

**Автономная некоммерческая организация профессионального образования
«ПЕРМСКИЙ ГУМАНИТАРНО-ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ»
(АНО ПО «ПГТК»)**

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ
МДК 02.04. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ**

для специальности

**09.02.11 Разработка и управление программным обеспечением
(код и наименование специальности)**

Квалификация выпускника

Программист

Форма обучения

Очная

Пермь 2026

Методические рекомендации по выполнению практических работ
МДК 02.04. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ составлен в
соответствии с требованиями Федерального государственного
образовательного стандарта среднего профессионального образования по
специальности 09.02.11 Разработка и управление программным обеспечением

Данные методические рекомендации помогут организовать
самостоятельную деятельность студентов на основе деятельного и
компетентного подходов к обучению, что соответствует ФГОС СПО по
специальности 09.02.11 Разработка и управление программным
обеспечением.

Автор – составитель: Могильникова Н.С., старший преподаватель.

Методические рекомендации по выполнению практических работ предназначен для оценивания достижений запланированных результатов по дисциплине МДК 02.04. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ. Методические рекомендации по выполнению практических работ представляет собой комплект материалов для проведения практических занятий (в форме практической подготовке) и осуществления контроля за выполнением работ.

Методические рекомендации по выполнению практических работ позволяет оценивать:

| Код ОК, ПК | Уметь | Знать | Владеть навыками |
|---|--|--|------------------|
| ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам; | распознавать задачу и/или проблему в профессиональном и/или социальном контексте, анализировать и выделять её составные части определять этапы решения задачи, составлять план действия, реализовывать составленный план, определять необходимые ресурсы выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы владеть актуальными методами работы в профессиональной и смежных сферах оценивать результат и последствия своих действий (самостоятельно или с помощью наставника) | актуальный профессиональный и социальный контекст, в котором приходится работать и жить структура плана для решения задач, алгоритмы выполнения работ в профессиональной и смежных областях основные источники информации и ресурсы для решения задач и/или проблем в профессиональном и/или социальном контексте методы работы в профессиональной и смежных сферах порядок оценки результатов решения задач профессиональной деятельности | |
| ОК 02 Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности; | определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации выделять наиболее значимое в перечне информации, структурировать получаемую информацию, оформлять результаты поиска оценивать практическую значимость результатов поиска применять средства информационных технологий для решения профессиональных задач использовать современное программное обеспечение в профессио- | номенклатура информационных источников, применяемых в профессиональной деятельности приемы структурирования информации формат оформления результатов поиска информации современные средства и устройства информатизации, порядок их применения программное обеспечение в профессиональной деятельности, в том числе цифровые средства психологические основы деятельности коллектива | |

| | | | |
|---|---|--|--|
| | нальной деятельности использовать различные цифровые средства для решения профессиональных задач | | |
| ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста; | грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке проявлять толерантность в рабочем коллективе | правила построения устных сообщений особенности социального и культурного контекста | |
| ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках. | понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы участвовать в диалогах на знакомые общие и профессиональные темы строить простые высказывания о себе и о своей профессиональной деятельности кратко обосновывать и объяснять свои действия (текущие и планируемые) писать простые связные сообщения на знакомые или интересующие профессиональные темы | правила построения простых и сложных предложений на профессиональные темы основные общеупотребительные глаголы (бытовая и профессиональная лексика) лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности особенности произношения правила чтения текстов профессиональной направленности | |
| ПК. 2.1 Проектировать модули программного обеспечения. | проектировать модули, соответствующие бизнес-задачам. создавать архитектурные диаграммы и документацию. определять структуру и интерфейсы модулей анализировать требования к модулю и определять его функциональность проектировать архитектуру модуля, включая выбор подходящих паттернов проектирования | основные принципы проектирования модулей программного обеспечения языки программирования и технологии для реализации модулей паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей методы анализа требований и способов определения функциональности модуля принципы создания интерфейсов для взаимодействия с другими модулями и си- | проектирования модулей ПО с учетом требований заказчика. создания архитектурных диаграмм и спецификаций модулей. определения интерфейсов и взаимодействия модулей в системе. |

| | | | |
|---|--|--|--|
| | и структуры данных создавать диаграммы классов, последовательностей и прочих диаграмм для визуализации проектируемого модуля выбирать подходящие языки программирования и технологии для реализации модуля проектировать интерфейсы программного обеспечения для взаимодействия с другими модулями и системами учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля проводить анализ и оптимизацию проектируемого модуля для повышения его эффективности и качества | стемами принципы обеспечения безопасности, производительности и масштабируемости при проектировании модулей методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества | |
| ПК. 2.2 Разрабатывать модули программного обеспечения. | разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий применять паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей анализировать требования и определять функциональность модуля создавать интерфейсы для взаимодействия с другими модулями и системами обеспечивать безопасность, производительность и масштабируемость при разработке модулей оптимизировать проектируемые модули для повышения их эффективности и качества работать с системой контроля версий улучшать производительность модулей, вы- | язык программирования, основные конструкции, синтаксис паттерны проектирования структуры данных принципы создания интерфейсов для взаимодействия с другими модулями и системами, таких как REST API, SOAP работа с инструментальным программным обеспечением методы оптимизации кода и алгоритмов эффективные алгоритмы и структуры данных для повышения производительности многопоточность в программных модулях методы оптимизации сетевых протоколов для ускорения обмена данными кэширование данных управление памятью техники повышения производительности программного обеспечения | создание модулей программного обеспечения на различных языках программирования отладки и тестирования разработанных модулей применение структурного и объектно-ориентированного программирования оптимизации кода и алгоритмов программных модулей для увеличения производительности мониторинга и анализа производительности приложений |

| | | | |
|---|---|---|--|
| | являя и устраняя узкие места проводить анализ и мониторинг производительности приложений применять инструменты для рефакторинга и оптимизации программного кода | | |
| ПК. 2.3 Выполнять интеграцию модулей и компонентов программного обеспечения. | интегрировать модули и компоненты, обеспечивая их взаимодействие работать с API и устанавливать соединения между компонентами отслеживать и устранять конфликты и ошибки интеграции анализировать и определять зависимости между модулями и компонентами работать с различными форматами данных и протоколами передачи данных | общих принципов функционирования аппаратных, программных и программно-аппаратных средств администрируемой информационно-коммуникационной системы международных стандартов локальных вычислительных сетей методы и подходы к интеграции модулей и компонентов принципы версионирования и управления изменениями при интеграции принципы безопасности при интеграции модулей и компонентов | интеграции программных модулей и компонентов в единое программное решение работы с API и веб-сервисами для взаимодействия между модулями работы с интеграционными платформами и инструментами обеспечения совместимости и стабильности системы |
| ПК. 2.4 Выполнять тестирование и отладку программного обеспечения. | анализировать требования к программному обеспечению и составлять планы тестирования. создавать тестовые сценарии и тест-кейсы для проверки функциональности и соответствия требованиям. выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования. анализировать результаты тестирования и документировать найденные ошибки. разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении. выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования | принципы и методы тестирования программного обеспечения. основы программирования и архитектуры программного обеспечения. основы баз данных и SQL-запросов. инструменты для автоматизации тестирования основы разработки и отладки программного обеспечения на разных языках программирования понятие дефекта программного обеспечения критерии качества ПО виды и типы тестирования ПО техники ручного тестирования техники автоматизированного тестирования жизненный цикл дефекта ПО принципы работы в системе контроля дефектов основные понятия о качестве ПО | отладки программного обеспечения на уровне программных модулей тестирования программного обеспечения формирования тестовых сценариев подготовки тестовых платформ (установка операционной системы, дополнительного ПО и другого по необходимости) оценки объема тестирования ПО с целью определения необходимых ресурсов для его выполнения настройки тестовой среды и аппаратных средств для выполнения тестирования ПО в соответствии с заданием на тестирование в пределах своей компетенции |

| | | | |
|--|--|--|---|
| | использовать системы контроля дефектов ПО составлять отчет о выполнении тестирования ПО | | формирования и представления отчетности о подготовке к выполнению задания на тестирование ПО в соответствии с установленными регламентами выполнения тестовых процедур на тестовых данных |
| ПК. 2.5 Осуществлять документирование программных модулей программного обеспечения. | описывать функциональность модулей в документации создавать диаграммы для иллюстрации работы модулей программировать с использованием комментариев для документирования кода использовать специальные метки/теги для отметки важных частей кода в документации вести журнал изменений и фиксировать обновления программных модулей разбивать модули на логические блоки и описывать каждый блок отдельно включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки проводить регулярное обновление документации при изменении модулей или добавлении нового функционала. | стандарты технической документации принципы документирования программного обеспечения инструменты для создания технической документации и комментирования кода | создания технической документации для модулей документирования кода, API и интерфейсов работы со специализированным ПО по документированию программного кода |

В результате текущей аттестации по МДК 02.04. Математическое моделирование осуществляется проверка сформированности умений и знаний, направленных на формирование соответствующих ФГОС СПО общих и профессиональных компетенций.

Перечень практических работ.

Практическое занятие «Построение простейших математических моделей. Построение простейших статистических моделей»

Цель работы: закрепить практические навыки по построению простейших математических и простейших статистических моделей.

Краткая теория

Построение математической модели процесса, явления или объекта начинается с построения упрощенного варианта модели, в котором учитываются только основные черты. В результате

прослеживаются основные связи между входными параметрами, ограничениями и показателем эффективности. Общего подхода к построению модели нет. В каждом конкретном случае при построении математической модели учитывается большое количество факторов: цель построения модели, круг решаемых задач, точность описания модели и точность выполнения вычислений. Математическая модель должна отражать все существенные факторы, определяющие ее поведение, и при этом быть простой и удобной для восприятия результатов. Каждая математическая модель процесса, явления или объекта в своей основе имеет математический количественный метод.

Применение математических количественных методов для обоснования выбора того или иного управляющего решения во всех областях человеческой деятельности называется *исследованием операций*. Целью исследования операций является нахождение с использованием специального математического аппарата решения, удовлетворяющего заданным условиям. На самом деле при решении практически любой задачи имеется неограниченное количество решений. Множество решений, удовлетворяющих заданным условиям (ограничениям), называется допустимым множеством решений. Выбор из множества допустимых решений одного решения, наилучшего в каком-либо смысле, называемого *оптимальным* решением, и есть задача исследования операций.

Модель — это материальный или идеальный объект, заменяющий оригинал, наделенный основными характеристиками (чертами) оригинала и предназначенный для проведения некоторых действий над ним с целью получения новых сведений об оригинале.

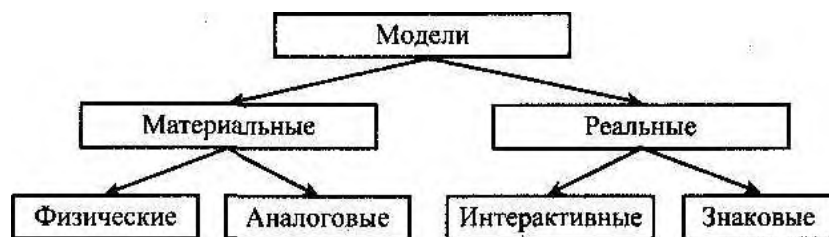


Рис. 1. Классификация моделей



Рисунок 1. Классификация математических моделей

При построении математической модели необходимо обеспечить *достаточную* точность вычислений (точность решения) и *необходимую* подробность модели. Любая математическая модель включает в себя описание основных, т. е. *необходимых* для исследования свойств и законов функционирования исследуемого объекта, процесса или явления. В своей основе каждая математическая модель имеет целевую функцию, которая описывает функционирование реального объекта, процесса или явления. В зависимости от исследуемого (моделируемого) объекта, явления или процесса *целевая функция* может быть представлена одной функциональной зависимостью, системой уравнений (линейных, нелинейных, дифференциальных и т. д.), набором статистических данных и т. д. При работе с целевой функцией исследователь воздействует на нее через **набор входных параметров** (рис. 2).

| | | |
|--------------------------|---|---------------------------|
| Входной параметр 1 | Модель системы (объекта или процесса) | Выходной параметр 1 |
| Входной параметр 2 | | Выходной параметр 2 |
| Входной параметр 3 | | Выходной параметр 3 |
| Входной параметр $n - 1$ | | Выходной параметр $m - 1$ |
| Входной параметр n | | Выходной параметр m |

Рисунок 2. Обобщенная схема математической модели

По способу реализации математические модели можно разделить следующим образом.

1. Линейное программирование.

Математическая модель целиком (целевая функция и ограничения) описывается уравнениями первого порядка. Линейное программирование включает в себя несколько методов решения (задач):

- симплексный;
- графический;
- транспортная задача;
- целочисленное программирование.

2. Нелинейное программирование.

Целевая функция и ограничения, составляющие математическую модель, содержат хотя бы одно нелинейное уравнение (уравнение второго порядка и выше). Нелинейное программирование содержит несколько методов решения (задач):

- графический;
- регулярного симплекса;
- деформируемого многогранника (Нелдера - Мида);
- градиентный.

3. Динамическое программирование.

Ориентировано на решение задач прокладки магистралей кратчайшим путем и перераспределения различных видов ресурсов.

4. Сетевое планирование.

Решает проблему построения графика выполнения работ, распределения производственных, финансовых и людских ресурсов.

5. Принятие решений и элементы планирования.

В этом случае и качестве целевой функции выступает набор статистических данных или некоторые данные прогноза. Решением задачи являются рекомендации о способах поведения (стратегии). Решение носит рекомендательный характер (приблизительное решение). Выбор стратегии целиком остается за человеком — ответственным лицом, принимающим решение. Для принятия решения разработаны следующие теории:

- теория игр;
- системы массового обслуживания.

Задание 1. Составить математическую модель следующей задачи. На складе имеется 300кг

сырья. Надо изготовить два вида продукции. На изготовление первого изделия требуется 2 кг сырья, а на изготовление второго изделия - 5 кг. Определить план выпуска двух изделий.

Задание 2. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал 3-х сортов. При этом на изготовление единицы изделия вида А расходуется 14 кг первого сорта, 12 кг второго сорта и 8 кг третьего сорта. На изготовление продукции вида В расходуется 8 кг первого сорта, 4 кг второго сорта, 2 кг третьего сорта. На складе фабрики имеется всего материала первого сорта 624 кг, второго сорта 541 кг, третьего сорта 376 кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида 7 руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида 3 руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

Задание 3. Составить математическую модель следующей задачи. Имеются три пункта поставки однородного груза А1, А2, А3 и пять пунктов В1, В2, В3, В4, В5 потребления этого груза. На пунктах А1, А2 и А3 находится груз соответственно в количестве 200, 450, 250 тонн. В пункты В1, В2, В3, В4, В5 требуется доставить соответственно 100, 125, 325, 250, 100 тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

| Пункты поставки | Пункты потребления | | | | |
|-----------------|--------------------|----|----|----|----|
| | В1 | В2 | В3 | В4 | В5 |
| А1 | 5 | 8 | 7 | 10 | 3 |
| А2 | 4 | 2 | 2 | 5 | 6 |
| А3 | 7 | 3 | 5 | 9 | 2 |

Контрольные вопросы

1. Что такое модель? Приведите классификацию моделей. Какие вы знаете виды математических моделей? Дайте определение целевой функции.
2. Что такое область допустимых решений? Что называется допустимым решением, оптимальным решением? Какие способы реализации математических моделей вы знаете?

Практическое занятие «Решение простейших однокритериальных задач»

Цель работы: определить оптимальное решение однокритериальных и многокритериальных задач в простейших случаях.

Краткая теория

В зависимости от вида показателя эффективности различают задачи принятия решений по скалярному показателю (однокритериальные задачи) и задачи принятия решений по векторному показателю (многокритериальные задачи).

Задачами **математического программирования** называют **однокритериальные задачи оптимизации**. Методы их решения оперируют с детерминированными математическими моделями. В этих моделях отражены разнообразные проблемы распределения ограниченных ресурсов в экономике, военном деле, создании новой техники и т.д. Пути решения этих проблем, так или иначе, связаны с планированием целенаправленной деятельности, т.е. с разработкой определенных установок на будущее.

Задача математического программирования формулируется следующим образом: найти значения переменных x_1, x_2, \dots, x_n , доставляющие максимум (минимум) заданной целевой функции $y = f(x_1, x_2, \dots, x_n)$ при условиях: $g_j(x_1, x_2, \dots, x_n) \leq (\geq, =) b_j, (j = 1, m)$.

Различают два вида задач математического программирования:

1. Задачи линейного программирования.
2. Задачи нелинейного программирования.

В первых задачах функция y и ограничения g_i линейны относительно переменных x . Во

вторых задачах целевая функция y и (или) условия g_i имеют разного рода нелинейности.

Графоаналитический метод решения задач оптимизации

Этим методом вручную решаются простые задачи оптимизации. Математические модели этих задач не должны быть сложными, т.к. в противном случае требуется много времени для их решения. Для начала рассмотрим однопараметрическую однокритериальную задачу оптимизации.

Постановка задачи: Дан один критерий y . Объект (процесс) описан уравнением (уравнениями), включающими один искомый параметр x . Имеется система ограничений:

$$\begin{aligned} 1) & x \geq a_1; \\ 2) & a_2 \leq x \leq b_1; \end{aligned}$$

и т.д.

Необходимо найти оптимальное значение параметра $x = x_{\text{опт}}$, обращающее целевую функцию $f(x)$ в максимум или минимум.

Задача решается в два этапа:

1. Построение области допустимых решений (ОДР).
2. Нахождение в пределах ОДР оптимального решения.

При построении ОДР на первом этапе рассматривается система ограничений. Все ограничения должны быть выполнены. Выполнение первого ограничения в приведенной выше постановке задачи оптимизации означает, что искомое значение параметра должно находиться правее a_1 , причем a_1 в разрешенный интервал входит (рис.1). Выполнение второго ограничения означает, что искомое значение параметра x должно находиться в интервале (на отрезке) $[a_2, b_1]$, следует иметь в виду, что границы интервала в интервал входят.

