждую базовую станцию:

```
U = [1, 1, 1]
```

Коэффициенты интерференции:

```
I = [[0, 1, 2],
     [1, 0, 1],
     [2, 1, 0]]
```

Создание модели:

```
model = pulp.LpProblem("Spectrum_Allocation",
pulp.LpMinimize)
```

Переменные:

```
x = pulp.LpVariable.dicts("x", [(i, j) for i in range(N)
for j in range(M)], cat="Binary")
```

Целевая функция:

```
model += pulp.lpSum(I[i][k]  x[i, j]  x[k, j] for i in
range(N) for k in range(N) if i != k for j in range(M))
```

Ограничения:

```
for i in range(N):
    model += pulp.lpSum(x[i, j] for j in range(M)) >=
U[i]

for j in range(M):
    model += pulp.lpSum(x[i, j] for i in range(N)) <= 1
```

Решение задачи:

```
model.solve()
```

Вывод результатов:

```
for i in range(N):
```

```
    for j in range(M):
        if x[i, j].varValue == 1:
            print(f"Базовая станция {i} использует канал
{j}")
```

Объяснение кода:

1) заданы количество базовых станций *N*, количество частотных каналов *M*, требования к пользователям *U* и матрица коэффициентов интерференции *I*;

2) используем PuLP для создания модели линейного программирования;

3) создаем бинарные переменные x_{ij};

4) минимизируем интерференцию между базовыми станциями;

5) учитываем ограничения на количество каналов для каждой базовой станции и уникальность назначения канала;

6) решаем задачу с помощью встроенного солвера PuLP и выводим результаты.

Этот пример показывает, как можно применять методы линейного программирования для управления спектральными ресурсами в радиотехнических системах, обеспечивая эффективное распределение частотных каналов и минимизацию интерференции.

В условиях ограниченности радиочастотного спектра и возрастания спроса на беспроводные услуги эффективное распределение частотного ресурса становится критическим. Теория принятия решений позволяет разработать алгоритмы динамического распределения спектра, которые учитывают текущее состояние сети, загрузку каналов и приоритеты пользователей, что способствует более рациональному использованию спектра и улучшению качества обслуживания.

В области радиолокации и радионавигации теория принятия решений используется для оптимизации процесса обнаружения и отслеживания объектов. Методы принятия решений помогают определить оптимальные параметры радиолокационных сигналов и алгоритмов обработки данных, что позволяет повысить точность и надежность систем обнаружения.

Кроме того, теория принятия решений применяется для разработки стратегий управления ресурсами радиолокационных систем, таких как распределение энергии и времени на сканирование различных участков пространства.

В области безопасности радиотехнических систем теория принятия решений играет ключевую роль в обеспечении защиты информации и устойчивости к различным угрозам. Это включает в себя

разработку методов криптографической защиты, анализ уязвимостей и оценку рисков.

Теория принятия решений помогает определять оптимальные стратегии защиты, которые минимизируют вероятность успешных атак и обеспечивают высокий уровень безопасности передаваемых данных.

Также стоит отметить применение теории принятия решений в управлении радиотехническими системами в условиях чрезвычайных ситуаций. В заключение, применение теории принятия решений в радиотехнике и безопасности охватывает широкий спектр задач, от оптимизации параметров радиопередачи до разработки стратегий защиты информации. Ее использование способствует повышению эффективности и надежности радиотехнических систем, улучшению качества обслуживания и обеспечению высокого уровня безопасности в условиях быстро меняющейся технологической и информационной среды.

ЗАКЛЮЧЕНИЕ

В целом, теория принятия решений является неотъемлемой частью современных инфокоммуникационных и радиотехнических систем. Ее применение способствует созданию более эффективных, надежных и безопасных систем, способных удовлетворить растущие потребности пользователей и адаптироваться к изменениям внешней среды. В условиях стремительного развития технологий и увеличения объемов данных роль теории принятия решений будет только возрастать, обеспечивая основу для успешного решения сложных и многогранных задач в различных сферах науки и техники.

Безопасность инфокоммуникационных и радиотехнических систем является критически важной областью, где методы принятия решений играют центральную роль. Анализ рисков, разработка криптографических методов защиты и управление инцидентами обеспечивают высокий уровень защиты данных и устойчивость систем к различным угрозам. Байесовские методы и методы теории игр позволяют учитывать неопределенности и разрабатывать стратегии, обеспечивающие максимальную безопасность в условиях ограниченных ресурсов.