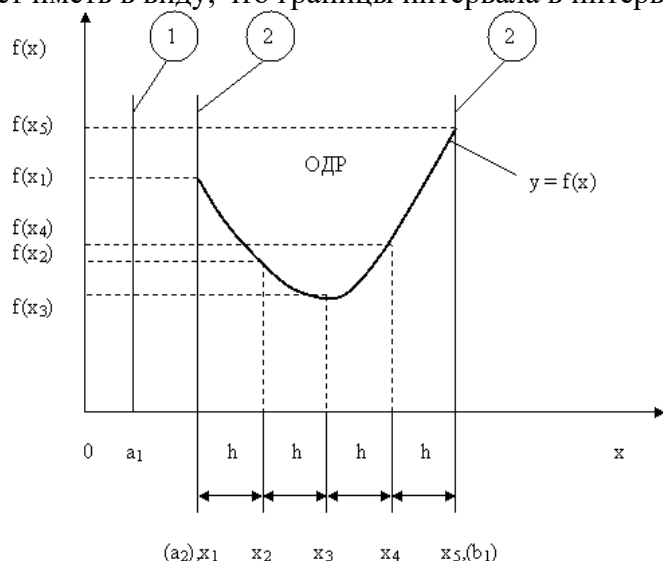


Рис.1. Графическая иллюстрация решения однопараметрической однокритериальной задачи оптимизации

Когда однопараметрическая однокритериальная задача оптимизации решается с применением графоаналитического метода вручную, то на втором этапе применяют метод перебора. Суть его заключается в следующем. В пределах ОДР через определенный интервал h выбирается ряд значений параметра x . В рассматриваемом нами случае ОДР разбита на четыре отрезка, и выбрано пять значений параметра x . Для этих значений параметра x рассчитываются соответствующие значения целевой функции. Среди них находят минимальное (максимальное) значение. Значение параметра x , обращающее целевую функцию в минимум (максимум), является оптимальным. Если в рассматриваемом нами случае $f(x)$ стремится к минимуму, то $x_{\text{опт}} = x_3$, если к максимуму, то $x_{\text{опт}} = x_5$.

При решении практических задач оптимизации всегда следует иметь в виду, какова целевая функция. Это значительно упрощает работу как при решении задач оптимизации вручную с применением графоаналитического метода, так и при решении таких задач с использованием ком-

пьютерных программ. Причем, это относится и к случаю использования готовых программ, и, что особенно важно, к разработке собственных программ.

Рассмотрим, например, следующий частный случай, когда целевая функция линейная (рис.2.).

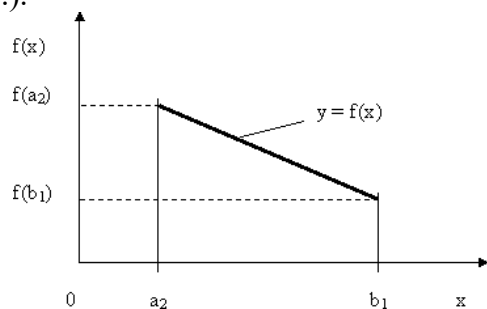


Рис.2. Графическая иллюстрация решения однопараметрической однокритериальной задачи оптимизации для случая линейной целевой функции

В данном случае на втором этапе вычисляют значения целевой функции только на границах ОДР. Эти значения сравнивают и выбирают наименьшее или наибольшее. Для примера, приведенного на рис. 2, если $f(x) \rightarrow \min$, то $x_{\text{опт}} = b_1$, если $f(x) \rightarrow \max$, то $x_{\text{опт}} = a_2$.

В задачах, как правило, присутствует не один, а несколько признаков предпочтения (критериев). Такие задачи называются **многокритериальными**. Критерии могут оказаться противоречивыми, т.е. решение, лучшее по определенному признаку, может оказаться худшим по другому признаку.

Например, минимизация стоимости и максимизация качества товара почти всегда противоречивы. В этом случае задача отыскания решения, предпочтительного по всем признакам, будет **некорректной**, т.е. не будет иметь ни одного решения.

В случае противоречивых критериев имеются следующие подходы к отысканию подходящего решения.

1) **Замена некоторых критериев ограничениями** вида \leq или \geq . Например, минимизация стоимости $f(x) \rightarrow \min$, может быть заменена ограничением вида $f(x) \leq A$, где A некоторая верхняя оценка стоимости, т.е. максимально допустимая стоимость.

2) **Свертка критериев**. Создается один глобальный скалярный критерий, целевая функция которого является некоторой функцией от исходных целевых функций. Наиболее употребимыми являются линейные свертки вида $\alpha f(x) + \beta g(x)$ (в случае двух критериев). Нетривиальной является задача отыскания адекватных значений коэффициентов α и β , отражающих относительную важность целевых функций $f(x)$ и $g(x)$.

3) **Ранжирование критериев**. Критерии ранжируются по степени важности.

4) **Отыскание решений, лучших хотя бы по одному критерию**.

Подходы 1) и 2) приводят к **однокритериальной задаче**. Подход 3) приводит к задаче с **упорядоченными критериями**.

Подход 4) приводит к задаче с **независимыми критериями**. В задаче с упорядоченными критериями критерии упорядочиваются по важности, и требуется найти оптимальное решение для наименее важного критерия на множестве решений, оптимальных для более важного критерия (см. рис 3).

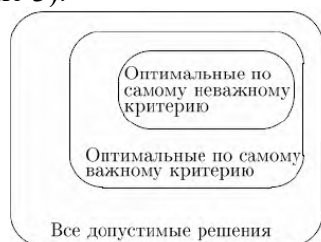


Рис 3. Множество решений.

Это самое большое множество всех допустимых решений, в него вложено множество решений, оптимальных по самому важному критерию, далее вложено множество оптимальных решений по второму по важности критерию, и т.д.

В задаче с независимыми критериями требуется найти множество **недоминируемых (эффективных) решений**. Недоминируемое решение лучше любого другого допустимого решения хотя бы по одному критерию либо не хуже по всем критериям.

Множество недоминируемых решений также называется **множеством Парето**.

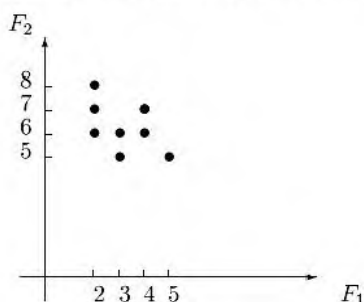
Задание 1. Решить графическим способом задачу. Для производства двух видов, изделия P_1 и P_2 используется, три вида сырья S_1, S_2, S_3 , запасы которого соответственно равны 100, 60, 180 единиц. Для производства одной единицы продукции P_1 используется 2 единицы сырья S_1 и по 1 единице сырья S_2 и S_3 . Для производства одной единицы продукции P_2 используется по 1 единице сырья S_1 и S_2 и 4 единицы сырья S_3 . Прибыль от реализации 1 единицы каждой продукции P_1 и P_2 соответственно равна 30 и 20 единиц. Необходимо составить такой план выпуска продукции P_1 и P_2 , при котором суммарная прибыль будет наибольшей.

Задание 2. Фирме необходимо выбрать наилучший вариант закупки оборудования, если задана закупочная цена каждого из вариантов оборудования и время изготовления и доставки. Под наилучшим вариантом понимается вариант с минимальной закупочной стоимостью и временем доставки.

Обозначим, соответственно, через x_i - номер, $F_1(x_i)$ - время изготовления и доставки, $F_2(x_i)$ - закупочную стоимость варианта закупки оборудования. Значения функций $F_1(x_i)$ и $F_2(x_i)$ заданы таблицей:

| | | | | | | | | |
|------------|---|---|---|---|---|---|---|---|
| x_i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $F_1(x_i)$ | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 5 |
| $F_2(x_i)$ | 6 | 7 | 8 | 5 | 6 | 6 | 7 | 5 |

Задача отыскания множества Парето в случае двух критериев вида $F_1(x) \rightarrow \min$ и $F_2(x) \rightarrow \min$ может быть решена графически следующим образом. Находим все точки с наименьшим значением $F_1(x)$.



Если их несколько, выбираем из них точку с наименьшим значением $F_2(x)$. Включаем ее в множество Парето. Отсекаем точки с большими либо равными значениями $F_1(x)$ и $F_2(x)$ (северо-восточный угол с вершиной в выбранной точке). Повторяем процедуру для оставшейся части допустимой области.

Из рисунка видно, что для нас представляют интерес пары $(F_1, F_2) \in \{(2, 6), (3, 5)\}$ и соответствующие решения $(x_1, x_2) \in \{(2, 2), (1, 2)\}$, которые являются недоминируемыми и образуют множество Парето рассматриваемой задачи.

Контрольные вопросы

1. Какие задачи называются однокритериальными?
2. Какие задачи называются многокритериальными?

3. Какие способы решения однокритериальных задач вы знаете?
4. Какие подходы к отысканию подходящего решения вы знаете у противоречивых критериев?
5. Какое множество называется множеством Парето?

Практическое занятие «Задача Коши для уравнения теплопроводности»

Цель работы: определить оптимальное решение однокритериальных и многокритериальных задач в простейших случаях.

Краткая теория

Многие задачи науки и техники сводятся к решению обыкновенных дифференциальных уравнений (ОДУ). ОДУ называются такие уравнения, которые содержат одну или несколько производных от искомой функции. В общем виде ОДУ можно записать следующим образом:

$F(x, y, y', y'', \dots, y^{(n)}) = 0$, где x - независимая переменная, $y^{(i)}$ - i -ая производная от искомой функции. n - порядок уравнения. Общее решение ОДУ n -го порядка содержит n произвольных постоянных c_1, \dots, c_n т.е. общее решение имеет вид $y = \varphi(x, c_1, \dots, c_n)$

Для выделения единственного решения необходимо задать n дополнительных условий. В зависимости от способа задания дополнительных условий существуют два различных типа задач: задача Коши и краевая задача. Если дополнительные условия задаются в одной точке, то такая задача называется задачей Коши. Дополнительные условия в задаче Коши называются начальными условиями. Если же дополнительные условия задаются в более чем одной точке, т.е. при различных значениях независимой переменной, то такая задача называется краевой. Сами дополнительные условия называются краевыми или граничными.

Ясно, что при $n=1$ можно говорить только о задаче Коши.

Примеры постановки задачи Коши:

$$\frac{dy}{dx} = x^2 y^3, \quad y(1) = 1$$

$$\frac{d^2 y}{dx^2} = \frac{dy}{dx} + xy^2, \quad y(1) = 1, y'(1) = 0$$

Примеры краевых задач:

$$\frac{d^2 y}{dx^2} + 2 \frac{dy}{dx} - y = \sin(x), \quad y(0) = 1, y(1) = 0$$

$$\frac{d^3 y}{dx^3} = x + x \frac{d^2 y}{dx^2} - \frac{dy}{dx}, \quad y(1) = 0, y'(1) = 0, y(3) = 2$$

Решить такие задачи аналитически удастся лишь для некоторых специальных типов уравнений.

Единственность решения задачи Коши для уравнения теплопроводности в классе ограниченных функций.

Теорема 1.

Пусть D – ограниченная область в R_1 . Если решение $u(x, t)$ смешанной задачи для уравнения теплопроводности

$$u_t(x, t) - a^2(x, t) \Delta_x u(x, t) = f(x, t), \quad (x, t) \in G = D \times (0; T)$$

с начальными условиями

$$u(x, 0) = u_0(x), \quad u_0 \in C(\overline{D})$$

с граничными условиями первого рода

$$u(x, t)|_{x \in \partial D} = v(x, t), \quad v(x, t) \in C(\partial D \times [0; T])$$

существует в классе функций $C^{2,1}_x, t(G) \cap C(\overline{G})$, то оно единственно в этом классе и непрерывно зависит от начальных и граничных данных (в равномерной метрике).

Доказательство теоремы 1. Единственность. Пусть u^- и u^+ - решение задачи. Тогда их разность $u^- - u^+$ удовлетворяет однородному уравнению теплопроводности с однородными начальными

ными и граничными условиями:

$$u_t(x,t) = a^2(x,t) \Delta x u(x,t), \quad (x,t) \in G,$$

$$u(x,0) = 0,$$

$$u(x,t)|_{x \in \partial D} = 0.$$

Согласно принципу максимума в ограниченной области выполняются неравенства $0 \leq u(x,t) \leq 0$ $(x,t) \in G^-$

Следовательно, $u = u^*$ в G^- .

Непрерывная зависимость. Пусть теперь u^* и u^* - решение задачи Коши отвечающие различным начально-краевым данным: u_0^*, u_0^* и v^*, v^* соответственно.

Тогда разность $u = u^* - u^*$ является решением смешанной задачи для однородного уравнения теплопроводности

$$u_t(x,t) = a^2(x,t) \Delta x u(x,t), \quad (x,t) \in G \text{ с начальными условиями } u(x,0) = u_0^* - u_0^*$$

и граничными условиями

$$u(x,t)|_{x \in \partial D} = v^* - v^*.$$

Воспользуемся для функции u принципом максимума, получаем оценку

$$|u^*(x,t) - u^*(x,t)| \leq \max \{ \sup_{D^-} |u_0^* - u_0^*|, \sup_{\partial D \times [0;T]} |v^* - v^*| \}, \quad (x,t) \in G^-,$$

что означает непрерывную зависимость решения от начальных и краевых данных в равномерной метрике. Из принципа единственности максимума в неограниченной области вытекает следующее

Теорема 2.

Если решение задачи Коши $u_t(x,t) - a^2(x,t) \Delta x u(x,t) = f(x,t)$, $(x,t) \in G = R^n \times (0;T)$, $u(x,0) = u_0(x)$, $u_0 \in C(R^n)$

С ограниченными начальными данными u_0 существует в классе функций $C_{2,1,x,t}(G) \cap C(G^-) \cap B(G^-)$, то оно единственно в нем и непрерывно зависит от начальных данных.

Практическое занятие «Сведение произвольной задачи линейного программирования к основной задаче линейного программирования»

Цель работы: Научиться сводить произвольную задачу линейного программирования к основной задаче линейного программирования. Решить задачу линейного программирования симплекс-методом.

Краткая теория

Задачи оптимального планирования, связанные с отысканием оптимума заданной целевой функции (линейной формы) при наличии ограничений в виде линейных уравнений или линейных неравенств относятся к задачам линейного программирования.

Линейное программирование - это направление математического программирования, изучающее методы решения экстремальных задач, которые характеризуются линейной зависимостью между переменными и линейным критерием.

Сущность линейного программирования состоит в нахождении точек наибольшего или наименьшего значения некоторой функции при определенном наборе ограничений, налагаемых на аргументы и образующих **систему ограничений**, которая имеет, как правило, бесконечное множество решений. Каждая совокупность значений переменных (аргументов функции **F**), которые удовлетворяют системе ограничений, называется **допустимым планом** задачи линейного программирования. Функция **F**, максимум или минимум которой определяется, называется **целевой функцией** задачи. Допустимый план, на котором достигается максимум или минимум функции **F**, называется **оптимальным планом** задачи.

Система ограничений, определяющая множество планов, диктуется условиями производства. Задачей линейного программирования (**ЗЛП**) является выбор из множества допустимых планов наиболее выгодного (оптимального).

Общая форма задачи линейного программирования формулируют следующим образом:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \{ \leq, \geq \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \{ \leq, \geq \\ \dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \{ \leq, \geq \end{cases}$$

$$x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0$$

$$F = c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow$$

Коэффициенты $a_{ij}, b_i, c_j, j = 1, 2, \dots, n, i = 1, 2, \dots, m$ – любые действительные числа (возможно 0).

Итак, решения, удовлетворяющие системе ограничений (1) условий задачи и требованиям неотрицательности (2), называются **допустимыми**, а решения, удовлетворяющие одновременно и требованиям минимизации (максимизации) (3) целевой функции, - **оптимальными**.

Выше описанная задача линейного программирования (ЗЛП) представлена в общей форме, но одна и та же (ЗЛП) может быть сформулирована в различных эквивалентных формах. Наиболее важными формами задачи линейного программирования являются **каноническая** и **стандартная**.

В **канонической форме** задача является задачей на максимум (минимум) некоторой линейной функции F , ее система ограничений состоит только из равенств (уравнений). При этом переменные задачи x_1, x_2, \dots, x_n являются неотрицательными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \end{cases}$$

$$x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0$$

$$F = c_1x_1 + c_2x_2 + \dots + c_nx_n \rightarrow \max(n)$$

К канонической форме можно преобразовать любую задачу линейного программирования.

Правило приведения ЗЛП к каноническому виду:

1. Если в исходной задаче некоторое ограничение (например, первое) было неравенством, то оно преобразуется в равенство, введением в левую часть некоторой неотрицательной переменной, при чем в неравенства « \leq » вводится дополнительная неотрицательная переменная со знаком «+»; в случае неравенства « \geq » - со знаком «-»

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + x_{n+1} = b_1 \quad (7)$$

Вводим переменную $x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 - \dots - a_{1n}x_n$.

Тогда неравенство (7) запишется в виде:

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n$$

В каждое из неравенств вводится своя “уравнивающая” переменная, после чего система ограничений становится системой уравнений.

2. Если в исходной задаче некоторая переменная не подчинена условию неотрицательности, то ее заменяют (в целевой функции и во всех ограничениях) разностью неотрицательных

переменных

$$x_k = x_k - \dots, \quad l - \text{свободный ин-}$$
$$x_k \geq 0, \quad x_{\text{декс}}$$

3. Если в ограничениях правая часть отрицательна, то следует умножить это ограничение на (-1)

4. Наконец, если исходная задача была задачей на минимум, то введением новой целевой функции $F1 = -F$ мы преобразуем нашу задачу на минимум функции F в задачу на максимум функции $F1$.

Таким образом, **всякую задачу линейного программирования можно сформулировать в канонической форме.**

В стандартной форме задача линейного программирования является задачей на максимум (минимум) линейной целевой функции. Система ограничений ее состоит из одних линейных неравенств типа « \leq » или « \geq ». Все переменные задачи неотрицательны.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2 \\ \dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m \end{cases}$$

$$F = c_1x_1 + c_2x_2 + \dots + c_nx_n -$$

$$x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0$$

Всякую задачу линейного программирования можно сформулировать в **стандартной форме**. Преобразование задачи на минимум в задачу на максимум, а также обеспечение неотрицательности переменных производится так же, как и раньше. Всякое равенство в системе ограничений равносильно системе взаимнопротивоположных неравенств:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i \Leftrightarrow$$
$$\Leftrightarrow \begin{cases} a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n \leq b_i, \\ -a_{i1}x_1 - a_{i2}x_2 - \dots - a_{in}x_n \leq b_i, \end{cases}$$

Существует и другие способы преобразования системы равенств в систему неравенств, т.е. **всякую задачу линейного программирования можно сформулировать в стандартной форме.**

Решение задач линейного программирования симплекс-методом.

Идея симплекс-метода заключается в последовательном улучшении первоначального плана путем упорядоченного перехода от одного опорного плана к другому и завершается нахождением оптимального плана. Симплекс-методом решаются только канонические задачи линейного программирования. Решение канонической задачи симплекс-методом существенно облегчается применением так называемых симплексных таблиц. Всякую каноническую задачу можно записать условно в виде таблицы. Таблица заполняется следующим образом: первые m строк содержат в условной форме уравнения системы ограничений, разрешенные относительно базисных переменных. В последней строке записана целевая функция, эта строка называется F-строкой. В столбцах записаны свободные переменные и свободные члены.

Условие оптимальности плана: если ЗЛП на максимум, то в F-строке не должно быть отрицательных элементов; если ЗЛП на минимум, то в F-строке не должно быть положительных элементов.

Алгоритм решения:

1. Исходную задачу линейного программирования приводим к каноническому виду путем введения базисных переменных.
2. Базисные переменные выражаем через свободные переменные.

3. Строим начальный план, полагая свободные переменные равными нулю, тогда базисные переменные будут равны свободным членам.
4. Строим первую симплекс-таблицу.
5. Проверяем план на оптимальность. Если план не оптимален, то его улучшаем.
6. Улучшение плана:
 - а) Выбор разрешающего столбца
 - б) Выбор разрешающей строки
 - в) Выбор разрешающего элемента
 - г) Замена переменные в базисе
 - д) Пересчёт элементов в новой симплекс-таблице
7. Вновь полученный план проверяем на оптимальность.

Задание 1. Для производства двух видов, изделия P_1 и P_2 используется, три вида сырья S_1, S_2, S_3 , запасы которого соответственно равны 100, 60, 180 единиц. Для производства одной единицы продукции P_1 используется 2 единицы сырья S_1 и по 1 единице сырья S_2 и S_3 . Для производства одной единицы продукции P_2 используется по 1 единице сырья S_1 и S_2 и 4 единицы сырья S_3 . Прибыль от реализации 1 единицы каждой продукции P_1 и P_2 соответственно равна 30 и 20 единиц. Необходимо составить симплекс-методом такой план выпуска продукции P_1 и P_2 , при котором суммарная прибыль будет наибольшей.

Задача 2. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется a_1 кг первого сорта, a_2 кг второго сорта и a_3 кг третьего сорта. На изготовление продукции вида В расходуется b_1 кг первого сорта, b_2 кг второго сорта, b_3 кг третьего сорта. На складе фабрики имеется всего материала первого сорта c_1 кг, второго сорта c_2 кг, третьего сорта c_3 кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида α руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида β руб. Определить максимальную прибыль от реализации всей продукции видов А и В симплекс-методом.

$a_1=19, a_2=16, a_3=19, b_1=31, b_2=9, b_3=1, c_1=1121, c_2=706, c_3=1066, \alpha=16, \beta=19$.

Задача 3. а) Привести к канонической форме задачу линейного программирования.

$$\begin{cases} 2x_1 - x_2 + 6x_3 \leq 12, \\ 3x_1 + 5x_2 - 12x_3 = 14 \\ -3x_1 + 6x_2 + 4x_3 \leq 18 \end{cases}$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$

$$F = -2x_1 - x_2 + x_3 \rightarrow \min$$

б) Напишите задачу в стандартной форме.

Контрольные вопросы

1. Какие задачи можно отнести к задачам линейного программирования?
2. Какова основная идея линейного программирования?
3. Что образует систем ограничений?
4. Что называется допустимым планом?
5. Что называется целевой функцией?
6. Как записывается общая форма задачи линейного программирования?

7. Как строится каноническая форма ЗЛП?
8. Как перевести ЗЛП в стандартную форму?
9. Какова идея симплекс-метода?
10. В чем суть условия оптимальности плана?
11. Из каких пунктов состоит алгоритм решения ЗЛП симплекс-методом?
12. Что такое симплекс-отношение?

Практическое занятие «Решение задач линейного программирования симплекс-методом»

Цели работы: научиться решать задачи геометрическим методом, научиться решать задачи симплексным методом, закрепить навыки записи взаимосвязи показателей задачи в виде математической модели.

Краткая теория

Линейное программирование - это направление математического программирования, изучающее методы решения экстремальных задач, которые характеризуются линейной зависимостью между переменными и линейным критерием.

Линейное программирование состоит в нахождении экстремального значения линейной функции многих переменных при наличии линейных ограничений, связывающих эти переменные.

Необходимым условием постановки задачи линейного программирования являются ограничения на наличие ресурсов, величину спроса, производственную мощность предприятия и другие производственные факторы.

Математическая модель любой задачи линейного программирования включает в себя:

- максимум или минимум целевой функции (критерий оптимальности);
- систему ограничений в форме линейных уравнений и неравенств;
- требование неотрицательности переменных.

Стандартной задачей линейного программирования называется задача, которая состоит в определении максимального (минимального) значения целевой функции (1) при выполнении условий (2) и (4).

Канонической (или основной) задачей линейного программирования называется задача, которая состоит в определении максимального (минимального) значения целевой функции (1) при выполнении условий (3) и (4).

Графический метод решения двумерной задачи линейного программирования (максимизация целевой функции)

Двумерная задача линейного программирования - задача линейного программирования, количество переменных которой равно 2.

В общем виде двумерную задачу линейного программирования можно представить следующим образом.

Определить значение переменных x_1 и x_2 , при которых линейная целевая функция F достигает максимума (минимума).

$F = c_1x_1 + c_2x_2 \rightarrow \max(\min)$ при ограничениях на переменные

$$\begin{cases} a_{11}x_1 + a_{12}x_2 \leq (=, \geq) b_1 \\ a_{21}x_1 + a_{22}x_2 \leq (=, \geq) b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 \leq (=, \geq) b_m \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

Среди ограничений могут одновременно встречаться знаки \geq , \leq и $=$. Коэффициенты a_{ij} , b_i , c_j ($i = 1..m, j = 1, 2$) - любые действительные числа (возможно и 0).

Двумерные задачи линейного программирования обычно решаются графически и решение

связано со свойствами выпуклых множеств.

Множество точек называется выпуклым, если оно вместе с любыми двумя точками содержит и их произвольную выпуклую комбинацию.

Геометрический смысл этого определения состоит в том, что множеству вместе с его произвольными точками полностью принадлежит и прямолинейный отрезок, их соединяющий. Примерами выпуклых множеств являются прямолинейный отрезок, полуплоскость, круг, шар, куб, полупространство и др.

Множество планов основной задачи линейного программирования является выпуклым (если оно не пусто). Непустое множество планов называется многогранником решений, а всякая угловая точка многогранника решений - вершиной.

Если основная задача линейного программирования имеет оптимальный план, то целевая функция задачи принимает максимальное значение в одной из вершин многогранника решений. Если максимальное значение достигается более чем в одной вершине, то целевая функция принимает его во всякой точке, являющейся выпуклой линейной комбинацией этих вершин.

Алгоритм решения двумерной задачи линейного программирования графическим методом

Решение задачи линейного программирования графическим методом включает следующие этапы.

1. На плоскости $X_1O X_2$ строят прямые, уравнения которых получаются в результате замены в ограничениях знаков неравенств на знаки точных равенств.
2. Находят полуплоскости, определяемые каждым из ограничений задачи.
3. Строят многоугольник решений.
4. Строят вектор $N(c_1, c_2)$, который указывает направление возрастания целевой функции.
5. Строят начальную прямую целевой функции $c_1x_1 + c_2x_2 = 0$ и затем передвигают ее в направлении вектора N до крайней угловой точки многоугольника решений. В результате находят точку, в которой целевая функция принимает максимальное значение, либо множество точек с одинаковым максимальным значением целевой функции, если начальная прямая сливается с одной из сторон многоугольника решений, либо устанавливают неограниченность сверху функции на множестве планов.
6. Определяют координаты точки максимум функции и вычисляют значение целевой функции в этой точке.

Минимальное значение линейной функции цели находится путем передвижения начальной прямой $c_1x_1 + c_2x_2 = 0$ в направлении, противоположном вектору $N(c_1, c_2)$.

Замечание 1: В алгоритме решения пункты 4-6 можно выполнять следующим образом:

Найти значение целевой функции в угловых точках многогранника решений. Точка, в которой функция принимает наибольшее значение и является точкой максимума.

Симплекс-метод метода решения задачи линейного программирования

Для решения задач линейного программирования предложено немало различных алгоритмов. Наиболее эффективным среди них является алгоритм, известный под названием **симплексный метод, или метод последовательного улучшения плана**.

Впервые симплексный метод был предложен американским ученым Дж. Данцингом в 1949 г., однако еще в 1939 г. идеи метода были разработаны российским математиком Л.В. Канторовичем.

Симплексный метод - это итерационный процесс, который начинается с одного решения и в поисках лучшего варианта движется по угловым точкам области возможных решений до тех пор, пока не достигнет оптимального значения, в частности по угловым точкам многоугольника решений, полученного геометрическим методом.

Симплексный метод основан на последовательном переходе от одного опорного плана задачи линейного программирования к другому, при этом значение целевой функции изменяется.

Алгоритм симплексного метода включает следующие этапы:

1. Составление первого опорного плана. Система ограничений задачи, решаемой симплексным методом, задана в виде системы неравенств смысла «<», правые части которых $b_i > 0$. Перейдем от системы неравенств к системе уравнений путем введения неотрицательных дополнительных переменных. Векторы-столбцы при этих переменных представляют собой единичные векторы и образуют базис, а соответствующие им переменные называются базисными:

$$\sum_{j=1}^n a_{ij} \cdot x_j + x_{n+i} = b_i, \quad i = \overline{1, m},$$

где x_{n+i} – базисные переменные, $i = \overline{1, m}$,
 x_j – свободные переменные, $j = \overline{1, n}$.

Решим эту систему относительно базисных переменных:

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

а функцию цели перепишем в виде уравнения

$$F(\bar{X}) = 0 - (- \sum_{j=1}^n c_j \cdot x_j).$$

Полагая, что основные переменные $x_1 = \dots = x_n = 0$, получим первый опорный план $X_1 = (0, 0, \dots, 0, b_1, b_2, \dots, b_m)$; $F(X_1) = 0$, который заносим в симплексную табл. Она состоит из коэффициентов системы ограничений и свободных членов. Последняя строка таблицы называется индексной и заполняется коэффициентами функции цели, взятыми с противоположным знаком.

2. Проверка плана на оптимальность. Если все коэффициенты индексной строки симплексной таблицы при решении задачи на максимум неотрицательны (> 0), то план является оптимальным. Если найдется хотя бы один коэффициент индексной строки меньше нуля, то план не оптимальный и его можно улучшить. В этом случае переходим к следующему этапу алгоритма.

3. Определение ведущих столбца и строки. Из отрицательных коэффициентов индексной строки выбираем наибольший по абсолютной величине, что и определяет ведущий столбец, который показывает, какая переменная на следующей итерации перейдет из свободных в базисные. Затем элементы столбца свободных членов симплексной таблицы делим на элементы того же знака ($+/+$; $-/-$) ведущего столбца. Результаты заносим в отдельный столбец d_i , которые будут всегда положительными. Строка симплексной таблицы, соответствующая минимальному значению d_i , является ведущей. Она определяет переменную x_i , которая на следующей итерации выйдет из базиса и станет свободной. Элемент симплексной таблицы, находящийся на пересечении ведущих столбца и строки, называют разрешающим и выделяют кружком.

4. Построение нового опорного плана. Переход к новому плану осуществляется в результате пересчета симплексной таблицы методом Жордана - Гаусса. Сначала заменим переменные в базисе, т. е. вместо x_i , в базис войдет переменная x_j , соответствующая ведущему столбцу.

Разделим все элементы ведущей строки предыдущей симплексной таблицы на разрешающий элемент и результаты деления занесем в строку следующей симплексной таблицы, соответствующую введенной в базис переменной x_j . В результате этого на месте разрешающего элемента в следующей симплексной таблице будем иметь 1, а в остальных клетках j столбца, включая клетку столбца индексной строки, записываем нули. Остальные новые элементы нового плана находятся по правилу прямоугольника: $НЭ = СТЭ \cdot А \cdot В / РЭ$, где СТЭ - элемент старого плана, РЭ - разрешающий элемент, А и В - элементы старого плана, образующие прямоугольник с элементами СТЭ и РЭ.

Далее возвращаемся ко второму этапу алгоритма — проверке плана на оптимальность.

При решении задачи линейного программирования на минимум целевой функции признаком оптимальности плана являются отрицательные значения всех коэффициентов индексной строки симплексной таблицы.

Задание 1

На предприятии имеется сырье видов I, II, III. Из него можно изготавливать изделия типов

А и В. Пусть запасы видов сырья на предприятии составляют b_1, b_2, b_3 ед. соответственно, изделие типа А дает прибыль c_1 ден.ед., а изделие типа В - c_2 ден.ед. Расход сырья на изготовление одного изделия задан в словных единицах таблицей.

Составить план выпуска изделий, при котором предприятие имеет наибольшую прибыль. Решить задачу графическим методом.

| 1 вариант | | | | | | | | |
|-----------|-------|----|-----|-----|----|-----|----|----|
| Изделие | Сырье | | | b1 | b2 | b3 | c1 | c2 |
| | I | II | III | 102 | 91 | 105 | 5 | 9 |
| | A | 6 | 3 | | | | | |
| B | 3 | 4 | 5 | | | | | |

Задание 2

Предприятие выпускает три вида изделий (N_1, N_2, N_3), используя три вида ресурсов (P_1, P_2, P_3). Запасы ресурсов (3) ограничены. Прибыль от реализации (П) единицы изделия и нормы расхода ресурсов представлены в таблице. Определить ассортимент и объемы выпуска продукции, получаемую прибыль, величину остатков. Найти решение задачи симплексным методом с представлением всех симплексных таблиц и проанализировать полученные результаты.

| Вариант 1 | | | | |
|-----------|----|----|----|----|
| | N1 | N2 | N3 | 3 |
| P1 | 1 | 8 | 5 | 43 |
| P2 | 4 | 1 | 6 | 74 |
| P3 | 5 | 2 | 2 | 35 |
| П | 5 | 7 | 6 | |

Контрольные вопросы:

1. Какие задачи относятся к задачам линейного программирования?
2. Как определяется область допустимых решений (многоугольник решений)?
3. Как строится начальный вектор и что он показывает?
4. Какие задачи линейного программирования можно решать геометрическим методом?
5. Каков признак оптимальности в симплексном методе?
6. Как строится опорный план?
7. Как определяется ведущий столбец и ведущая строка симплексной таблицы?
8. Как осуществляется перерасчет элементов симплексной таблицы?
9. Оцените по рациональности метода и сложности методы решения задач ЛП.

Практическое занятие «Нахождение начального решения транспортной задачи.

Решение транспортной задачи методом потенциалов»

Цель работы: Найти начальное решение транспортной задачи двумя методами: методом северо-западного угла и методом наименьшей стоимости. Найти оптимальное решение транспортной задачи методом потенциалов.

Краткая теория

Симплексный метод для решения задач линейного программирования является универсальным, он позволяет решить любую задачу, но решение иных задач связано с трудоемкими расчетами. Можно выделить класс задач, которые решаются более простыми специальными методами. К числу таких задач относятся так называемые **транспортные задачи**.

Классическая транспортная задача - о наиболее экономном плане перевозок однородного продукта или взаимозаменяемых продуктов из пунктов отправления в пункты назначения.

Классическая транспортная задача (сокращенно ТЗ) формулируется следующим образом.

В пунктах отправления A_1, A_2, \dots, A_m , которые будем называть также поставщиками,

сосредоточены запасы однородного груза в количествах a_1, a_2, \dots, a_m соответственно.

В пункты назначения B_1, B_2, \dots, B_n , именуемые потребителями, надлежит доставить соответственно b_1, b_2, \dots, b_n единиц груза. Известен транспортный тариф c_{ij} - стоимость перевозки единицы груза из пункта A_i в пункт B_j , $i = 1, 2, \dots, m, j = 1, 2, \dots, n$. Требуется составить такой план перевозок груза, при котором общая стоимость F всех перевозок была бы наименьшей, при этом все заявки были бы выполнены.

В термин "транспортный тариф" вкладывается условное понимание стоимости единицы груза - это может быть себестоимость, расстояние, тариф, время, расход топлива или электроэнергии и др.

Пусть суммарные запасы грузов у поставщиков равны суммарным потребностям потребителей. Это условие называется условием баланса. Если для ТЗ условие баланса выполняется, то модель ТЗ называется закрытой, если условие баланса не выполнено, то модель ТЗ - открытая.

Составим математическую модель ТЗ. Пусть x_{ij} - количество груза, которое поставщик A_i отправляет потребителю B_j ($i = 1, 2, \dots, m, j = 1, 2, \dots, n$) со стоимостью перевозок c_{ij} . Данные задачи можно представить в виде таблицы 1.

Таблица 1.

| Поставщики | Потребители | | | | Запасы |
|-------------|----------------|----------------|-----|----------------|-----------------------|
| | B_1 | B_2 | ... | B_n | |
| A_1 | $c_{11}x_{11}$ | $c_{12}x_{12}$ | ... | $c_{1n}x_{1n}$ | a_1 |
| A_2 | $c_{21}x_{21}$ | $c_{22}x_{22}$ | ... | $c_{2n}x_{2n}$ | a_2 |
| ... | ... | ... | ... | ... | ... |
| A_m | $c_{m1}x_{m1}$ | $c_{m2}x_{m2}$ | ... | $c_{mn}x_{mn}$ | a_m |
| Потребности | b_1 | b_2 | ... | b_n | $\sum a_i = \sum b_j$ |

Теорема 1. Транспортная задача при выполнении условия баланса всегда имеет решение.

Теорема 2. Система ограничений транспортной задачи содержит $m+n-1$ линейно - независимых уравнений.

При решении задач практический смысл теоремы 2 заключается в следующем: число назначенных перевозок равно $m+n-1$.

Процедура решения ТЗ будет состоять в последовательном улучшении опорных планов и проверки их на оптимальность.

Методы построения начального плана.

Существует несколько методов построения первоначального опорного плана ТЗ (опорный план - план, удовлетворяющий системе ограничений и условию неотрицательности). Рассмотрим только два из них: метод северо-западного угла и метод наименьшей стоимости.

Как уже отмечалось, в опорном плане не более $r = m + n - 1$ переменных x_{ij} , отличных от нуля. Если таких переменных равно r , то такой план называют невырожденным, в про-

тивном случае - вырожденным.

Метод северо-западного угла. Назначение перевозок начинаем с левой верхней клетки (северо-западный угол). Сравнивая ресурсы поставщика и потребности потребителя, назначаем максимально возможную перевозку. Если ресурсов поставщика недостаточно, то переходим к следующему поставщику. Если ресурсов у поставщика достаточно, то назначив нужную перевозку первому потребителю, переходим к следующему потребителю. При назначении перевозок для удобства записываем остаток ресурсов (потребностей); если ресурсы закончились или потребности удовлетворены, то ставим букву "к" (конец). Если при назначении перевозки одновременно закончились запасы ресурсов у поставщика и удовлетворены потребности потребителя, то из "игры" выводим только одного участника, другому оставляем нуль запасов или нуль потребностей.

Метод наименьшей стоимости. Выбираем клетку с наименьшей тарифной ставкой и назначаем максимально возможную перевозку. Если запасы закончились или потребности удовлетворены, то поставщика или потребителя исключаем. Среди оставшихся клеток снова выбираем клетку с наименьшей стоимостью и назначаем максимально возможную перевозку. Если в результате назначения перевозки закончились запасы поставщика или удовлетворены потребности потребителя, то его исключаем из дальнейшего рассмотрения.

Теорема 3(условие оптимальности плана). Сумма потенциалов поставщика и потребителя равна тарифной ставке для занятых клеток; сумма потенциалов поставщика и потребителя не превышает тарифную ставку для свободных клеток:

Замечание. Опорный план должен быть невырожденным.