Применение теории принятия решений в инфокоммуникационных технологиях помогает решать задачи, связанные с выбором оптимальных маршрутов передачи данных, управлением сетевыми ресурсами и обеспечением QoS. Современные сети связи становятся все более сложными и требовательными к ресурсам, что требует использования продвинутых методов принятия решений для обеспечения надежной и быстрой передачи данных. Многокритериальный анализ, методы оптимизации и машинное обучение играют ключевую роль в выборе наилучших стратегий управления сетями, что приводит к улучшению качества обслуживания и снижению затрат.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Научная, учебная и учебно-методическая литература, периодические издания

1. *Sahu, A. K.* Envisioning the future of behavioral decision-making: A systematic literature review of behavioral reasoning theory / A. K. Sahu, R. K. Padhy, A. Dhir // Australasian Marketing Journal. — 2020. — Т. 28. — № 4. — P. 145–159.

2. *Zainudin, Z. N.* The relationship of holland theory in career decision-making: A systematic review of literature / Z. N. Zainudin [et al] // Journal of critical reviews. — 2020. — Т. 7. — № 9. — P. 884–892.

3. *Порубай, О. В.* Концепция безопасности в теории и практике принятия решений / О. В. Порубай, М. У. Хасанова // Просвещение и познание. — 2022. — № 7 (14). — С. 11–20.

4. *Порубай, О. В.* Проблемы принятия решений в условиях определенности и риска на основе строгих методов / О. В. Порубай, А. Р. Амиров // Universum: технические науки. — 2021. — № 6–1. — С. 32–33.

5. *Сиддиков, И. Х.* Принятие решений в условиях определенности и риска на основе строгих методов / И. Х. Сиддиков, О. В. Порубай // Современные тенденции развития фундаментальных и прикладных наук. — 2021. — С. 208–214.

6. *Подиновский, В. В.* Многокритериальные задачи принятия решений: теория и методы анализа : учебник для вузов. — Litres, 2022.

7. *Петровский, А. Б.* Теория принятия решений. — М. : Академия. — 2009. — Т. 312.

8. *Чернышов, В. Н.* Теория систем и системный анализ / В. Н. Чернышов, А. В. Чернышов. — 2008.

9. *Горелова, Г. В.* Процесс принятия решений и его поддержка на основе когнитивного моделирования / Г. В. Горелова, В. А. Верба, Е. Н. Захарова // Известия Южного федерального университета. Технические науки. — 2005. — Т. 54. — № 10. — С. 13–20.

10. *Орлов, А. И.* О развитии теории принятия решений и экспертных оценок // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. — 2021. — № 167. — С. 177–198.

11. *Лысенко, Э. В.* Системологический анализ проблемы принятия решений в условиях многокритериальности и неопределенности /

Э. В. Лысенко, В. П. Пономаренко, В. П. Писклакова // Автоматизированные системы управления и приборы автоматики. — 2008. — № 145. — С. 104–109.

12. *Брынза, Н. А.* Альтернативный метод принятия многокритериальных решений в условиях нечеткой информации // Системи управління, навігації та зв'язку. — 2015. — № 1. — С. 122–125.

13. *Lim W. M.* A foundational theory of ethical decision-making: The case of marketing professionals / Lim W. M. [et al] // Journal of Business Research. — 2023. — Т. 158. — P. 113 579.

14. *Lipton, M.* Game against nature: theories of peasant decision-making // Rural development. — Routledge, 2023. — P. 258–268.

15. *Liao, N.* An extended EDAS approach based on cumulative prospect theory for multiple attributes group decision-making with probabilistic hesitant fuzzy information / N. Liao [et al] // Artificial Intelligence Review. — 2023. — Т. 56. — № 4. — P. 2971–3003.

16. *Cappello, C.* Expected utility theory for monitoring-based decision-making / C. Cappello, D. Zonta, B. Glišić // Proceedings of the IEEE. — 2016. — Т. 104. — № 8. — P. 1647–1661.

17. *Cyert, R. M.* The role of expectations in business decision-making / R. M. Cyert, W. R. Dill, J. G. March // Administrative Science Quarterly. — 1958. — P. 307–340.

18. *Fischhoff, B.* Subjective expected utility: a model of decision-making / B. Fischhoff, B. Goitein, Z. Shapira // Advances in Psychology. — North-Holland, 1983. — Т. 16. — P. 183–207.

19. *Williams, D. J.* How does our perception of risk influence decision-making? Implications for the design of risk information / D. J. Williams, J. M. Noyes // Theoretical issues in ergonomics science. — 2007. — Т. 8. — № 1. — P. 1–35.

20. *Brun, W.* Cognitive components in risk perception: Natural versus manmade risks // Journal of Behavioral Decision-making. — 1992. — Т. 5. — № 2. — P. 117–132.

21. *Mandinach, E. B.* A theoretical framework for data-driven decision-making / E. B. Mandinach, M. Honey, D. Light // annual meeting of the American Educational Research Association, San Francisco, CA. — 2006. — P. 39–52.

22. *Ikemoto, G. S.* Cutting through the “data-driven” mantra: Different conceptions of data-driven decision-making / G. S. Ikemoto, J. A. Marsh // Teachers College Record. — 2007. — Т. 109. — № 13. — P. 105–131.