Задание 1. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве **a1, a2 и a3** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно **b1, b2, b3, b4, b5** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

| Пунктыпоставки | Пункты потребления | | | | |
|----------------|--------------------|-----|-----|-----|-----|
| | B1 | B2 | B3 | B4 | B5 |
| A1 | D11 | D12 | D13 | D14 | D15 |
| A2 | D21 | D22 | D23 | D24 | D25 |
| A3 | D31 | D32 | D33 | D34 | D35 |

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=200, a2=250, a3=200,

b1=190,b2=100,b3=120,b4=110,b5=130.

Контрольные вопросы

1. Какие задачи называются транспортными?
2. В чем суть классической транспортной задачи?
3. Что означает термин «транспортный тариф»?
4. Как записывается условие баланса?
5. Как выглядит математическая постановка транспортной задачи?
6. В чем суть метода северо-западного угла?
7. Основная идея метода наименьшей стоимости?
8. В чем суть метода потенциалов?
9. Какие клетки называются потенциальными?
10. Какие виды контуров вы знаете?

Практическое занятие «Применение метода стрельбы для решения линейной краевой задачи»

Цель работы: научиться применять метод стрельбы для решения линейной краевой задачи.

Пример краевой задачи

Примером двухточечной краевой задачи является задача:

$$y'' = f(x, y, y'), \quad 0 < x < 1,$$

$$y(0) = Y_0, \quad y(1) = Y_1.$$

с граничными условиями на обоих концах отрезка $0 \leq x \leq 1$, на котором надо найти решение $y = y(x)$. На этом примере мы схематически изложим некоторые способы численного решения краевых задач.

Если функция $f(x, y, y')$ линейна по аргументам y и y' , то мы имеем линейную краевую задачу, иначе - нелинейную краевую задачу.

Линейная краевая задача

Рассмотрим частную, но довольно распространенную краевую задачу следующего вида:

$$Ly = y'' - p(x)y = f(x), \quad 0 < x < 1,$$

$$y(0) = Y_0, \quad y(1) = Y_1.$$

Для этой задачи проиллюстрируем два способа решения: один основан на идее численного построения общего решения линейного дифференциального уравнения, другой сводит исходную дифференциальную краевую задачу к системе линейных алгебраических уравнений, решение которой находится методом прогонки.

Метод численного построения общего решения

Для нахождения решения краевой задачи можно численно построить решение дифференциального уравнения, представимое в виде

$$y(x) = C_1 y_1(x) + C_2 y_2(x) + y_0(x),$$

где $y_0(x)$ какое-либо решение неоднородного уравнения

$$y'' - p(x)y = f(x),$$

а $y_1(x)$ и $y_2(x)$ два любые линейно независимые решения однородного уравнения $y'' - p(x)y = 0$.

Постоянные C_1 и C_2 находятся из граничных условий задачи.

Так как решения произвольны, то их можно построить различными способами. Например, можно $y_0(x)$, $y_1(x)$, $y_2(x)$ какие-то начальные условия и решить одну задачу Коши для неоднородного и две задачи Коши для однородного уравнений. Эти условия, в частности, могут быть такими:

$$y_0(0) = 0, \quad y_0'(0) = 0 \quad \text{для неоднородного уравнения;}$$

$$\begin{aligned} y_1(0) &= 1, & y_1'(0) &= 0; \\ y_2(0) &= 0, & y_2'(0) &= 1 \end{aligned} \quad \text{для однородного уравнения.}$$

Однако при реализации этого способа, например, в случае $p(x) \gg 1$ для рассматриваемого уравнения могут возникнуть трудности, связанные с неустойчивостью задачи Коши. В этом случае можно попытаться построить $y_0(x)$, $y_1(x)$, $y_2(x)$ с помощью решения одной краевой задачи для неоднородного уравнения и двух краевых задач для однородного уравнения. Краевые условия для этих задач могут быть, например, следующими:

$$y_0(0) = 0, \quad y_0'(1) = 0 \quad \text{для неоднородного уравнения;}$$

$$\begin{aligned} y_1(0) &= 1, & y_1'(1) &= 0; \\ y_2(0) &= 0, & y_2'(1) &= 1 \end{aligned} \quad \text{для однородного уравнения.}$$

Эти задачи могут быть решены методом прогонки. Условия устойчивости метода прогонки при $p(x) > 0$, как легко проверить, выполнены. Этот подход может оказаться полезным, если краевые условия таковы, что для исходной задачи метод прогонки применен быть не может.

Отметим, что с учетом специфики краевых условий исходной задачи можно строить общее решение вида

$$y(x) = y_0(x) + C y_1(x),$$

где $y_0(x)$ некоторое решение неоднородного уравнения, а $y_1(x)$ некоторое решение однородного уравнения.

Конечно-разностный метод (метод прогонки)

При нахождении решения линейной краевой задачи:

$$y'' - p(x)y = f(x), \quad 0 < x < 1,$$

$$y(0) = Y_0, \quad y(1) = Y_1.$$

Для $p(x) \gg 1$ методом построения общего решения, если оно находится с помощью решения задач Коши, могут возникнуть трудности, связанные с вычислительной неустойчивостью задачи

Коши.

Для решения поставленной задачи можно воспользоваться разностной схемой:

$$\frac{y_{m+1} - 2y_m + y_{m-1}}{h^2} - p(x_m) y_m = f(x_m),$$

$$0 < m < M, \quad Mh=1, \quad y_0 = Y_0, \quad y_M = Y_1$$

и решить разностную задачу методом прогонки. Условия применимости метода прогонки при $p(x) > 0$, как легко проверить, выполнены.

Нелинейная краевая задача

Краевая задача

$$y'' = f(x, y, y'), \quad 0 < x < 1,$$

$$y(0) = Y_0, \quad y(1) = Y_1.$$

является нелинейной краевой задачей, если функция $f(x, y, y')$ нелинейна хотя бы по одному из аргументов y или y' .

В настоящей работе реализованы два способа решения нелинейных краевых задач: *метод стрельбы* и *метод линеаризации* (метод Ньютона), который сводит решение нелинейной краевой задачи к решению серии линейных краевых задач.

Метод стрельбы

Метод стрельбы для решения краевой задачи базируется на том, что имеются удобные способы численного решения задачи Коши, т. е. задачи следующего вида

$$y'' = f(x, y, y'), \quad 0 < x < 1,$$

$$y(0) = Y_0,$$

$$y'(0) = \tan \alpha,$$

где Y_0 ордината точки $(0, Y_0)$, из которой выходит интегральная кривая; угол наклона интегральной кривой к оси x при выходе из точки $(0, Y_0)$. При фиксированном Y_0 решение задачи имеет вид $y(x, \alpha)$. При $x=1$ решение $y = y(x, \alpha)$ зависит только от $y(x, \alpha)|_{x=1} = y(1, \alpha)$.

Используя указанное замечание о решении задачи Коши, можно задачу переформулировать следующим образом:

Найти такой угол $\alpha = \alpha^*$, при котором интегральная кривая, выходящая из точки $(0, Y_0)$ под углом α к оси абсцисс, попадет в точку $(1, Y_1)$:

$$y(1, \alpha) = Y_1.$$

Решение задачи при этом $\alpha = \alpha^*$ совпадает с искомым решением задачи. Таким образом, дело сводится к решению уравнения (рис. 9).

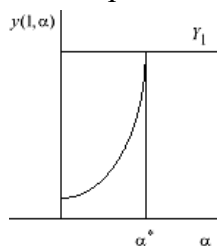


Рис. 9

Задание 1

Начните выполнение работы с темы «Линейная краевая задача». Выбрав с помощью меню один из методов решения линейной краевой задачи, перейдите к пункту меню «Параметры». Наберите следующую краевую задачу:

$$y'' - py = -p, \quad 0 < x < 1,$$

$$y(0) = 1, \quad y(1) = 1.$$

Для $p = \text{const} > 0$. Решением этой задачи является функция $y \equiv 1$. Установите значение шага сетки $h = 0,05$.

Найдите решение этой задачи методом построения общего решения и методом прогонки для разных p , начиная с умеренных значений и увеличивая их до величины порядка 1200.

Сравните получаемые решения с точным и объясните наблюдаемые эффекты. Попытайтесь найти решение этой же задачи методом стрельбы. Проанализируйте, как влияет при разных p точность задания недостающего начального условия на левом конце интервала на успешное решение задачи методом стрельбы.

Объясните полученные результаты. Замените левое краевое условие (положите, например, $y'(0) = 0$), и посмотрите, как изменится характер решения.

Практическое занятие «Задача о распределении средств между предприятиями»

Цель работы: Решить простейшие задачи методом динамического программирования.

Краткая теория

Динамическое программирование – метод оптимизации, приспособленный, к задачам, в которых процесс принятия решения может быть разбит на отдельные этапы (шаги). Такие задачи называются многошаговыми.

Характерные особенности задач динамического программирования:

- 1) Неоднозначность решения.
- 2) Возможность деления вычислительного процесса на этапы.
- 3) Общий критерий - сумма частных критериев на этапах.

Динамическое программирование позволяет осуществлять оптимальное планирование многошаговых процессов, зависящих от времени. Процесс называется управляемым, если можно влиять на ход его развития. Управлением называется совокупность решений, принимаемых на каждом этапе для влияния на ход процесса. Началом этапа (шага) управляемого процесса считается момент принятия решения. Планируя многошаговый процесс, исходят из интересов всего процесса в целом, всегда необходимо иметь в виду конечную цель.

Метод динамического программирования состоит в том, что оптимальное управление строится постепенно. На каждом этапе оптимизируется управление только этого этапа, причем управление выбирается с учётом последствий, т.е. оптимальное управление для данного этапа должно учитывать весь последующий ход процесса, для чего необходимо знать все управления на последующих этапах. Поскольку процесс заканчивается на последнем этапе, оптимальное решение не должно учитывать последующего управления. Таким образом, процесс вычисления протекает в обратном направлении, от конца к началу.

Постановка задачи динамического программирования.

Пусть S_0, S_k, S_n - состояния системы на начальном, k -ом и конечном этапе, u_0, u_k, u_n управления системой на начальном, k -ом и конечном этапах. Управление u_k переводит систему из состояния S_{k-1} в состояние S_k . Показатель эффективности на k -ом этапе обозначим через $W_k(S_{k-1})$.

Принцип оптимальности Беллмана можно сформулировать следующим образом: каковы бы не были начальное состояние и начальное решение, последующее решение должно быть оптимальным по отношению к состоянию, полученному в результате начального решения. Иными словами, принцип оптимальности утверждает, что если в данный момент выбрано не наилучшее решение, то последствия этого нельзя исправить в будущем.

Задача определения кратчайших расстояний по заданной сети

Пусть дано конечное число точек P_1, P_2, \dots, P_n , соединенных всевозможными отрезками линий, называемых звеньями или связями. Тогда совокупность точек и их связей называют сетью. Сеть называется достаточно связанной, если существует путь, состоящий из звеньев и соединяющий любые две точки сети. Пусть каждому звену поставлено в соответствие действительное

неотрицательное число. l_{ij} - его длина. Необходимо определить кратчайшее расстояние по сети от каждой точки до всех остальных и соответствующие пути, по которым, они проходят. Пронумеруем точки сети в любом порядке и укажем длину каждого звена. Две точки называются соседними, если они непосредственно соединены связью. Положим, $l_{ji} = l_{ij}$. Для решения задачи используем метод динамического программирования и отыскиваем кратчайшее расстояние не от фиксированной точки до всех остальных, а от всех остальных до фиксированной через соседние точки. Связь, через которую проходит кратчайшее расстояние, после каждого шага отмечаем стрелкой. Для удобства точки сети обозначим кружками с номерами точек.

Задача 1

Двум предприятиям А и В на 4 квартала выделено $S_0 = 1000$ единиц средств. Каждый квартал предприятие А получает x средств, предприятие В - y средств. При этом от выделенных средств предприятие А получает $5x$ единиц и остаток средств $0,3x$ единиц, а предприятие В - доход $4y$ единиц и остаток выделенных средств $0,5y$ единиц. Необходимо распределить средства между предприятиями поквартально таким образом, чтобы за весь год оба предприятия получили максимальный доход.

Задача 2

Дана сеть, состоящая из 7 точек, и известны расстояния между точками. Необходимо определить кратчайшее расстояние от любой точки до точки 7.

Контрольные вопросы

1. Что называется динамическим программированием?
2. Какие характерные особенности задач динамического программирования узнаете?
3. Что называется управлением?
4. В чем состоит метод динамического программирования?
5. Сформулируйте принцип оптимальности Беллмана?
6. Что называется сетью, звеньями?
7. Что такое характеристика точки?

Опишите алгоритм решения задачи определения кратчайшего расстояния по заданной сети?

Практическое занятие «Задача о замене оборудования»

Цели работы: научиться решать задачи динамического программирования, научиться разбивать весь процесс решения задачи на этапы, научиться выбирать оптимальную стратегию поведения.

Краткая теория

1. Понятие задачи динамического программирования

Рассматриваемые ранее задачи характеризуются тем, что в них не учитываются изменения оптимизируемых параметров во времени – процессы считаются статичными. Выбирается некоторый период времени, и для него определяются проектируемые или планируемые значения показателей. При этом предполагается, что управляемые или неуправляемые параметры системы в течение всего планового времени не будут изменяться или, по крайней мере, не претерпят серьезных изменений, требующих пересмотра принятых решений.

Однако в реальной жизни есть задачи, в которых необходимо учитывать изменения параметров систем во времени. Эти параметры могут меняться непрерывно или дискретно – от этапа к этапу. Например, из года в год меняется возраст машин и оборудования, изменяется производственная мощность и производительность труда на предприятиях. Очевидно, что необходимо принимать оптимальные решения на год (или другой срок) и одновременно на весь рассматриваемый период в целом с учётом возможных изменений параметров. Для решения такого вида задач, которые получили название **многошаговые**, разработан соответствующий математический аппа-

рат, который получил название **динамическое программирование**.

Задача может быть сформулирована следующим образом: Определить u_i^* (u_i^* не только число, а может быть вектором, функцией) на каждом шаге, $i = 1, 2, \dots, m$, и тем самым u^* всей операции в целом.

Рассмотрим подход к решению данной задачи. Характерным для динамического программирования является то, что переменные рассматриваются вместе, а не последовательно – одна за другой. При этом вычислительная тема строится таким образом, что вместо одной задачи с n переменными решается серия задач с небольшим числом, а чаще с одной переменной. Сам же вычислительный процесс производится на основе метода последовательных приближений в два круга:

1. *От последнего шага к первому.*
2. *От первого шага к последнему или же наоборот, в зависимости от исходных данных.*

На первом круге ищется так называемое условное оптимальное решение. Оно выбирается так, чтобы все предыдущие шаги обеспечили максимальную эффективность последующего. Основу такого подхода составляет принцип оптимальности Беллмана, который формулируется следующим образом:

Нельзя получить оптимальное значение целевой функции i -шагового процесса, если для любого u_i , выбранного на шаге i , значение целевой функции для оставшихся $i-1$ шагов не является оптимальным при этом выбранном на i -шаге значении u_i .

Такой процесс продолжается до тех пор, пока решение не потеряет свой условный характер, т. е. до первого шага или последнего. Для него решение просто оптимально. Поэтому второй круг начинают именно с этого шага и последовательно переходят от условных к оптимальным решениям, тем самым обеспечивается оптимальность операции в целом.

Задание 1

Капитал 40 млн.руб. инвестор должен вложить в четыре инвестиционных проекта так, чтобы получить максимальный доход. Доходность проектов дана в таблице (вложения кратны 8 млн. руб.)

| u | Прибыль от внедрения | | | |
|----|----------------------|-----------|------------|-------|
| | f4(u) | f3(u) | f2(u) | f1(u) |
| 8 | <u>55</u> | <u>39</u> | 35 | 32 |
| 16 | 58 | 53 | 76 | 68 |
| 24 | 90 | 80 | <u>120</u> | 115 |
| 32 | 100 | 120 | 135 | 134 |
| 40 | 140 | 145 | 158 | 147 |

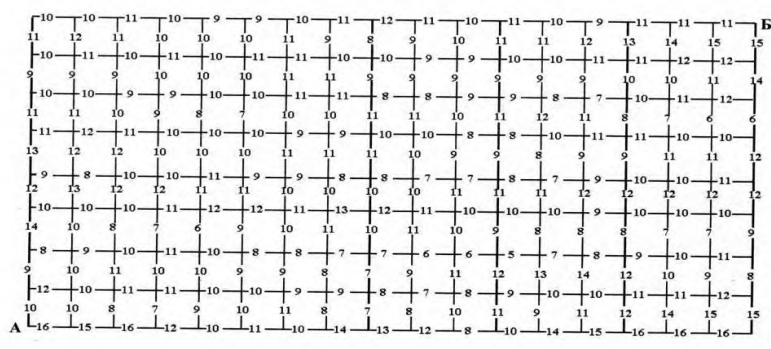
Задание 2

Распределите оптимальным образом денежные средства инвестора величиной 5 у.е. между четырьмя предприятиями. Доход каждого предприятия от вложения в него u у.е. определяется функцией дохода $f(u)$. Эти функции приведены в таблице.

| u | Прибыль от внедрения по предприятиям | | | |
|---|--------------------------------------|-------|-------|-------|
| | f4(u) | f3(u) | f2(u) | f1(u) |
| 1 | f4(1) | 6 | 3 | 4 |
| 2 | 10 | f3(2) | 4 | 6 |
| 3 | 11 | 11 | f2(3) | 8 |
| 4 | 12 | 13 | 11 | f1(4) |
| 5 | 18 | 15 | 18 | 16 |

Задание 3

Из пункта А в пункт В необходимо проложить автомобильную трассу по самому экономичному пути.



Контрольные вопросы:

1. Какие задачи можно решать методами динамического программирования?
2. В чем заключаются достоинства и недостатки динамического программирования?
3. Объясните алгоритм решения задач динамического программирования.
4. Укажите принцип выбора направления движения.
5. В чем заключается принцип оптимальности?
6. Каков алгоритм распределения ресурсов?

Практическое занятие «Нахождение кратчайших путей в графе. Решение задачи о максимальном потоке»

Цель работы: Решить задачу о нахождении кратчайших путей в графе. Решить задачу о нахождении максимального потока.

Краткая теория

Граф это множество точек или вершин и множество линий или ребер, соединяющих между собой все или часть этих точек. *Вершины*, прилегающие к одному и тому же ребру, называются *смежными*. Если *ребра* ориентированы, что обычно показывают *стрелками*, то они называются *дугами*, и граф с такими ребрами называется **ориентированным графом**. Если *ребра* не имеют *ориентации*, граф называется **неориентированным**.

Графы обычно изображаются в виде геометрических фигур, так что вершины графа изображаются точками, а ребра - линиями, соединяющими точки (рис. 1).

Петля это дуга, начальная и конечная вершина которой совпадают.

Простой граф - граф без кратных ребер и петель.

Степень вершины это удвоенное количество петель, находящихся у этой вершины плюс количество остальных прилегающих к ней ребер.