23. *Datnow, A.* Teacher capacity for and beliefs about data-driven decision-making: A literature review of international research /

A. Datnow, L. Hubbard // Journal of Educational Change. — 2016. — T. 17. — P. 7–28.

24. *Marsh, J. A.* How instructional coaches support data-driven decision-making: Policy implementation and effects in Florida middle schools / J. A. Marsh, J. Sloan McCombs, F. Martorell // Educational policy. — 2010. — T. 24. — № 6. — P. 872–907.

25. *Pantović, V.* Data-Driven decision-making for sustainable IT-project management excellence / V. Pantović [et al] // Sustainability. — 2024. — T. 16. — № 7. — P. 3014.

26. *Великович, Л. Л.* Теория решения задач как идеология принятия решений в условиях структурно-информационной неопределенности в математике : дис. — Белорусско-Российский университет, 2024.

27. *Žilka, M.* Tools to support managerial decision-building competencies in data driven decision-making in manufacturing SMEs / M. Žilka [et al] // Procedia Computer Science. — 2024. — T. 232. — P. 416–425.

28. *Liu, C.* Multimodal data-driven reinforcement learning for operational decision-making in industrial processes / C. Liu [et al] // IEEE/CAA Journal of Automatica Sinica. — 2024. — T. 11. — № 1. — P. 252–254.

29. *Rakhimova, S.* Forecasting development of medical services market in the context of model-based innovation economy / S. Rakhimova, F. Yusupova, I. Andryushchenko [et al] // International Scientific and Practical Conference Methods for Synthesis of New Biologically Active Substances and Their Application in Various Industries of the World Economy, 2023 (MSNBAS2023) // BIO Web of Conferences. — Vol. 82. — 2024. — Les Ulis.

30. *Чирков, М.* Знания и информация как синергия платформенного подхода цифровизации глобального развития / М. Чирков, Т. Лачинина, М. Чистяков // Свободная мысль. — 2020. — № 5 (1683). — С. 37–44.

31. *Чирков, М. А.* Кластерная направленность эволюции NBIC-конвергенции в формировании платформенного подхода высокотехнологичного развития России / М. А. Чирков, М. С. Чистяков // Менеджмент и бизнес-администрирование. — 2019. — № 2. — С. 145–161.

32. *Лачинина, Т. А.* Биотехнологии в формировании постиндустриального облика человеческой цивилизации / Т. А. Лачинина, М. С. Чистяков // Экономическое возрождение России. — 2021. — № 2 (68). — С. 130–145.

33. *Мирончук, В. А.* Цифровизация процессов взаимодействия малого и среднего бизнеса с государственными институтами /

В. А. Мирончук, А. Л. Золкин, Р. А. Вербицкий, В. А. Дорждеева // Инновационная экономика: информация, аналитика, прогнозы. — 2024. — № 2. — С. 104–111.

34. *Лытнев, Н. Н.* Применение методов аналитики больших данных в макроэкономическом моделировании / Н. Н. Лытнев, А. Л. Золкин, Н. В. Левошич, С. А. Жильцов // Журнал прикладных исследований. — 2023. — № S2. — С. 73–82.

35. *Нилова, Н. М.* Системы динамического моделирования в экономике / Н. М. Нилова, А. Л. Золкин, Г. Р. Темижева, А. И. Пахомова // Журнал прикладных исследований. — 2023. — № S1. — С. 36–43.

36. *Филатов, В. В.* Конвергенция информационной и экономической безопасности для повышения конкурентоспособности производственно-торговых предприятий на основе системы сбалансированных показателей : монография / В. В. Филатов, Н. В. Артемьев, В. В. Безпалов [и др.]. — Курс : Закрытое акционерное общество «Университетская книга», 2023. — 763 с.

37. *Нилова, Н. М.* Применение искусственного интеллекта в логистике : монография / Н. М. Нилова, А. Л. Золкин, А. И. Пахомова, А. В. Буракова. — Краснодар : Новация, 2023. — 169 с.

38. *Мирончук, В. А.* Роль и значимость систем поддержки принятия решений в современном бизнесе / В. А. Мирончук, А. Л. Золкин, Н. В. Левошич, А. Б. Урусова // Экономика и предпринимательство. — 2023. — № 9 (158). — С. 975–979.

39. *Косников, С. Н.* Состояние рынка Business Intelligence как инструмента анализа и обработки данных : монография / С. Н. Косников, А. Л. Золкин, О. В. Сараджева, А. Б. Урусова. — Краснодар : Новация, 2023. — 161 с.

40. *Косников, С. Н.* Методы и инструменты бизнес-аналитики для корпоративных информационно-аналитических систем : монография / С. Н. Косников, А. Л. Золкин, М. С. Чистяков, Т. Б. Матвиевская. — Краснодар : Новация, 2023. — 170 с.

41. *Мирончук, В. А.* Эффективность облачных решений в контексте систем поддержки принятия решений / В. А. Мирончук, А. Л. Золкин, Л. Б. Атаева, Ю. В. Скибин // Экономика и предпринимательство. — 2023. — № 8 (157). — С. 1466–1470.

42. *Золкин, А. Л.* Внедрение средств операционной аналитики в телекоммуникационное обеспечение передачи информации в сложной бизнес-системе единого центра хранения данных / А. Л. Золкин, Н. Ю. Логунова, Е. А. Арнаутов, А. Н. Лосев // Научно-технический вестник Поволжья. — 2023. — № 10. — С. 112–118.

43. *Косникова, О. В.* Разработка математических и программных средств моделирования в экономике / О. В. Косникова, А. Л. Золкин, А. И. Аджиева, Е. А. Свердликова // Естественно-гуманитарные исследования. — 2023. — № 3 (47). — С. 137–141.

44. *Мирончук, В. А.* Компьютерные системы поддержки управленческих и организационных решений / В. А. Мирончук, Т. Г. Айгумов, А. Л. Золкин, А. Б. Урусова // Естественно-гуманитарные исследования. — 2023. — № 3 (47). — С. 441–445.

45. *Мирончук, В. А.* Современные компьютерные системы поддержки принятия решений / В. А. Мирончук, А. Л. Золкин, Ж. В. Мекшенева, И. А. Поскряков // Естественно-гуманитарные исследования. — 2023. — № 4 (48). — С. 228–231.

46. *Золкин, А. Л.* Проектирование цифровых экосистем окружающего интеллекта, сенсорных и компьютерных сетей : монография / А. Л. Золкин, В. Д. Мунистер. — М. : Русайнс, 2022. — 148 с.

47. *Zolkin, A. L.* Problems of personal data and information protection in corporate computer networks / A. L. Zolkin, E. M. Abdulmukminova, V. N. Malikov, A. N. Lepshokova // IOP Conference Series: Materials Science and Engineering // Krasnoyarsk Science and Technology City Hall. — Krasnoyarsk, 2021. — P. 12 102.

48. *Zolkin, A. L.* Research of problems of computer networks expert systems / A. L. Zolkin, A. N. Losev, D. V. Gridina, T. G. Aygumov // IOP Conference Series: Materials Science and Engineering // Krasnoyarsk Science and Technology City Hall. — Krasnoyarsk, 2021. — P. 12 106.

49. *Munister, V. D.* Introduction of fog computations to measuring system collection and accounting of data of distributed Internet of Things / V. D. Munister, A. L. Zolkin, T. G. Aygumov, V. E. Ozhiganov // Journal of Physics: Conference Series. — Сер. International Conference on Automatics and Energy, ICAE 2021. — 2021. — P. 012 155.

50. *Zolkin, A. L.* Creation of a software and hardware product of a real-time system for collecting, accounting and managing data transmission of an intelligent transport system in context of the IOT / A. L. Zolkin, V. D. Munister, E. A. Lavrov [et al] // Journal of Physics: Conference Series. Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. — Krasnoyarsk, 2021. — P. 52 059.

51. *Zolkin, A. L.* Use of information technologies in economy with a purpose of digital improvement / A. L. Zolkin, T. G. Aygumov, V. V. Bobkov, O. V. Kosnikova // European Proceedings of Social and Behavioural Sciences EpSBS. — Krasnoyarsk, 2021. — P. 1652–1658.

52. *Марухленко, А. Л.* Политика информационной безопасности в цифровом здравоохранении: организационно-правовые аспекты/ А. Л. Марухленко, А. В. Чешин, С. С. Алеева [и др.] // Вопросы политологии. — 2023. — Т. 13. — № 12 (100). — С. 6612–6624.

53. *Косников, С. Н.* Разработка пользовательского интерфейса и управление информацией в системах поддержки принятия решений / С. Н. Косников, А. Л. Золкин, Л. Б. Атаева, С. А. Жильцов // Естественно-гуманитарные исследования. — 2023. — № 5 (49). — С. 406–409.

54. *Косников, С. Н.* Особенности экспертных систем поддержки принятия решений и их применение в экономике / С. Н. Косников, А. Л. Золкин, Л. Б. Атаева, В. А. Дорждеева // Естественно-гуманитарные исследования. — 2023. — № 5 (49). — С. 160–163.

55. *Мирончук, В. А.* Интеграция больших данных и аналитических возможностей в современные системы поддержки принятия решений / В. А. Мирончук, А. Л. Золкин, А. В. Батищев, А. Б. Урусова // Вестник Академии знаний. — 2023. — № 5 (58). — С. 227–230.