Пустым называется граф без ребер. *Полным* называется граф, в котором каждые две вершины смежные.

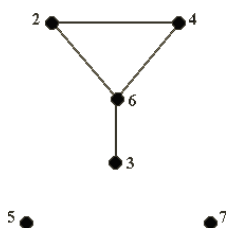


Рис. 1

Путь в ориентированном графе - это последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Маршрут в графе путь, ориентацией дуг которого можно пренебречь.

Цепь маршрут, в котором все ребра попарно различны.

Цикл замкнутый маршрут, являющийся цепью.

Маршрут, в котором все вершины попарно различны, называют **простой цепью**. Цикл, в котором все вершины, кроме первой и последней, попарно различны, называются **простым циклом**.

Подграф графа это граф, являющийся *подмоделью* исходного графа, т.е. подграф содержит некоторые вершины исходного графа и некоторые ребра (только те, оба конца которых входят в подграф).

Подграф называется **остовным** подграфом, если множество его вершин совпадает с множеством вершин самого графа.

Граф называется **связным**, если любая пара его вершин связана. *Связными компонентами* графа называются подграфы данного графа, вершины которых связаны.

Дерево - это связный граф без циклов. Деревья особенно часто возникают на практике при изображении различных иерархий. Например, популярны генеалогические деревья.

Граф без цикла называется **лесом**. Вершины *степени 1* в дереве называются **листьями**. **Деревья** - очень удобный инструмент представления информации самого разного вида. Деревья *отличаются* от простых графов тем, что **при обходе дерева невозможны циклы**. Это делает графы очень удобной формой организации данных для различных алгоритмов.

Очевидно, что графический способ представления графов непригоден для ПК. Поэтому существуют другие способы представления графов.

В теории графов применяются

1. **Матрица инцидентности**. Это матрица A с n строками, соответствующими вершинам, и m столбцами, соответствующего ребрам. Для ориентированного графа столбец, соответствующий дуге (x, y) содержит - 1 в строке, соответствующей вершине x и 1 , в строке, соответствующей вершине y . Во всех остальных 0 . Петлю, т.е. дугу (x, x) можно представлять иным значением в строке x , например, 2 . Если граф неориентированный, то столбец, соответствующий ребру (x, y) содержит 1 , соответствующие x и y и нули во всех остальных строках.

2. **Матрица смежности**. Это матрица $n \times n$ где n - число вершин, где $b_{ij} = 1$, если существует ребро, идущее из вершины x в вершину y и $b_{ij} = 0$ в противном случае.

Нахождение минимального остова в графе

Алгоритм решения

1. Упорядочить ребра графа по возрастанию весов;
2. Выбрать ребро с минимальным весом, не образующее цикл с ранее выбранными ребрами. Занести выбранное ребро в список ребер строящегося остова;
3. Проверить, все ли вершины графа вошли в построенный остов. Если нет, то выполнить пункт 2.

Нахождение кратчайшего пути в графе

Пусть дан граф, дугам которого приписаны веса. Задача о нахождении кратчайшего пути состоит в нахождении кратчайшего пути от заданной начальной вершины до заданной конечной вершины, при условии, что такой путь существует.

Данная задача может быть разбита на две:

1. для начальной заданной вершины найти все кратчайшие пути от этой вершины к другим;
2. найти кратчайшие пути между всеми парами вершин.

Рассмотрим алгоритм решения для задачи первого типа:

Необходимо найти путь от s - начальной вершины до t - конечной вершины. Каждой вершине присваиваем пометки $I(X_i)$.

1. $I(s) = 0$, $I(X_i)$ равно бесконечности для всех X_i не равных s и считать эти по-

метки временными. Положить $p = s$.

2. Для всех Xi , принадлежащих $\Gamma(p)$ и пометки которых временны, изменить пометки последующему правилу: $I(Xi) = \min[I(Xi), I(p) + c(p, Xi)]$

3. среди всех вершин с временными пометками найти такую, для которой $I(Xi^*) = \min[I(Xi)]$

4. считать пометку вершины Xi^* постоянной и положить $p = Xi^*$.

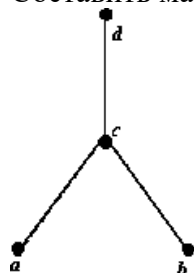
5. если $p = t$, то $I(p)$ является длиной кратчайшего пути, если нет, перейти к шагу 2.

Как только все пометки расставлены, кратчайшие пути получают, используя соотношение $I(Xi') + c(Xi', Xi) = I(Xi)$ (1).

Для решения задачи второго типа можно применять данный алгоритм для каждой вершины.

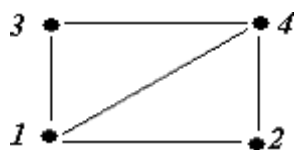
Задача 1

Составить матрицы инцидентности и смежности для графа:



Задача 2

Составить матрицы инцидентности и смежности для графа:



Контрольные вопросы

1. Дайте определение граф.
2. В чем состоит отличие ориентированного графа от неориентированного графа?
3. В чем отличие пустого графа от простого графа?
4. Как определить степень вершины?
5. Чем отличается цепь в графе от цикла?
6. Дайте понятие подграф графа.
7. В чем суть связанного графа?
8. Как находятся матрицы инцидентности и матрицы смежности?
9. Как найти минимальный остов дерева?
10. Как найти кратчайшее расстояние в графе?

Практическое занятие «Составление систем уравнений Колмогорова. Нахождение финальных вероятностей. Нахождение характеристик простейших систем массового обслуживания»

Цель работы: отработать и закрепить умения составлять системы уравнений Колмогорова, отработать и закрепить умения находить финальные вероятности, отработать и закрепить умения определить основные показатели СМО.

Краткая теория

Марковский случайный процесс

Построение математических моделей в условиях неопределенности - очень сложная или невыполнимая задача. Лишь для некоторых упрощенных случаев можно построить математическую модель.

Следует различать два вида неопределенности:

- вероятностные характеристики либо известны, либо могут быть получены в результате эксперимента. Такая неопределенность называется стохастической, и для большинства объектов, содержащих такую неопределенность, можно построить математическую модель, например выход из строя оборудования, приход нового клиента и т. д.
- вероятностные характеристики определить невозможно. В этом случае задачу можно попытаться решить с помощью экспертных оценок, но результат будет весьма приблизительным, например, каковы будут модели женской одежды через пять лет?

Строгую математическую модель с аналитическим вычислением всех интересующих величин можно построить только в том случае, если случайный процесс носит марковский характер.

Случайный процесс будет марковским, если вероятностные характеристики процесса в момент времени t зависят только от текущего (настоящего) состояния процесса в этот момент времени t и не зависят от того, как (каким способом и когда) рассматриваемый процесс перешел в текущее состояние.

Из всего многообразия марковских процессов хорошо изучены и представляют большой практический интерес **марковские случайные процессы с дискретными состояниями и непрерывным временем**.

Под дискретным состоянием будем понимать, что процесс переходит из одного состояния в другое скачкообразно за очень короткое время (практически мгновенно), и количество этих состояний известно (фиксировано).

Под непрерывным временем будем понимать такое, при котором переход из одного допустимого состояния в другое допустимое состояние происходит в произвольные моменты времени, т. е. заранее не определенные.

Потоки событий. Однородные события, следующие друг за другом в произвольные моменты времени (случайно), называются потоком событий (или входным потоком заявок). Примерами потоков событий могут быть: поток пассажиров в авиакассе, поток посетителей парикмахерской, поток отказов технического устройства и т.д. Здесь под событием понимается факт поступления заявок на обработку (приход покупателя, наличие отказа технического средства, поступление телефонного вызова и т.д.), а не результат его обработки (как это рассматривается в теории вероятностей). Поэтому в системах массового обслуживания вероятностными характеристиками будет обладать не отдельное событие, а интервал времени.

Интенсивностью λ потока событий называется среднее число событий за единицу времени. Интенсивность λ может быть как числом постоянным (константой), так и величиной, зависящей от времени t . Например, количество пассажиров в городском транспорте в «часы пик» резко увеличивается по сравнению с другим временем суток.

Финальные вероятности состояний

Будем рассматривать марковские процессы с дискретными состояниями и непрерывным временем. Техническое устройство состоит из трёх узлов и в любой момент времени может находиться в одном из восьми состояний (рис. 1).

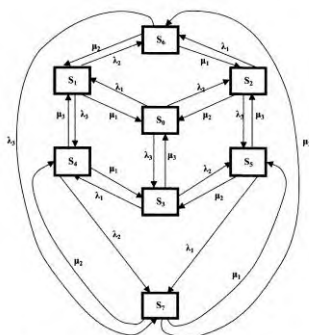


Рис. Состояния технического устройства

Возможные состояния устройства таковы:

S0 - все три узла исправны;

S1 - первый узел неисправен, второй и третий исправны;

S2 - второй узел неисправен, первый и третий исправны;

S3 - третий узел неисправен, первый и второй исправны;

S4 - первый и третий узлы неисправны, второй исправен;

S5 - второй и третий узлы неисправны, первый исправен;

S6 - первый и второй узлы неисправны, третий исправен;

S7 - все три узла неисправны.

Размеченным графом будем считать такой граф, у которого стрелками указаны переходы из одного состояния в другое, а рядом со стрелкой указана интенсивность перехода. Будем различать две интенсивности - прямую λ , и обратную μ . Тогда λ_1, λ_2 и λ_3 - интенсивности потоков отказов соответственно первого, второго и третьего узлов, а μ_1, μ_2 и μ_3 - соответственно интенсивности потоков возвратов (ремонт) узлов. Если для ремонта каждого узла имеется отдельный специалист, то среднее время ремонта каждого узла есть величина постоянная и не имеет значения, один или несколько узлов вышли из строя.

На основе построенного размеченного графа (см. рис. 1) создадим математическую модель.

Наше техническое устройство в соответствии с построенным графом в любой момент времени будет находиться в одном из восьми возможных состояний. Обозначим вероятность каждого i -го состояния как $p_i(t)$, тогда

$$\sum_{i=1} p_i(t) = 1.$$

Для определения вероятности каждого состояния технического устройства составим соответствующие дифференциальные уравнения:

$$\begin{aligned} \frac{d p_0(t)}{dt} &= -\mu_1 p_0 - \mu_2 p_0 - (\mu_1 + \lambda_1 + \lambda_2 + \lambda_3) p_0 \\ \frac{d p_1(t)}{dt} &= \mu_1 p_0 + \mu_2 p_0 - (\mu_1 + \lambda_1 + \lambda_2 + \lambda_3) p_1 \\ \frac{d p_2(t)}{dt} &= \mu_2 p_0 + \mu_1 p_0 - (\mu_2 + \lambda_2 + \lambda_1 + \lambda_3) p_2 \\ \frac{d p_3(t)}{dt} &= \mu_3 p_0 + \mu_1 p_1 + \mu_2 p_2 - (\mu_3 + \lambda_3 + \lambda_1 + \lambda_2) p_3 \\ \frac{d p_4(t)}{dt} &= \lambda_1 p_0 + \lambda_2 p_1 + \lambda_3 p_2 - (\lambda_1 + \mu_1 + \mu_2) p_4 \\ \frac{d p_5(t)}{dt} &= \lambda_2 p_0 + \lambda_1 p_1 + \lambda_3 p_2 - (\lambda_2 + \mu_2 + \mu_1) p_5 \\ \frac{d p_6(t)}{dt} &= \lambda_3 p_0 + \lambda_1 p_2 + \lambda_2 p_1 - (\lambda_3 + \mu_3 + \mu_1 + \mu_2) p_6 \\ \frac{d p_7(t)}{dt} &= \mu_1 p_4 + \mu_2 p_5 + \mu_3 p_6 - (\lambda_1 + \lambda_2 + \lambda_3) p_7 \end{aligned}$$

Эта система дифференциальных уравнений называется **системой уравнений Колмогорова**. Имеем систему из восьми линейных дифференциальных уравнений с восемью неизвестными. Известно, что сумма всех вероятностей равна единице, т. е.

$$p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 = 1$$

Таким образом, любое из уравнений, входящее в систему уравнений, можно записать, используя последнее уравнение, и найти значения вероятностей для каждого события.

Для облегчения процесса составления дифференциальных уравнений можно применить следующее правило:

В *левой* части каждого уравнения следует записать производную вероятности i -го состояния устройства. В *правой* части сумма произведений потока событий, входящих в текущее состояние, умноженная на вероятность состояния, из которого исходит поток, минус суммарная интенсивность исходящих потоков событий из текущего состояния, умноженная на вероятность текущего состояния.

Если финальные вероятности существуют: $\lim_{t \rightarrow \infty} p_i(t) = p_i$ при $i = 1, 2, 3, \dots, n$,

то их сумма будет равна единице:

$$\sum_{i=1} p_i = 1.$$

Финальные вероятности показывают, какое среднее время устройство будет находиться в каждом состоянии. Финальные вероятности находятся из системы дифференциальных уравнений, если их правые части приравнять нулю.

СМО. Основные понятия.

С системами массового обслуживания (СМО) приходится сталкиваться очень часто. Это и работа телефонной станции, и различные очереди (на автозаправке, в поликлинике, в билетной кассе и т.д.), работа некоторых организаций (магазины, мастерские, парикмахерские и т.д.).

Каждая СМО имеет как минимум три элемента: обслуживающий инструмент (станок, касса, канал связи и т.д.), который в дальнейшем будем называть каналом обслуживания или просто *каналом*; *входной поток*, т.е. поток заявок, поступающих на обслуживание; *выходной поток*, т.е. заявки, выполненные СМО (обеспеченные услугой).

Каждая поступившая заявка и принятая на обслуживание внутри СМО обрабатывается некоторое время, называемое *временем обслуживания* - *тоб*. Все заявки поступают случайным образом и независимо друг от друга. Будем рассматривать простейший случай: в каждый момент времени может поступить только одна заявка. Случаи поступления двух и более заявок в один и тот же момент времени не рассматриваются. Таким образом, в некоторые моменты времени поступившие заявки будут скапливаться на входе СМО и ожидать своей обработки либо покидать

СМО необслуженными. В другие моменты времени СМО может простаивать, т.е. не иметь заявок на обслуживание.

График работы СМО представляет собой ступенчатую функцию, т.е. состояние СМО изменяется скачкообразно.

При моделировании работы СМО ставится задача связать технические характеристики СМО. По способу функционирования СМО могут быть:

- открытыми, т.е. поток заявок не зависит от внутреннего состояния СМО;
- закрытыми, т.е. входной поток зависит от состояния СМО (один ремонтный рабочий обслуживает все каналы по мере их выхода из строя).

Одноканальные СМО с отказами

При изучении СМО используем следующие предположения:

1. Входной поток является пуассоновским с параметром λ .
2. Время обслуживания подчиняется экспоненциальному закону с параметром λ :

$$F(t) = \begin{cases} 0, & \text{если } t < 0; \\ 1 - e^{-\lambda t}, & \text{если } t \geq 0 \end{cases}$$

3. Время обслуживания требования не зависит от количества требований, поступивших в систему.

Такая система в любой момент времени t может находиться в одном из двух состояний: E_0 -

в системе 0 требований (система свободна);

E1 - в системе 1 требование (система занята). Далее мы будем находить вероятности:

P_0 – система находится в состоянии E0; P_1 – система находится в состоянии E1.

Начиная с некоторого момента времени, вероятность $P_0(t)$ перестает зависеть от времени и становится постоянной; постоянной будет и $P_1(t)$. Эти величины равны соответственно

$$P_0 = \mu/\lambda + \mu, P_1 = 1 - P_0 = \lambda/\lambda + \mu.$$

В таких случаях говорят, что в системе установился стационарный режим работы. Будем находить коэффициент загрузки системы по формуле

$$\phi = P_1/P_0 = \lambda/\mu.$$

Напомним, что λ – среднее число требований, прибывающих в систему за единицу времени, μ – среднее число обслуженных требований.

Вероятности заставить систему свободной и заставить её занятой, соответственно равны теперь $P_0 = \mu/(\lambda + \mu) = 1/(\lambda/\mu + 1) = 1/(\phi + 1)$, $P_1 = \phi/(\phi + 1)$.

Ясно, что чем больше коэффициент загрузки, тем больше вероятность отказа системы. Это не выгодно потребителю (но выгодно организатору системы, ибо мала вероятность простоя P_0). Если уменьшить коэффициент загрузки, то уменьшится вероятность отказа СМО (это выгодно потребителю), но увеличится вероятность простоя (что не выгодно организаторам системы). Мы имеем дело с противоположными тенденциями и, следовательно, необходимо решать задачи оптимизации режима работы СМО.

Одноканальные СМО с ожиданием

Такие системы при условии, что нет ограничений на длину очереди, имеют бесчисленное множество состояний:

E0, E1, E2, E3, ...

E0 - в системе 0 требований (система свободна); E1 – в системе 1 требование (система занята);

E2 - в системе 1 требование, и одно требование ожидает в очереди;

E3 - в системе 1 требование, и два требования ожидают в очереди и т. д. Для нахождения вероятностей используется следующая формула:

$$P_0 = 1 - \phi, \phi = \lambda/\mu. \text{ Следовательно,}$$

$$P_k = (1 - \phi)\phi^k, k = 1, 2, \dots$$

Условие $\phi < 1$ является необходимым и достаточным для наличия стационарного режима работы системы.

Интересно знать, почему стационарный режим существует только при этом условии?

Это условие означает, что среднее число требований, поступивших в СМО, меньше, чем интенсивность самого обслуживания; поэтому система успевает ритмично работать. Теперь ясно,

почему система не может работать при условии, когда коэффициент загрузки больше 1. Но почему нет установившегося режима, когда коэффициент загрузки равен 1? Ведь в этом случае, сколько в среднем требований поступает в СМО, столько в среднем и обслуживается. Однако требования поступают в систему неравномерно, и время их обслуживания тоже колеблется, так что могут быть и простои, и перегрузки. Вот поэтому при таком условии не поддерживается стационарный режим.

Подсчет средних характеристик

При изучении СМО важнейшими являются средние значения (математические ожидания) таких случайных величин:

n – количество требований, находящихся в системе; v – длина очереди;

w – время ожидания в очереди. Ниже их формулы:

$$n = \phi/(1 - \phi);$$

$$v = \phi^2/(1 - \phi);$$

$$w = [\phi/(1 - \phi)] * [1/\mu].$$

Многоканальные СМО с отказами

Сделаем следующие предположения относительно таких систем:

- входной поток пуассоновский;
- время обслуживания распределено по экспоненциальному закону;
- время обслуживания не зависит от входного потока;
- все линии обслуживания работают независимо.

Будем считать, что система содержит некоторое количество линий обслуживания s . Она может находиться в состояниях $E_0, E_1, E_2, E_3, \dots, E_s$. Расчёт переходных вероятностей показывает, что из каждого из свободных состояний система может переходить в соседнее состояние, либо в такое же, в каком была.