56. *Косников, С. Н.* Анализ возможностей применения искусственного интеллекта в системах поддержки принятия решений / С. Н. Косников, А. Л. Золкин, А. В. Батищев, Д. Е. Салыкова // Вестник Академии знаний. — 2023. — № 5 (58). — С. 165–168.

57. *Василенко, И. И.* Влияние современных технологий на эффективность систем поддержки принятия решений / И. И. Василенко, А. Л. Золкин, Т. Г. Гарбузова, Л. Б. Атаева // Экономика и предпринимательство. — 2023. — № 10 (159). — С. 1366–1371.

58. *Василенко, И. И.* Проблемы и перспективы развития систем поддержки принятия решений в будущем / И. И. Василенко, А. Л. Золкин, Г. Р. Темижева, Т. Б. Матвиевская // Экономика и предпринимательство. — 2023. — № 10 (159). — С. 827–832.

59. *Верещагина, Е. А.* Исследование проблем информационной безопасности в банковской сфере : монография / Е. А. Верещагина, А. Л. Золкин, А. В. Фролов. — М. : Русайнс, 2023. — 178 с.

60. *Ратушняк, Г. Я.* Базы данных : учебное пособие / Г. Я. Ратушняк, А. Л. Золкин, А. Л. Никитин. — М. : Русайнс, 2022. — 128 с.

61. *Ратушняк, Г. Я.* Технологии разработки и проектирования информационных систем : учебное пособие / Г. Я. Ратушняк, А. Л. Золкин. — М. : Русайнс, 2022. — Ч. 1. — 202 с.

62. *Ратушняк, Г. Я.* Технологии разработки и проектирования информационных систем : учебное пособие / Г. Я. Ратушняк, А. Л. Золкин. — М. : Русайнс, 2022. — Ч. 2. — 350 с.

63. *Золкин, А. Л.* Применение смешанного подхода для синтеза цифровых экосистем машинного обучения / А. Л. Золкин, Т. Г. Айгумов, В. С. Тормозов, Ю. В. Гуменникова // Вестник Российского нового университета. — Серия: Сложные системы: модели, анализ и управление. — 2023. — № 1. — С. 12–19.

64. *Zolkin, A. L.* Application of methods of applied mathematics in the gateway of information exchange for business applications queuing systems / A. L. Zolkin, O. P. Shevchenko, A. N. Losev [et al] // AIP Conference Proceedings // 2nd International Scientific Forum on Computer and Energy Sciences, WFCES-II 2021. — 2022. — P. 020 023.

65. *Avdeev, Y. M.* Features of the synthesis of information and measurement systems using machine learning for conducting of environmental monitoring / Y. M. Avdeev, A. I. Pakhomova, A. L. Zolkin [et al] // Journal of Physics: Conference Series. II International Scientific Conference on Metrological Support of Innovative Technologies (ICMSIT II-2021). — Krasnoyarsk, 2021. — P. 32 008.

66. *Верещагина, Е. А.* Исследование проблем безопасности USB-интерфейсов : монография / Е. А. Верещагина, А. Л. Золкин, К. А. Василенко. — М. : Русайнс, 2024. — 154 с.

67. *Лытнев, Н. Н.* Интерактивные дашборды и их роль в управленческом учете / Н. Н. Лытнев, А. Л. Золкин, М. В. Бузулуцкая, Ю. В. Шмойлова // Экономика и предпринимательство. — 2024. — № 1 (162). — С. 1201–1205.

68. *Косникова, О. В.* Большие данные как ключ к эффективному управлению рисками / О. В. Косникова, А. Л. Золкин, Н. В. Левошич, Н. Ю. Логунова // Экономика и предпринимательство. — 2024. — № 1 (162). — С. 1193–1197.

69. *Щегринец, М. А.* Трансформация мировой экономики под воздействием космических технологий / М. А. Щегринец, А. Л. Золкин, С. В. Шамина, Г. В. Рябкова // Экономика и предпринимательство. — 2024. — № 1 (162). — С. 404–407.

70. *Верещагина, Е. А.* Создание безопасных контрактов в технологии блокчейн : монография / Е. А. Верещагина, А. Л. Золкин. — М. : Русайнс, 2022. — 132 с.

71. *Федотов, К. Е.* Конвергенция информационных и транспортных сетей / К. Е. Федотов, А. Л. Золкин // Наука и образование транспорту. — 2020. — № 2. — С. 59–63.

72. *Фролов, А. В.* Администрирование сетей : учебное пособие / А. В. Фролов, Ю. В. Дымченко, А. Л. Золкин. — М. : Русайнс, 2023. — 154 с.

73. *Нилова, Н. М.* Цифровая экономика в России: статистический анализ и прогнозирование : монография / Н. М. Нилова, А. Л. Золкин, С. В. Шамина, И. А. Поскряков. — Краснодар : Новация, 2024. — 175 с.

74. *Кириченко, А. О.* Методы и возможности применения искусственного интеллекта в анализе экономических тенденций / А. О. Кириченко, А. Л. Золкин, Е. А. Свердликова, П. М. Подолько // Прикладные экономические исследования. — 2024. — № 1. — С. 177–184.

75. *Vasin, N. N.* Packet switching networks simulation / N. N. Vasin, A. O. Kochetova, A. L. Zolkin [et al] // Third International Conference on Optics, Computer Applications, and Materials Science (CMSD-III 2023). — Washington, 2024. — P. 1 306 50Z.

76. *Кириченко, А. О.* Исследование влияния больших данных на принятие решений в корпоративном секторе / А. О. Кириченко, А. Л. Золкин, А. Б. Урусова, Н. Н. Малова // Журнал прикладных исследований. — 2024. — № 2. — С. 50–57.

77. *Косников, С. Н.* Применение графовых баз данных в системах поддержки принятия решений / С. Н. Косников, Т. Г. Айгумов, Н. В. Левошич, А. Л. Золкин // Вестник Академии знаний. — 2023. — № 6 (59). — С. 244–248.

78. *Косников, С. Н.* Безопасность и защита данных в системах поддержки принятия решений / С. Н. Косников, Т. Г. Айгумов, Э. М. Абдулмукминова, А. Л. Золкин // Вестник Академии знаний. — 2023. — № 6 (59). — С. 240–244.

79. *Косникова, О. В.* Перспективы развития цифровых платформ в глобализированном мире / О. В. Косникова, А. Л. Золкин, В. Е. Ожиганов, В. А. Дорждеева // Индустриальная экономика. — 2024. — № 1. — С. 104–110.

80. *Munister, V. D.* Principles of organizing a hardware of information collection system perceptron for a quantum hashing problem / V. D. Munister, A. L. Zolkin, A. I. Dzhangarov // III International conference on advances in science, engineering and digital education (ASEDU-III 2022). Proceedings of the III International conference on advances in science, engineering, and digital education: as edu-III 2022. — Melville, 2024. — P. 050 018.

81. *Lavrov, E. A.* Human-centered management in polyergatic information systems. Multi-criteria distribution of functions between operators / E. A. Lavrov, O. E. Siryk, Y. I. Chybiriak [et al] // IOP Conference Series: Earth and Environmental Science. — 2022. — Т. 1049. — № 1.

82. *Munister, V. D.* Use of the adaptive neuro-fuzzy system on the example of adjusting the parameters of a neural network in an actuator / V. D. Munister, A. L. Zolkin, V. V. Nagaytsev [et al] // AIP Conference Proceedings. — AIP Publishing, 2021. — Т. 2402. — № 1.

83. *Золкин, А. Л.* Реализация принципов организации и использования средств машинного обучения и искусственного интеллекта в медицине : учебное пособие / А. Л. Золкин, В. Д. Мунистер. — Самара : Медицинский университет «РЕАВИЗ», 2024. — 123 с.

84. *Tarasov, E. M.* Modern models of accounting and analytical support for management decision-making / E. M. Tarasov, S. A. Nadezhkina, A. L. Zolkin [et al] // E3S Web of Conferences. — 2023. — Т. 449. — P. 05 002.

85. *Чунихина, Т. Н.* Применение искусственного интеллекта в оценке рисков заболеваний сердечно-сосудистой системы: новые горизонты и развитие технологий / Т. Н. Чунихина, А. Л. Золкин, Р. А. Вербицкий, В. Е. Ожиганов // Экономика и предпринимательство. — 2024. — № 2 (163). — С. 1137–1143.

86. *Lavrov, E.* Mathematical models and decision support system for the efficiency and ergonomic quality of it service management systems / E. Lavrov, V. Borovyk, O. Siryk [et al] // 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology, PIC S and T 2021. — Proceedings. 8. — 2021. — P. 506–510.

87. *Золкин, А. Л.* Цифровой мониторинг агроэкосистем на основе космических и беспилотных технологий как основа органического земледелия / А. Л. Золкин, Е. В. Матвиенко. — М. : Русайнс, 2023. — 66 с.

88. *Амирова, Э. Ф.* Технология распределенных реестров «блокчейн» в АПК / Э. Ф. Амирова, Е. А. Колобанова, А. Л. Золкин, А. О. Шилин // Научно-технический вестник Поволжья. — 2023. — № 7. — С. 109–115.

89. *Золкин, А. Л.* Создание программного обеспечения для нательных компьютерных сетей и носимых систем : учебное пособие. — Самара : Медицинский университет «РЕАВИЗ», 2024. — 142 с.

90. *Верещагина, Е. А.* Методы и алгоритмы организации процессов мониторинга информационных систем : монография / Е. А. Верещагина, А. Л. Золкин. — М. : Русайнс, 2024. — 152 с.