Для нахождения вероятностей используется следующая формула: $P_k = \frac{\varphi^k}{k!} P_0$, $\varphi = \lambda/\mu$, где $k = 1, 2, \dots$

Так как сумма всех вероятностей составляет 1, то $\sum_{k=0}^s \frac{\varphi^k}{k!} P_0 = 1$.

Отсюда следуют формулы:

$$P_0 = \frac{1}{\sum_{k=0}^s \frac{\varphi^k}{k!}}; \quad P_n = \frac{\varphi^n}{n! \sum_{k=0}^s \frac{\varphi^k}{k!}}, \quad n = 0, 1, 2, \dots$$

Увеличение коэффициента загрузки системы ведет к увеличению вероятности отказа системы. Это не устраивает потребителей. Уменьшение вероятности отказа системы может быть достигнуто за счёт увеличения количества линий обслуживания.

Однако резкое увеличение количества линий не устраивает организатора, потому что ведёт к дополнительным затратам на приобретение новых линий обслуживания, и увеличивает вероятность простоя линий. Расчет показывает, что среднее число свободных линий обслуживания $\rho = s - \varphi(1 - P_s)$.

Теперь ясно, что при сильном увеличении количества линий обслуживания, увеличится среднее число простаивающих линий.

Таким образом, мы имеем дело с двумя противоположными тенденциями. Задача сводится к выбору оптимального варианта. С этой целью будем минимизировать функцию стоимости СМО – $C(s)$. Если через c_1 мы обозначим стоимость одного отказа (организатор системы платит штраф за каждый отказ), а через c_2 – стоимость простоя одной линии за единицу времени, то функция стоимости будет иметь следующий вид:

$$C(s) = c_1 \lambda P_s + c_2 \rho.$$

Или в развернутом виде:

$$C(s) = c_1 \lambda \frac{\varphi^s}{s! \sum_{k=0}^s \frac{\varphi^k}{k!}} + c_2 (s - \varphi(1 - P_s)).$$

Сначала с увеличением s она убывает, а затем растёт. Наша задача состоит в том, чтобы найти её минимум.

Многоканальные СМО с ожиданием

Предположения относительно систем, введенные ранее, остаются в силе. Изучение системы ведется по обычной схеме:

1. Выясняются возможные состояния системы (здесь их бесконечное множество).
2. Находятся переменные вероятности.
3. Составляется система уравнений для нахождения P_k – вероятностей пребывания системы в каждом из своих состояний.
4. Изучаем стационарный режим работы СМО.
5. Находятся все вероятности, через P_0 . Результат таков:

$$P_0 = \left(1 + \sum_{k=1}^{\infty} \frac{\varphi^k}{k!} + \frac{\varphi^{s+1}}{s!(s-\varphi)} \right)^{-1}$$

6. Ведётся подсчет средних характеристик: j - среднее количество занятых линий; q - среднее число свободных линий; $P(w > 0)$ – вероятность ожидания; v - средняя длина очереди.

$$j = \varphi; \quad q = s - \varphi;$$

$$P(w > 0) = \varphi^s P_0 / s! (1 - \varphi/s); \quad v = \varphi^{s+1} P_0 / (s-1)! (s - \varphi)^2.$$

Задача 1

Техническое устройство состоит из трёх узлов и в любой момент времени может находиться в одном из восьми состояний (рис. 1). Численные значения интенсивности потоков событий: $\lambda_1=2$; $\lambda_2=2$; $\lambda_3=2$; $\mu_1=2$; $\mu_2=2$; $\mu_3=4$. Найдите финальные вероятности состояний устройства.

Задача 2

Интенсивность потока автомобилей, поступающих на моечную станцию (одноканальная СМО) - 8 автомобилей в час, а интенсивность обслуживания – 9 автомобилей в час. Предполагая, что станция работает в стационарном режиме, найти среднее число автомобилей, находящихся на станции, среднюю длину очереди и среднее время ожидания обслуживания.

Задача 3

Какое оптимальное число линий обслуживания должна иметь СМО, если $\lambda = 7$, $\mu = 8$, $c_1 = 4$, $c_2 = 2$.

Задача 4

Определить число взлетно-посадочных полос для самолётов с учетом требования, что вероятность ожидания $P(w > 0)$ должна быть меньше, чем 0,06. Интенсивность потока равна 18 требований в сутки и интенсивность линий обслуживания - 22 самолётов в сутки.

ки.

Контрольные вопросы:

1. Дайте определение марковскому процессу.
2. Какие типы неопределенностей встречаются.
3. Дайте определение потоку событий.
4. Как составить уравнения Колмогорова.
5. Какие виды СМО Вы знаете?
6. При каких предположениях изучаются одноканальные СМО с отказами?
7. Почему стационарный режим в одноканальных СМО с ожиданием существует только при условии $\varphi > 0$?
8. Какие средние характеристики можно рассчитать в одноканальных СМО с ожиданием?

Практическое занятие «Решение задач массового обслуживания методами имитационного моделирования»

Цель работы: научиться оценивать надежность простейших систем методом Монте - Карло; научиться рассчитывать СМО с отказами методом Монте-Карло.

Краткая теория

Суть имитационного моделирования

Имитационное моделирование - получение экспериментальной информации о сложном объекте, которая не может быть получена иным путем, как экспериментируя с его моделью на ПЭВМ.

Как остроумно подметил Ю. Адлер, сочетание слов имитация и моделирование недопустимо и является тавтологией. Но, рассматривая исторический процесс формирования этого термина,

пришли к выводу, что это словосочетание определяет в моделировании такую область, которая *относится к получению экспериментальной информации о сложном объекте, которая не может быть получена иным путем, как экспериментируя с его моделью на ПЭВМ.*

Имитационный объект имеет вероятностный характер функционирования. Для исследователя представляют интерес выводы, носящие характер статистических показателей, оформленных, может быть, даже в виде графиков или таблиц, в которых каждому варианту исследуемых параметров поставлены в соответствие определенные средние значения с набором характеристик их распределения, без получения зависимости в аналитическом виде.

Эта особенность является и достоинством, и одновременно, недостатком имитационным моделей. Достоинство в том, что резко расширяется класс изучаемых объектов, а недостаток - в отсутствии простого управляющего выражения, позволяющего прогнозировать результат повторного эксперимента. Но в реальной жизни также невозможно для сколько-нибудь сложного объекта получить точное значение экономического показателя, а только лишь его ожидаемое значение с возможными отклонениями.

Главной функцией имитационной модели является воспроизведение с заданной степенью точности прогнозируемых параметров её функционирования, представляющих исследовательский интерес. Как объект, так и его модель, должны обладать системными признаками.

Функционирование объекта характеризуется значительным числом параметров. Особое место среди них занимает временной фактор. В большинстве моделей имеется возможность масштабирования или введения машинного времени, т. е. интервала, в котором остальные параметры системы сохраняют свои значения или заменяются некоторыми обобщенными величинами. Таким образом, за счет этих двух процессов – укрупнения единицы временного интервала и расчета событий этого интервала за зависящий от мощности ПЭВМ временной промежуток – и создается возможность прогноза и расчета вариантов управленческих действий.

Метод Монте-Карло

Неопределённость в предыдущих темах была стохастической. Поэтому строили аналитическую математическую модель и требовали, чтобы в данных задачах, рассматриваемые процессы были марковскими. На практике это не всегда выполняется и тогда требуется использовать методы имитационного моделирования. Что это такое рассказывалось в предыдущем параграфе, а теперь поговорим о самих методах имитационного моделирования.

Метод Монте-Карло является методом статистического моделирования или имитационного моделирования.

Метод Монте-Карло - это численный метод решения задач при помощи моделирования случайных величин.

Датой рождения метода Монте-Карло принято считать 1948 г. Создателями метода считают математиков Дж. Неймана и С. Улама.

Теоретическая основа метода была известна давно. Однако до появления ЭВМ этот метод не мог найти широкого применения.

Само название метода происходит от названия города Монте-Карло в княжестве Монако, знаменитого своими игорными домами. Дело в том, что одним из простейших механических приборов для получения случайных величин является рулетка. Возникает вопрос: помогает ли метод Монте-Карло выигрывать в рулетку? Нет, не помогает. И даже не занимается этим.

Идея метода чрезвычайно проста и состоит в следующем.

Вместо того чтобы описывать процесс с помощью аналитического аппарата, проводится розыгрыш случайного явления с помощью специально организованной процедуры, включающей в себя случайность и дающей случайный результат. Реализация случайного процесса каждый раз складывается по-разному, т. е. мы получаем различные исходы рассматриваемого процесса. Это множество реализаций можно использовать как некий искусственно полученный статистический материал, который может быть обработан обычными методами математической статистики. После такой обработки можно получить: вероятность события, математическое ожидание и т. д.

При помощи метода Монте-Карло может быть решена любая вероятностная задача, но оправданным он является тогда, когда процедура розыгрыша проще, а не сложнее аналитического

расчета.

Оценка надежности простейших систем методом Монте-Карло

Пример: Система состоит из двух блоков, соединенных последовательно. Система оказывает при отказе хотя бы одного блока. Первый блок содержит два элемента: А, В (они соединены параллельно) и оказывает при одновременном отказе обоих элементов. Второй содержит один элемент С и отказывает при отказе этого элемента.

а) Найти методом Монте-Карло оценку P^* надежности (вероятности безотказной работы) системы, зная вероятности безотказной работы элементов: $P(A)=0,8$, $P(B)=0,85$, $P(C)=0,6$; б) найти абсолютную погрешность $|P-P^*|$, где P - надежность системы, вычисленная аналитически. Произвести 50 испытаний.

Решение. а) Выбираем из таблицы приложения (*равномерно распределенные числа*) три случайных числа: 0,10, 0,09 и 0,73; по правилу *) (*если случайное число меньше вероятности события, то событие наступило; если случайное число больше или равно вероятности события, то событие не наступило*) разыграем события А, В, С, состоящие в безотказной работоспособности элементов А, В, С. Результаты испытания будем записывать в расчетную таблицу. Поскольку $P(A)=0,8$ и $0,10 < 0,8$, то событие наступило, т.е. элемент А в этом испытании работает безотказно. Так как $P(B)=0,85$ и $0,09 < 0,85$, то событие В наступило, т.е. элемент В работает безотказно.

Таким образом, оба элемента первого блока работают; следовательно, работает и сам первый блок. В соответствующих клетках табл. ставим знак «+».

| Номер испытания | Блок | Случайные числа, моделирующие элементы | | | Заключение о работе элементов | | | блоков | системы |
|-----------------|------------------|--|------|------|-------------------------------|---|---|--------|---------|
| | | А | В | С | А | В | С | | |
| 1 | Первый Второй | 0,10 | 0,09 | 0,73 | + | + | - | + | - |
| 2 | Первый Второй | 0,25 | 0,33 | 0,76 | + | + | - | + | - |
| 3 | Первый Второй | 0,52 | 0,01 | 0,35 | + | + | + | + | + |
| 4 | Первый Второй | 0,86 | 0,34 | 0,67 | - | + | - | + | - |

Так как $P(C)=0,6$ и $0,73 > 0,6$, то событие С не наступило, т.е. элемент С получает отказ. Другими словами, второй блок, а значит и вся система, получают отказ. В соответствующих клетках табл. 57 ставим минус.

Аналогично разыгрываются и остальные испытания. В табл. приведены результаты четырех испытаний.

Произведя 50 испытаний, получим, что в 28 из них система работала безотказно. В качестве оценки искомой надежности P примем относительную частоту $P^*=28/50=0,56$.

б) Найдем надежность системы P аналитически. Вероятности безотказной работы первого и второго блоков соответственно равны:

$$P_1 = 1 - P(A) * P(B) = 1 - 0,2 * 0,15 = 0,97, P_A = P(C) = 0,6$$

$$\text{Вероятность безотказной работы системы } P = P_1 * P_2 = 0,97 * 0,6 = 0,582$$

$$\text{Искомая абсолютная погрешность } |P - P^*| = 0,582 - 0,56 = 0,022.$$

Расчет СМО с отказами методом Монте-Карло

Пример: В трехканальную систему массового обслуживания с отказом поступает пуассоновский поток заявок. Время между поступлениями двух последовательных заявок распределено по показательному закону $f(t)=5e^{-5t}$. Длительность обслуживания каждой заявки равна 0,5 мин. Найти методом Монте-Карло математическое ожидание a числа обслуженных заявок за время $T=4$ мин.

Решение:

Пусть $T_1=0$ - момент поступления первой заявки. Заявка поступит в первый канал и будет им обслужена. Момент окончания обслуживания первой заявки $T_1+0,5=0+0,5=0,5$. В счетчик обслуженных заявок записываем единицу.

Моменты поступления последующих заявок найдем по формуле $T_i = T_{i-1} + \tau_i$,

где τ_i - длительность времени между двумя последовательными заявками с номерами $i-1$ и i .

Возможные $\tau_i = - (1/\lambda) \ln r_i = (1/\lambda)(-\ln r_i)$.

Учитывая, что, по условию, $\lambda=5$, получим $\tau_i=0,2(-\ln r_i)$.

Случайные числа r_i берем из таблицы приложения, начиная с первой строки сверху. Для нахождения времени между поступлениями первой и второй заявок возьмем случайное число $r=0,10$.

Тогда $\tau_2=0,2*(-\ln 0,10)=0,2*2,30=0,460$. Первая заявка поступила в момент $T_1=0$.

Следовательно, вторая заявка поступила в момент $T_2= T_1+0,460=0,460$. В этот момент первый канал еще занят обслуживанием первой заявки, поэтому вторая заявка поступит во второй и будет им обслужена. Момент окончания обслуживания второй заявки $T_2+0,5=0,460+0,5=0,960$. В счетчик обслуженных заявок добавляем единицу.

По очередному случайному числу $r=0,09$ разыграем время τ_3 между поступлениями второй и третьей заявок:

$\tau_3=0,2(-\ln 0,09)=0,2*2,41=0,482$.

Вторая заявка поступила в момент $T_2= 0,460$. Поэтому третья заявка поступила в момент $T_3= T_2+0,482=0,460+0,482=0,942$. В этот момент первый канал уже свободен и третья заявка поступит в первый канал. Момент окончания обслуживания третьей заявки

$T_3+0,5=0,942+0,5=1,442$. В счетчик обслуженных заявок добавляем единицу.

Дальнейший расчет производят аналогично, причем если момент поступления заявки все каналы заняты (момент поступления заявки меньше каждого из моментов окончания обслуживания), то в счетчик отказов добавляем единицу.

Заметим, что обслуживание 20-й заявки закончится в момент $4,148 > 4$, поэтому эта заявка получает отказ.

| номер заявки | Случайное число r | $-\ln r$ | Время между двумя последовательными заявками $\tau_i=0,2(-\ln r_i)$ | Момент поступления заявки $T_i = T_{i-1} + \tau_i$ | Момент $T_i+0,5$ окончания обслуживания заявки каналом | | | Счетчик | |
|--------------|---------------------|----------|---|--|--|-------|-------|--------------------|---------|
| | | | | | 1 | 2 | 3 | Обслуженных заявок | отказов |
| 1 | | | | 0 | 0,500 | | | 1 | |
| 2 | 0,10 | 2,30 | 0,460 | 0,460 | | 0,960 | | 1 | |
| 3 | 0,09 | 2,41 | 0,482 | 0,942 | 1,442 | | | 1 | |
| 4 | 0,73 | 0,32 | 0,064 | 1,006 | | 1,506 | | 1 | |
| 5 | 0,25 | 1,39 | 0,278 | 1,284 | | | 1,784 | 1 | |
| 6 | 0,33 | 1,11 | 0,222 | 1,506 | 2,006 | | | 1 | |
| 7 | 0,76 | 0,27 | 0,054 | 1,560 | | 2,060 | | 1 | |
| 8 | 0,52 | 0,65 | 0,130 | 1,690 | | | | | 1 |
| 9 | 0,01 | 4,60 | 0,920 | 2,610 | 3,110 | | | 1 | |
| 10 | 0,35 | 1,05 | 0,210 | 2,820 | | 3,320 | | 1 | |
| 11 | 0,86 | 0,15 | 0,030 | 2,850 | | | 3,350 | 1 | |
| 12 | 0,34 | 1,08 | 0,216 | 3,066 | | | | | 1 |
| 13 | 0,67 | 0,40 | 0,080 | 3,146 | 3,646 | | | 1 | |

| | | | | | | | | | |
|----|------|------|-------|--------|-------|-------|-------|-------|---|
| 14 | 0,35 | 1,05 | 0,210 | 3,356 | | 3,856 | | 1 | |
| 15 | 0,48 | 0,73 | 0,146 | 3,502 | | | 4,002 | | 1 |
| 16 | 0,76 | 0,27 | 0,054 | 3,556 | | | | | 1 |
| 17 | 0,80 | 0,22 | 0,044 | 3,600 | | | | | 1 |
| 18 | 0,95 | 0,05 | 0,010 | 3,610 | | | | | 1 |
| 19 | 0,90 | 0,10 | 0,020 | 3,630 | | | | | 1 |
| 20 | 0,91 | 0,09 | 0,018 | 3,648 | 4,148 | | | | 1 |
| 21 | 0,17 | 1,77 | 0,354 | 4,002 | | | | | |
| | | | | (стоп) | | | итого | X1=12 | 8 |

спы
та-
ние
пре
кра

щают (в таблице записывают «стоп»), если момент поступления заявки $T > 4$.

Из таблицы находим, что за 4 мин всего поступило 20 заявок; обслужено $x_1 = 12$. Выполним аналогично еще пять испытаний, получим $x_2 = 15$, $x_3 = 14$, $x_4 = 12$, $x_5 = 13$, $x_6 = 15$.

В качестве оценки искомого математического ожидания а числа обслуженных заявок прием выборочную среднюю

$$a^* = \bar{x} = (2 \cdot 12 + 13 + 14 + 2 \cdot 15) / 6 = 13,5.$$

Задание 1

Система состоит из двух блоков, соединенных последовательно. Первый блок содержит три элемента: А, В, С, а второй - два элемента: D, Е. Элементы каждого блока соединены параллельно.

а) Найти методом Монте-Карло оценку P^* надежности системы, зная вероятности безотказной работы элементов: $P(A) = 0,8$; $P(B) = 0,9$; $P(C) = 0,85$; $P(D) = 0,7$; $P(E) = 0,6$;

б) найти абсолютную погрешность $|P - P^*|$, где P - надежность системы, вычисленная аналитически. Произвести 15 испытаний.

Задание 2

В двухканальную систему массового обслуживания с отказом поступает пуассоновский поток заявок. Время между поступлениями двух последовательных заявок распределено по показательному закону $f(t) = 4e^{-4t}$. Длительность обслуживания каждой заявки равна 1 мин. Найти методом Монте-Карло математическое ожидание a числа обслуженных заявок за время $T = 8$ мин.

Контрольные вопросы:

1. В чем заключается суть имитационного моделирования?
2. В чем заключаются достоинства и недостатки такого типа моделирования?
3. Как применяется метод Монте-Карло?
4. Какие способы получения случайных величин Вы знаете?

Практическое занятие «Инспекция кода модулей проекта»

Цель работы получить практические навыки разработки модулей программной системы и интеграции этих модулей.

Краткая теория

Термин «интеграция» относится к такой операции в процессе разработки ПО, при которой вы объединяете отдельные программные компоненты в единую систему. В небольших проектах интеграция может занять одно утро и заключаться в объединении горстки классов. В больших - могут потребоваться недели или месяцы, чтобы связать воедино весь набор программ. Независимо от размера задач в них применяются одни и те же принципы.

Тема интеграции тесно переплетается с вопросом последовательности конструирования. Порядок, в котором вы создаете классы или компоненты, влияет на порядок их интеграции: вы не можете интегрировать то, что еще не было создано. Последовательности интеграции и конструирования имеют большое значение.

Поскольку интеграция выполняется после того, как разработчик завершил модульное тестирование, и одновременно с системным тестированием, ее иногда считают операцией, относящейся к тестированию. Однако она достаточно сложна, и поэтому ее следует рассматривать как независимый вид деятельности.

Аккуратная интеграция обеспечивает:

- упрощенную диагностику дефектов;
- меньшее число ошибок;
- меньшее количество «лесов»;
- раннее создание первой работающей версии продукта;
- уменьшение общего времени разработки;
- лучшие отношения с заказчиком;
- улучшение морального климата;
- увеличение шансов завершения проекта;
- более надежные оценки графика проекта;
- более аккуратные отчеты о состоянии;
- лучшее качество кода;
- меньшее количество документации.

Интеграция программ выполняется посредством *поэтапного* или *инкрементного* подхода.

Поэтапная интеграция состоит из этапов, перечисленных ниже:

1. «Модульная разработка»: проектирование, кодирование, тестирование и отладка каждого класса.
2. «Системная интеграция»: объединение классов в одну огромную систему.
3. «Системная дезинтеграция»: тестирование и отладка всей системы.

Проблема поэтапной интеграции в том, что, когда классы в системе впервые соединяются вместе, неизбежно возникают новые проблемы и их причины могут быть в чем угодно. Поскольку у вас масса классов, которые никогда раньше не работали вместе, виновником может быть плохо протестированный класс, ошибка в интерфейсе между двумя классами или ошибка, вызванная взаимодействием двух классов. Все классы находятся под подозрением.

Неопределенность местонахождения любой из проблем сочетается с тем фактом, что все эти проблемы вдруг проявляют себя одновременно. Это заставляет вас иметь дело не только с проблемами, вызванными взаимодействием классов, но и другими ошибками, которые трудно диагностировать, так как они взаимодействуют.

Поэтому поэтапную интеграцию называют еще «интеграцией большого взрыва»

Поэтапную интеграцию нельзя начинать до начала последних стадий проекта, когда будут разработаны и протестированы все классы. Когда классы, наконец, будут объединены и проявится большое число ошибок, программисты тут же ударятся в паническую отладку вместо методического определения и исправления ошибок.

Для небольших программ - нет, а для крошечных - поэтапная интеграция может быть наилучшим подходом. Если программа состоит из двух-трех классов, поэтапная интеграция может сэкономить ваше время, если вам повезет. Но в большинстве случаев инкрементный подход будет лучше.

При **инкрементной интеграции** вы пишете и тестируете маленькие участки программы, а затем комбинируете эти кусочки друг с другом по одному. При таком подходе - по одному элементу за раз - вы выполняете перечисленные далее действия:

1. Разрабатываете небольшую, функциональную часть системы. Это может быть наименьшая функциональная часть, самая сложная часть, основная часть или их комбинация. Тщательно тестируете и отлаживаете ее. Она послужит скелетом, на котором будут наращиваться мускулы, нервы и кожа, составляющие остальные части системы.
2. Проектируете, кодируете, тестируете и отлаживаете класс.
3. Прикрепляете новый класс к скелету. Тестируете и отлаживаете соединение скелета и нового класса. Убеждаетесь, что эта комбинация работает, прежде чем переходить к добавлению нового класса. Если дело сделано, повторяете процесс, начиная с п. 2.

Инкрементный подход имеет массу преимуществ перед традиционным поэтапным подходом независимо от того, какую инкрементную стратегию вы используете:

Ошибки можно легко обнаружить. Когда во время инкрементной интеграции возникает новая проблема, то очевидно, что к этому причастен новый класс. Либо его интерфейс с остальной частью программы неправилен, либо его взаимодействие с ранее интегрированными классами приводит к ошибке. В любом случае вы точно знаете, где искать проблему.

В таком проекте система раньше становится работоспособной Когда код интегрирован и способен выполняться, даже если система еще не пригодна к использованию, это выглядит так, будто это скоро произойдет. При инкрементной интеграции программисты раньше видят результаты своей работы, поэтому их моральное состояние лучше, чем в том случае, когда они подозревают, что их проект может никогда не сделать первый вдох.

Вы получаете улучшенный мониторинг состояния При частой интеграции реализованная и нереализованная функциональность видна с первого взгляда. Менеджеры будут иметь лучшее представление о состоянии проекта, видя, что 50% системы уже работает, а не слыша, что кодирование «завершено на 99%».

Вы улучшите отношения с заказчиком Если частая интеграция влияет на моральное состояние разработчиков, то она также оказывает влияние и на моральное состояние заказчика. Клиенты любят видеть признаки прогресса, а инкрементная интеграция предоставляет им такую возможность достаточно часто.

Системные модули тестируются гораздо полнее Интеграция начинается на ранних стадиях проекта. Вы интегрируете каждый класс по мере его готовности, а не ожидая одного внушительного мероприятия по интеграции в конце разработки. Программист тестирует классы в обоих случаях, но в качестве элемента общей системы они используются гораздо чаще при инкрементной, чем при поэтапной интеграции.

Вы можете создать систему за более короткое время Если интеграция тщательно спланирована, вы можете проектировать одну часть системы в то время, когда другая часть уже кодируется. Это не уменьшает общее число человеко-часов, требуемых для полного проектирования и кодирования, но позволяет выполнять часть работ параллельно, что является преимуществом в тех случаях, когда время имеет критическое значение.

При поэтапной интеграции вам не нужно планировать порядок создания компонентов проекта. Все компоненты интегрируются одновременно, поэтому вы можете разрабатывать их в любом порядке - главное, чтобы они все были готовы к часу X.

При инкрементной интеграции вы должны планировать более аккуратно. Большинство систем требует интеграции некоторых компонентов перед интеграцией других. Так что планирование интеграции влияет на планирование конструирования - порядок, в котором конструируются компоненты, должен обеспечивать порядок, в котором они будут интегрироваться.

Нисходящая интеграция

При нисходящей интеграции класс на вершине иерархии пишется и интегрируется первым. Вершина иерархии - это главное окно, управляющий цикл приложения, объект, содержащий метод `main()` в программе на Java, функция `WinMain()` в программировании для Microsoft Windows или аналогичные. Для работы этого верхнего класса пишутся заглушки. Затем, по мере интеграции классов сверху вниз, классы заглушек заменяются реальными.

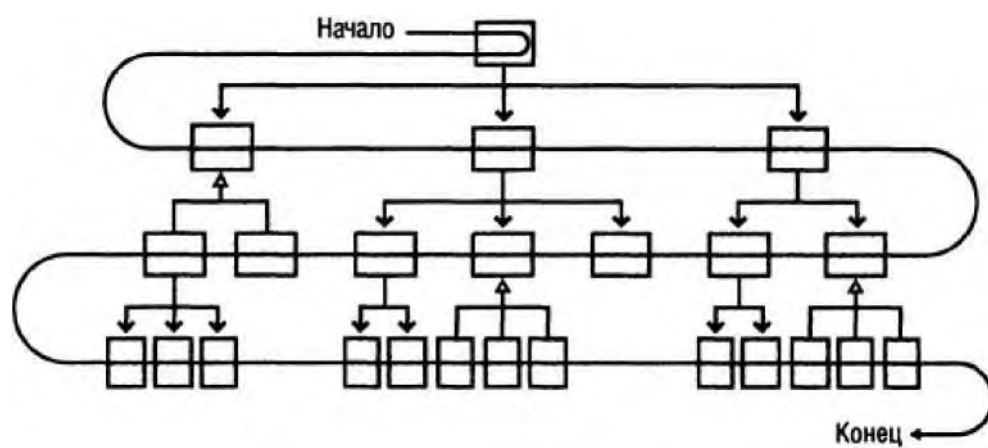


Рис.3 Нисходящая интеграция

При нисходящей интеграции вы создаете те классы, которые находятся на вершине иерархии, первыми, а те, что внизу, - последними.

Хорошей альтернативой нисходящей интеграции в чистом виде может стать подход с вертикальным секционированием.

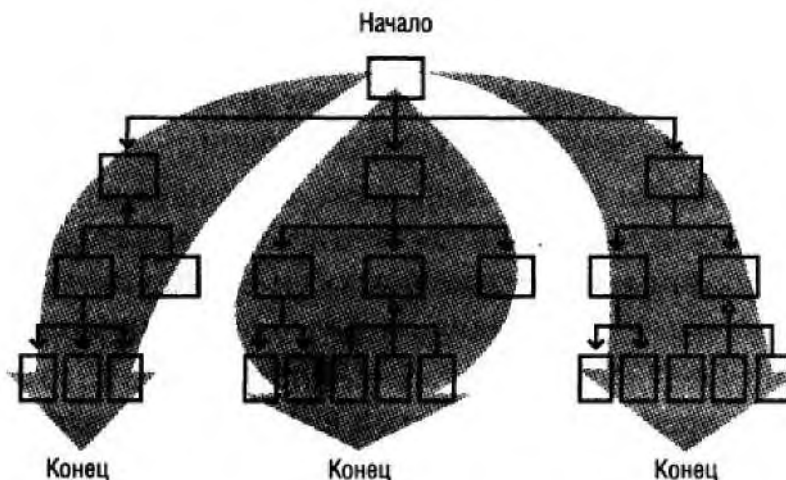


Рис. 4 Вертикальное секционирование

При этом систему реализуют сверху вниз по частям, возможно, по очереди выделяя функциональные области и переходя от одной к другой.

Восходящая интеграция

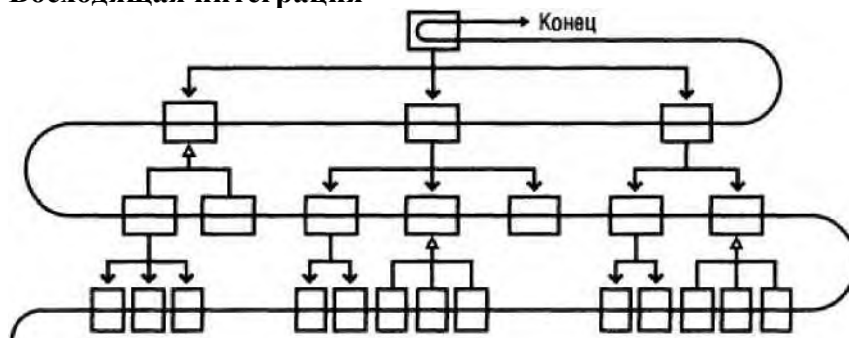


Рис.5 Восходящая интеграция

При восходящей интеграции вы пишете и интегрируете сначала классы, находящиеся в низу иерархии. Добавление низкоуровневых классов по одному, а не всех одновременно - вот что делает восходящую интеграцию инкрементной стратегией. Сначала вы пишете тестовые драйверы для выполнения низкоуровневых классов, а затем добавляете эти классы к тестовым драйверам, пристраивая их по мере готовности. Добавляя класс более высокого уровня, вы заменяете классы драйверов реальными.

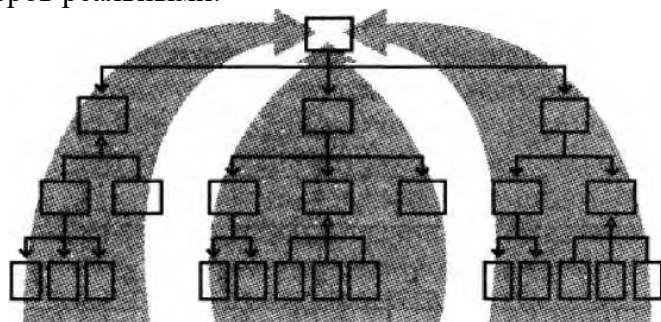


Рис. 6 Гибридный подход при восходящей интеграции

Как и нисходящую, восходящую интеграцию в чистом виде используют редко — вместо нее можно применять гибридный подход, реализующий секционную интеграцию.

Сэндвич-интеграция

Проблемы с нисходящей и восходящей интеграциями в чистом виде привели к тому, что некоторые эксперты стали рекомендовать сэндвич-подход.

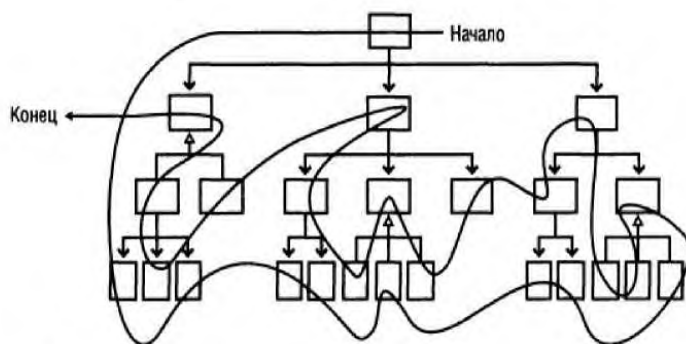


Рис. 7 Сэндвич-интеграция

Сначала вы объединяете высокоуровневые классы бизнес-объектов на вершине иерархии. Затем добавляете классы, взаимодействующие с аппаратной частью, и широко используемые вспомогательные классы в низу иерархии.

Напоследок вы оставляете классы среднего уровня.

Риск-ориентированная интеграция

Риск-ориентированную интеграцию, которую также называют «интеграцией, начиная с самых сложных частей» (hard part first integration), похожа на сэндвич-интеграцию тем, что пытается избежать проблем, присущих нисходящей или восходящей интеграциям в чистом виде. Кроме того, в ней также есть тенденция к объединению классов верхнего и нижнего уровней в первую очередь, оставляя классы среднего уровня напоследок. Однако суть в другом.

При риск-ориентированной интеграции вы определяете степень риска, связанную с каждым классом. Вы решаете, какие части системы будут самыми трудными, и реализуете их первыми.

Функционально-ориентированная интеграция

Еще один подход — интеграция одной функции в каждый момент времени. Под «функцией» понимается не нечто расплывчатое, а какое-нибудь поддающееся определению свойство системы, в которой выполняется интеграция.

Когда интегрируемая функция превышает по размерам отдельный класс, то «единица приращения» инкрементной интеграции становится больше отдельного класса. Это немного снижает преимущество инкрементного подхода в том плане, что уменьшает вашу уверенность об источнике новых ошибок. Однако если вы тщательно тестировали классы, реализующие эту функцию, перед интеграцией, то это лишь небольшой недостаток. Вы можете использовать стратегии инкрементной интеграции рекурсивно, сформировав сначала из небольших кусков отдельные свойства, а затем инкрементно объединив их в систему.

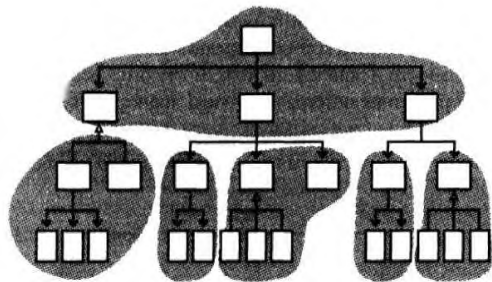


Рис. 8 Функционально-ориентированная интеграция

Обычно процесс начинается с формирования скелета, поскольку он способен поддерживать остальную функциональность. В интерактивной системе такой изначальной опцией может стать система интерактивного меню. Вы можете прикреплять остальную функциональность к той опции, которую интегрировали первой.

Т-образная интеграция

Последний подход, который часто упоминается в связи с проблемами нисходящей и восходящей методик, называется «Т-образной интеграцией». При таком подходе выбирается некоторый вертикальный слой, который разрабатывается и интегрируется раньше других. Этот слой должен проходить сквозь всю систему от начала до конца и позволять выявлять основные проблемы в допущениях, сделанных при проектировании системы. Реализовав этот вертикальный участок (и устранив все связанные с этим проблемы), можно разрабатывать основную канву системы (например, системное меню для настольного приложения). Этот подход часто комбинируют с риск-ориентированной и функционально-ориентированной интеграциями.

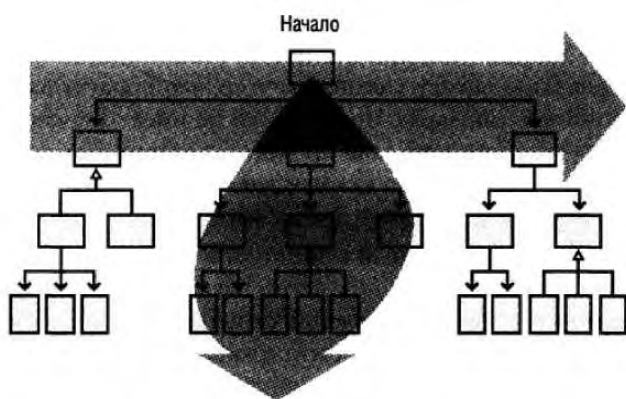


Рис. 9 Т-образная интеграция

Задание 1

1. Оформить внешнюю спецификацию.
2. Составить в виде блок-схемы алгоритм решения задачи.
3. Спроектировать и разработать модули программы для решения задачи на любом алгоритмическом языке программирования.
4. Выполнить отладку и тестирование модулей программы.
5. Выполнить инкрементную интеграцию модулей с использованием одного из подходов.
6. Выполнить системное тестирование программы.
7. Оформить отчет по лабораторной работе.

Задание 2

Задан двумерный массив размерности $n \times m$. Отсортировать элементы строк массива по возрастанию значений, а затем отсортировать строки массива по возрастанию среднего арифметического элементов строк.

Реализовать сортировку разными способами и сравнить эффективность этих способов для разных исходных данных.

Контрольные вопросы

1. Каково значение фазы интеграции программных модулей?
2. Какие есть подходы к интегрированию программных модулей?
3. Критерии эффективности и оптимизация программ.

Практическое занятие «Построение прогнозов»

Цель работы: научиться применять МНК для линейного сглаживания данные, научиться сглаживать данные с помощью квадратичной функции.

Краткая теория

Метод наименьших квадратов - один из методов регрессионного анализа для оценки неизвестных величин по результатам измерений, содержащим случайные ошибки.

Метод наименьших квадратов применяется также для приближённого представления заданной функции другими (более простыми) функциями и часто оказывается полезным при обработке наблюдений.

Метод наименьших квадратов предусматривает нахождение параметров функциональной зависимости из условия минимума суммы квадратов отклонений.