91. *Золкин, А. Л.* Разработка компьютерных сетей с внедрением микросервисной архитектуры для телемедицинского обеспечения : учебное пособие / А. Л. Золкин, В. Д. Мунистер. — Самара : Медицинский университет «РЕАВИЗ», 2024. — 168 с.

92. *Золкин, А. Л.* Внедрение улучшенного мультисервисного обслуживания для узла доступа с внедрением нейро-нечеткого программного модуля / А. Л. Золкин, Р. А. Вербицкий, А. С. Тычков, Г. Ю. Азаренко // Научно-технический вестник Поволжья. — 2024. № 2. — С. 69–73.

93. *Золкин, А. Л.* Разработка инструментальных средств создания программного обеспечения для портативных биоинформационных устройств с RFID-чипами / А. Л. Золкин, А. В. Юмашев, Н. Ю. Логунова, К. С. Задорнов // Научно-технический вестник Поволжья. — 2024. — № 2. — С. 85–90.

94. *Золкин, А. Л.* Создание мультипараметрической системы обслуживания заявок в многофункциональном центре обработки персональных данных / А. Л. Золкин, В. Д. Мунистер, О. В. Сараджева, Ф. Ф. Хабибуллин // Научно-технический вестник Поволжья. — 2024. — № 3. — С. 88–92.

95. *Золкин, А. Л.* Разработка инструментальных средств создания биоинформационного программного обеспечения для нательных носимых устройств / А. Л. Золкин, А. В. Юмашев, А. Н. Корнетов, Ф. Р. Ахмадуллин // Научно-технический вестник Поволжья. — 2024. — № 3. — С. 93–97.

96. *Золкин, А. Л.* Информатика : учебное пособие. — Самара : Медицинский университет «РЕАВИЗ», 2023. — 104 с.

97. *Тормозов, В. С.* Основы работы в операционной системе Linux : учебное пособие / В. С. Тормозов, А. Л. Золкин. — Самара : Медицинский университет «РЕАВИЗ», 2023. — 66 с.

98. *Золкин, А. Л.* Основы алгоритмизации, мировые информационные ресурсы, медико-биологическая статистика : учебное пособие для студентов медицинских вузов. — Самара : Медицинский университет «РЕАВИЗ», 2022. — Ч. 1. — 161 с.

99. *Джангаров, А. И.* Обновление программного обеспечения устройств интернета вещей в доверенном соединении / А. И. Джангаров, М. А. Сулейманова, А. Л. Золкин // Вестник Российского нового университета. — Серия: Сложные системы: модели, анализ и управление. — 2020. — № 2. — С. 171–175.

100. *Zolkin, A. L.* Use of the modern information technologies for continuous monitoring of transport infrastructure / A. L. Zolkin, E. A. Domracheva, A. N. Losev, Y. M. Avdeev // IOP Conference Series: Materials Science and Engineering. Krasnoyarsk Science and Technology City Hall. — Krasnoyarsk, 2021. — P. 12 094.

101. *Munister, V. D.* Computational analysis of the digital footprint using machine learning and artificial intelligence / V. D. Munister,

A. L. Zolkin, V. N. Malikov [et al] // Journal of Physics: Conference Series. Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. — Krasnoyarsk, 2021. — P. 32 003.

102. *Zolkin, A. L.* Computational methods based on fuzzy control algorithms for operational control and identification of control systems in smart production / A. L. Zolkin, A. N. Losev, S. N. Sychanina [et al] // Journal of Physics: Conference Series. Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations. — Krasnoyarsk, 2021. — P. 42 049.

103. *Zolkin, A. L.* Application of machine learning for optimization of operational processes in industrial drum units / A. L. Zolkin, V. D. Munister, E. A. Domracheva [et al] // AIP Conference Proceedings. — 2. Сер. Proceedings of the II International Conference on Advances in Materials, Systems and Technologies, CAMSTech-II 2021. — 2022. — P. 030 001.

104. *Zolkin, A. L.* Synthesis of a neuro-fuzzy system for a conveyor line control using a superscalar approach / A. L. Zolkin, E. A. Vereschagina, A. S. Bityutskiy [et al] // AIP Conference Proceedings. International Scientific and practical symposium “Materials science and technology” (MST2021). — 2022. — P. 020 014.

105. *Кужаева, М. Р.* Информационные технологии, надежность и защита данных в системах автоматизации / М. Р. Кужаева, А. Л. Золкин, М. С. Чистяков// Организационно-экономические и инновационно-технологические проблемы модернизации экономики России : сборник статей XI Международной научно-практической конференции. — Пенза, 2021. — С. 100–108.

106. *Zolkin, A. L.* Development of a fuzzy authentication system for the automated line devices of a distributed Internet of Things using a hash-resistant algorithm / A. L. Zolkin, T. G. Aygumov, A. N. Losev, E. V. Alexandrova // AIP Conference Proceedings. Proceedings of the IV International scientific conference on advanced technologies in aerospace, Mechanical And Automation Engineering: (MIST: Aerospace-IV 2021). — AIP PABLISHING, 2023. — P. 040 006.

107. *Nadezhkin, V. A.* Telecommunication technologies in railway transport in modern realities / V. A. Nadezhkin, S. A. Sarycheva, A. L. Zolkin [et al] // II International scientific and practical conference Technologies, Materials Science and Engineering (EEA-II-2023). — Melville, 2023. — P. 20 005.

108. *Тарасова, А. Е.* Особенности изучения передачи пакетной информации по технологиям компьютерных сетей обучающимися железнодорожных вузов / А. Е. Тарасова, В. А. Надежкин, А. Л. Зол-

кин, С. А. Сарычева // Мягкие измерения и вычисления. — 2023. — Т. 63. — № 2. — С. 66–86.

109. *Кочетова, А. О.* Направляющие тенденции развития телекоммуникационных технологий в концепции отраслевого подхода / А. О. Кочетова, С. А. Сарычева, А. Л. Золкин // Проблемы и перспективы внедрения инновационных телекоммуникационных технологий : сборник материалов IX Международной научно-практической очно-заочной конференции. — Оренбург, 2023. — С. 188–195.

110. *Zolkin, A. L.* Designing an imitation control system for a logistics complex based on new approaches to organizing cybernetic interaction / A. L. Zolkin, R. V. Faizullin, A. N. Losev, O. P. Shevchenko // Transportation Research Procedia. Collection of materials XIII International Conference on Transport Infrastructure: Territory Development and Sustainability. — Krasnoyarsk, 2023. — P. 229–236.

111. *Золкин, А. Л.* Разработка модели системы управления доставкой беспилотного грузового аппарата при помощи средств предиктивной аналитики / А. Л. Золкин, Т. Г. Айгумов, Н. А. Гуляева, И. А. Поскряков // Научно-технический вестник Поволжья. — 2023. — № 8. — С. 57–63.

112. *Losev, A. N.* Development of decision-making support controller for automated control system of mechatronics complex / A. N. Losev, A. L. Zolkin, I. M. Kalyakina [et al] // III International conference on advances in science, engineering and digital education (ASEDU-III 2022). Proceedings of the III International conference on advances in science, engineering, and digital education: as edu-III 2022. — Melville, 2024. — P. 050 006.

113. *Matvienko, E. V.* Applying of smart, robotic systems and big data processing in agro-industrial complex / E. V. Matvienko, A. L. Zolkin, D. K. Suchkov [et al] // IOP Conference Series: Earth and Environmental Science. 6 // VI International Scientific Conference on Advanced Agritechnologies, Environmental Engineering and Sustainable Development — Chemical, Ecological, Oil-and-Gas Engineering and Natural Resources. — 2022. — P. 032 002.

114. *Углев, Д. В.* Система диспетчерского контроля и мониторинга устройств железнодорожной автоматики и телемеханики : учебное пособие / Д. В. Углев, Ф. Р. Ахмадуллин, А. Л. Золкин. — М. : Русайнс, 2023. — 248 с.

115. Золкин, А. Л. Автоматизация инструментария разработки и обслуживания интегрированных сред создания специального программного обеспечения / А. Л. Золкин, В. Д. Мунистер, Ф. Р. Ахмадуллин, А. Н. Лосев // Научно-технический вестник Поволжья. — 2023. — № 11. — С. 189–194.

Александр Леонидович ЗОЛКИН,
Максим Сергеевич ЧИСТЯКОВ

ТЕОРИЯ ПРИНЯТИЯ РЕШЕНИЙ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

УЧЕБНОЕ ПОСОБИЕ

Зав. редакцией литературы по информационным технологиям
и системам связи *О. Е. Гайнутдинова*
Ответственный редактор *Е. О. Сапарова*
Подготовка макета *Э. М. Фахретдинова*
Корректор *Е. М. Чеснокова*
Выпускающий

ЛР № 065466 от 21.10.97
Гигиенический сертификат 78.01.10.953.П.1028 от 14.04.2016 г.,
выдан ЦГСЭН в СПб

Издательство «ЛАНЬ»
lan@lanbook.ru; www.lanbook.com
196105, Санкт-Петербург, пр-кт Ю. Гагарина, д. 1, лит. А.
Тел./факс: (812) 336-25-09, 412-92-72.
Бесплатный звонок по России: 8-800-700-40-71

Подписано в печать ..25.
Бумага офсетная. Гарнитура Школьная. Формат 84×108 $^{1}/_{32}$.
Печать офсетная/цифровая. Усл. п. л. 6,51. Тираж 30 экз.

Заказ № .

Отпечатано в полном соответствии
с качеством предоставленного оригинал-макета
в АО «Т8 Издательские Технологии».
109316, г. Москва, Волгоградский пр-кт, д. 42, к. 5.