Задание 1

С помощью МНК подобрать параметры a и b линейной функции $y=ax+b$, приближенно описывающей следующие опытные данные. Построить полученную прямую и исходные точки в одной системе координат.

| Вариант | | | | | | | |
|---------|-------|---|---|---|---|---|----|
| 1 | x_i | 1 | 3 | 4 | 5 | 6 | 8 |
| | y_i | 6 | 4 | 4 | 2 | 3 | 2 |
| 2 | x_i | 2 | 3 | 4 | 5 | 7 | 8 |
| | y_i | 1 | 3 | 4 | 6 | 6 | 9 |
| 3 | x_i | 1 | 2 | 4 | 6 | 7 | 8 |
| | y_i | 7 | 6 | 4 | 5 | 3 | 3 |
| 4 | x_i | 2 | 3 | 4 | 5 | 7 | 8 |
| | y_i | 2 | 6 | 6 | 7 | 8 | 10 |

Задание 2

С помощью МНК подобрать параметры a и b квадратичной функции $y=ax^2+bx+c$, приближенно описывающей следующие опытные данные. Построить полученную линию и исходные точки в одной системе координат.

| Вариант | | | | | | | |
|---------|-------|-----|-----|-----|-----|-----|-----|
| 1 | x_i | 2 | 2,5 | 3 | 3,5 | 4 | 4,5 |
| | y_i | 4,2 | 2,5 | 6,2 | 7,5 | 2,6 | 1 |
| 2 | x_i | 1 | 1,5 | 2 | 2,6 | 2,9 | 3,1 |
| | y_i | 2,6 | 5,6 | 4,3 | 1,6 | 2,6 | 3,4 |
| 3 | x_i | 2 | 3 | 3,6 | 3,8 | 4,2 | 4,6 |
| | y_i | 0 | 2,3 | 2,5 | 2,9 | 1 | 4,5 |
| 4 | x_i | 5 | 5,5 | 6 | 6,5 | 7 | 7,5 |
| | y_i | 4,5 | 4,6 | 8,5 | 2,6 | 4,5 | 6,7 |

Контрольные вопросы:

1. Какова общая постановка задачи нахождения эмпирических формул?
2. Каким образом можно оценивать качество приближения?
3. Каким образом графически можно интерпретировать постановку задачи нахождения эмпирических формул?
4. В чем сходство и различие постановки задачи метода наименьших квадратов и задачи интерполяции?
5. Какие виды приближающих функций обычно применяются?
6. В чем суть метода приближения таблично заданной функции по методу наименьших квадратов линейной функцией?
7. Как сводится задача построения различных эмпирических формул к задаче нахождения линейной функции?

Практическое занятие «Решение матричной игры методом итераций»

Цель работы: научиться выбирать оптимальную стратегию игры

Краткая теория

Два предприятия производят продукцию и поставляют её на рынок региона. Они являются единственными поставщиками продукции в регион, поэтому полностью определяют рынок данной продукции в регионе.

Каждое из предприятий имеет возможность производить продукцию с применением одной из трёх различных технологий. В зависимости от качества продукции, произведённой по каждой технологии, предприятия могут установить цену единицы продукции на уровне 12, 8 и 4 денежных единиц соответственно. При этом предприятия имеют различные затраты на производство единицы продукции. (табл. 1).

Таблица 1

Затраты на единицу продукции, произведённой на предприятиях региона (д.е.).

| Технология | Цена реализации единицы продукции, д.е. | Полная себестоимость единицы продукции, д.е. | |
|------------|---|--|---------------|
| | | Предприятие А | Предприятие В |
| 1 | 12 | 8 | 10 |
| 2 | 8 | 5 | 4 |
| 3 | 4 | 2 | 1 |

В результате маркетингового исследования рынка продукции региона была определена функция спроса на продукцию:

$$Y = 10 - 0,6 * X,$$

где Y – количество продукции, которое приобретёт население региона (тыс. ед.), а X – средняя цена продукции предприятий, д.е.

Значения долей продукции предприятия А, приобретённой населением, зависят от соотношения цен на продукцию предприятия А и предприятия В. В результате маркетингового исследования эта зависимость установлена и значения вычислены (табл. 2).

Таблица 2. Доля продукции предприятия А, приобретаемой населением в зависимости от соотношения цен на продукцию.

| Цена реализации 1 ед. продукции, д.е. | | Доля продукции предприятия А, купленной населением |
|---------------------------------------|---------------|--|
| Предприятие А | Предприятие В | |
| 12 | 12 | 0,31 |
| 12 | 8 | 0,33 |
| 12 | 4 | 0,18 |
| 8 | 12 | 0,7 |
| 8 | 8 | 0,3 |
| 8 | 4 | 0,2 |
| 4 | 12 | 0,92 |
| 4 | 8 | 0,85 |
| 4 | 4 | 0,72 |

В задаче необходимо определить:

1. Существует ли в данной задаче ситуация равновесия при выборе технологий производства продукции обоими предприятиями?
2. Существуют ли технологии, которые предприятия заведомо не будут выбирать вследствие невыгодности?
3. Сколько продукции будет реализовано в ситуации равновесия? Какое предприятие окажется в выигрышном положении?

Задача 1

Борьба двух предприятий за рынок в регионе (N – номер варианта)

Две компании, занимающиеся производством антивирусного программного обеспечения, практически полностью делят рынок некоторого региона. Разрабатывая новую версию программного продукта для мобильных телефонов, каждая из компаний может использовать один из четырех вариантов продвижения нового программного продукта на рынок, который влияет на конечную стоимость продукции.

В зависимости от сделанного выбора компании могут установить цену реализации единицы продукции на уровне 25, 22, 19 и 16 условных единиц соответственно. Соотношение цен реализации и себестоимость представлены в таблице:

| Вариант продвижения нового продукта | Цена реализации единицы продукции, у.е. | Полная себестоимость единицы продукции, у.е. | |
|-------------------------------------|---|--|----------------|
| | | Компания А | Компания В |
| 1 | 25 | 17 | $21 - 0,1 * N$ |
| 2 | 22 | 15 | $10 + 0,1 * N$ |
| 3 | 19 | $10 + 0,1 * N$ | 10 |

N – номер варианта, предложенный преподавателем.

В результате маркетингового исследования рынка была определена функция спроса на программные продукты:

$$Y = 20 - 0,5 * X,$$

где Y - количество продукции, которое будет реализовано в регионе (тыс. ед.), а X - средняя цена продукции компаний, у.е.

Значения долей продукции, реализованной компанией А, зависят от соотношения цен на продукцию компании А и компании В. Маркетинговое исследование позволило установить эту зависимость:

| Цена реализации 1 ед. продукции, у.е. | | Доля реализованной продукции компании А |
|---------------------------------------|------------|---|
| Компания А | Компания В | |
| 25 | 25 | 0,31 |
| 25 | 22 | 0,33 |
| 25 | 19 | 0,25 |
| 25 | 16 | 0,2 |
| 22 | 25 | 0,4 |
| 22 | 22 | 0,35 |
| 22 | 19 | 0,32 |
| 22 | 16 | 0,28 |
| 19 | 25 | 0,52 |
| 19 | 22 | 0,48 |
| 19 | 19 | 0,4 |
| 19 | 16 | 0,35 |
| 16 | 25 | 0,6 |
| 16 | 22 | 0,58 |
| 16 | 19 | 0,55 |
| 16 | 16 | 0,5 |

1. Существует ли в данной задаче ситуация равновесия при выборе варианта продвижения продукта на рынок обоими компаниями?
2. Существуют ли варианты, которые компании заведомо не будут выбирать вследствие невыгодности?
3. Сколько продукции будет реализовано в ситуации равновесия? Какая компания получит больше прибыль в ситуации равновесия? Какая компания будет иметь большую долю рынка в ситуации равновесия? Дайте краткую экономическую интерпретацию результатов решения задачи.

Практическое занятие «Моделирование прогноза»

Цель работы: изучение возможностей и формирование умения использования универсальной компьютерной технологии для решения задач выявления тенденций и прогнозирования разви-

тия процесса на основе моделирования рядов динамики (с помощью табличного процессора Excel)

Краткая теория

Тренд - это функция заданного вида, с помощью которой можно аппроксимировать построенный по данным таблицы график. Тренд служит для выявления тенденций развития процесса, представленного в виде диаграммы, и обеспечивает прогноз на заданный период.

В MS Excel предусмотрено несколько стандартных типов тренда: линейный, логарифмический, степенной, экспоненциальный, полиномиальный, скользящее среднее. Необходимые условия для построения тренда:

- период времени, за который изучается исследуемый процесс, должен быть достаточным для выявления закономерности;
- тренд в анализируемый период должен развиваться эволюционно;
- процесс, представленный диаграммой, должен обладать определенной инертностью.
- Тренд можно строить для диаграмм типа:
 - линейчатый график,
 - гистограмма,
 - диаграмма с областями,
 - XY-точечная диаграмма.

При установлении наиболее подходящего типа регрессионной зависимости для описания процесса изменения показателей какой-либо величины используют показатель достоверности описания функции. Тип регрессионной линии считается установленным, если величина достоверности аппроксимации $R^2=1$. Однако, если аппроксимации $R^2 < 0,6$ уместно говорить о том, что тип зависимости для описания процесса изменения показателя не подходит.

Если ни в одном из вариантов исследуемых типов регрессионных линий (трендов) величина достоверности аппроксимации не равна единице, то выбирают тот тип, для которого величина достоверности аппроксимации максимальна.

Индивидуальное задание

Создайте новую рабочую книгу.

1. Выберите таблицу с данными согласно своему индивидуальному варианту.
2. Сохраните результат работы в файл.
3. В ячейку A1 введите – описание переменной x , в ячейку B1 – описание переменной y .
4. Осуществите ввод исследуемых данных в столбцы A и B ниже описанных переменных.
5. Оформите созданную расчетную таблицу
6. Сохраните результат работы в файл.
7. Установить курсор в ячейку C1 и постройте диаграмму “Объем реализации производства наделю” по диапазону значений столбца B.
8. Произведите оформление построенной диаграммы
9. Сохраните результат работы в файл.
10. Выберите Зависимость 1 согласно индивидуальному варианту тип для первой линии тренда.
11. Постройте первый тренд для диаграммы.
12. Произведите настройку оформления вида полученного тренда
13. Выберите Зависимость 2 согласно индивидуальному варианту тип для второй линии тренда.
14. Постройте второй тренд для диаграммы.
15. Произведите настройку оформления вида построенных трендов
16. Произведите анализ полученных результатов.
17. Сохраните результат работы в файл.
18. Предъявите работу преподавателю. Заключительные действия
19. Закройте все открытые файлы электронной таблицы.

20. Закончите работу с MS Excel.

Задание 1

| | | | | | | | | |
|-------------------------------------|----|----|----|----|----|----|----|----|
| День | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Количество проданных ящиков деталей | 13 | 19 | 29 | 30 | 37 | 44 | 49 | 55 |

Исследуемые зависимости: линейная, степенная.

Задание 2

| | | | | | | | | | | |
|---|---|----|----|----|----|----|----|----|----|----|
| Неделя | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Количество поступивших упаковок продукции | 9 | 16 | 20 | 27 | 34 | 39 | 44 | 52 | 58 | 64 |

Исследуемые зависимости: экспоненциальная, логарифмическая.

Контрольные вопросы

1. Что отражает величина достоверности аппроксимации?
2. Дайте определение тренда.
3. В каких случаях необходимо использовать построение трендов?
4. На основе каких данных выбирается наилучшая регрессионная линия?
5. Как изменить формат представления регрессионной линии?
6. Какие типы регрессионных зависимостей Вам известны?
7. Опишите действия необходимые для построения линии тренда по построенной диаграмме.
8. Возможен ли ретроспективный анализ данных с использованием линий тренда?
9. Возможно ли использование регрессионных зависимостей при решении задач по оптимизации ресурсов и запасов?
10. Опишите ситуации, в которых правомочно представление нескольких графиков в одной системе координат.

Практическое занятие «Выбор оптимального решения с помощью дерева решений»

Цель работы: освоение основных методов и способов построения деревьев решений, приобретение практических навыков по использованию инструментария Deductor 4.

Краткая теория

Дерева решений (decision trees) являются одним из самых мощных средств решения задачи отнесения какого-либо объекта (строчки набора данных) к одному из заранее известных классов. Дерево решений – это классификатор, полученный из обучающего множества, содержащего объекты и их характеристики, на основе обучения. Дерево состоит из узлов и листьев, указывающих на класс.

Результатом работы алгоритма является список иерархических правил образующих дерево. Каждое правило - это интуитивно-понятная конструкция вида «Если...то...» (if - then). Дерево может использоваться для классификации объектов, не вошедших в обучающее множество. Чтобы принять решение, к какому классу следует отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы имеют вид «значение параметра А больше В?». Если ответ положительный, осуществляется переход к правому узлу следующего уровня; затем снова следует вопрос, связанный с соответствующим узлом и т.

д.

Настройка назначения полей

Необходимо определить, как будут использоваться поля исходного набора данных при обучении дерева и дальнейшей практической работе с ним.

В левой части окна представлен список всех полей исходного набора данных. Для настройки поля следует выделить его в списке, при этом в правой части окна будут отображены текущие параметры поля:

1. Имя поля – идентификатор поля, определенный для него в источнике данных. Изменить его здесь нельзя.

2. Тип данных – тип данных, содержащихся в поле (вещественный, строковый, дата). Он также задается в источнике данных и здесь изменен быть не может.

3. Назначение – здесь необходимо выбрать порядок использования данного поля при обучении и работе дерева решений. Выбор производится с помощью списка, открываемого кнопкой и содержащего следующие варианты:

- Входное – значения поля будут являться исходными данными для построения и дальнейшей практической работы дерева решений, на их основе будет производиться классификация.

- Выходное – будет содержать результаты классификации. Выходное поле может быть только одно и оно должно быть дискретным.

- Информационное – поле не будет использоваться при обучении дерева, но будет помещено в результирующий набор в исходном состоянии.

- Неиспользуемое – поле не будет использоваться при построении и работе дерева решений и будет исключено из результирующей выборки. В отличие от непригодного, такое поле может быть использовано, если в этом возникнет необходимость.

- Непригодное – поле не может быть использовано при построении и работе алгоритма, но будет помещено в результирующий набор в исходном состоянии.

4. Вид данных – указывает на характер данных, содержащихся в поле (непрерывный или дискретный). Изменить это свойство здесь нельзя.

Статус непригодного поля устанавливается только автоматически и в дальнейшем может быть изменен только на неиспользуемое или информационное. Поле будет запрещено к использованию если:

- поле является дискретным и содержит всего одно уникальное значение;
- непрерывное поле с нулевой дисперсией;
- поле содержит пропущенные значения.

В случае если текущее поле содержит непрерывные (числовые) данные, отображается секция «Статистика», где показываются максимальное и минимальное значения поля, его среднее значение и стандартное отклонение. Если выделенное поле содержит дискретные (строковые) данные, то для него открывается секция «Уникальные значения», в которой отображается общее число уникальных значений поля, а также список самих уникальных значений.

Нормализация полей

Целью нормализации значений полей является преобразование данных к виду, наиболее подходящему для обработки средствами пакета Deductor. Для дерева решений данные, поступающие на вход, должны иметь числовой тип. В этом случае нормализатор может преобразовать дискретные данные к набору уникальных индексов.

Окно настройки нормализации полей вызывается с помощью кнопки «Настройка нормализации». В окне слева приведен полный список входных и выходных полей. При этом каждое поле помечено значком, обозначающим вид нормализации поля:

- линейная – линейная нормализация исходных значений;
- уникальные значения – преобразование уникальных значений в их индексы.

Для числовых (непрерывных) полей с линейной нормализацией дополнительные параметры недоступны. В полях «Минимум» и «Максимум» секции «Диапазон значений» можно посмотреть минимальное и максимальное значения этого поля.

Для дискретных полей справа находится список уникальных значений поля, где для каждого значения указывается количество его повторений. Поле «Количество значений» показывает общее число уникальных значений, принимаемых полем.

Настройка обучающей выборки

Обучающая выборка может быть разбита на три множества – обучающее, тестовое и валидационное.

1. Обучающее множество – включает записи (примеры), которые будут использоваться в качестве входных данных, а также соответствующие желаемые выходные значения.
2. Тестовое множество – также включает записи, содержащие входные и желаемые выходные значения, но используемое не для обучения модели, а для проверки его результатов.
3. Валидационное множество – множество примеров, используемое как для оценки результатов обучения модели, так и определения ее параметров.

Для разбиения исходного множества на обучающее, тестовое и валидационное необходимо настроить несколько параметров:

1. Из списка «Способ разделения исходного множества» выбирается порядок отбора записей во все три множества. Если выбран вариант «по порядку», то порядок следования записей при их разделении не меняется. Множества последовательно формируются в соответствии с определенным для них числом записей. Если выбран вариант «случайно», то отбор записей происходит случайным образом.

2. Затем необходимо указать, какие множества будут использоваться. Для того чтобы множество было сформировано, нужно установить флажок слева от его названия. Если флажок сброшен, то множество использовано не будет. Обучающее множество используется всегда, поэтому сбросить флажок для него нельзя.

3. Для каждого из используемых множеств необходимо задать его размер. Размер может быть задан непосредственно количеством записей или в процентах от объема исходной выборки. Для этого достаточно дважды щелкнуть мышью в соответствующей клетке и ввести нужное значение с клавиатуры. При этом размер, введенный в процентах, автоматически пересчитывается в количество строк и наоборот. В поле «Количество строк (всего)» отображается общее количество записей в исходной выборке данных, которое может быть задействовано для формирования множеств. Если суммарное число строк для всех используемых множеств меньше полного числа строк исходной выборки, то размеры множеств можно задавать произвольно. Можно, например, использовать не все записи, а только часть из них. Если же суммарное указанное число строк превышает максимальное для данной исходной выборки, то автоматически включается баланс множеств, т.е. при указании для одного из множеств размера, в результате которого будет превышено максимальное число записей в исходной выборке, размер остальных множеств будет автоматически уменьшен таким образом, чтобы суммарный размер множеств не превышал доступного числа записей. В строке

«Итого» указывается количество записей, задействованных во всех используемых множествах, а также процент от полного числа записей исходной выборки, который они составляют.

4. В столбце «Порядок сортировки» можно определить порядок следования записей внутри каждого множества. Для этого необходимо дважды щелкнуть мышью в столбце

«Порядок сортировки» для соответствующего множества и с помощью появившейся кнопки выбора открыть список, в котором выбрать один из возможных вариантов:

- по возрастанию – записи в данном множестве будут следовать в порядке возрастания;
- по убыванию – записи в данном множестве будут следовать в порядке убывания;
- случайно – записи в данном множестве будут следовать в случайном порядке.

Для того чтобы обучающее множество было репрезентативным необходимо, чтобы все уникальные значения всех дискретных столбцов содержались в данном наборе данных.

Настройка параметров обучения

Необходимо установить параметры, в соответствии с которыми будет проводиться обуче-

ние дерева:

1. «Минимальное количество примеров, при котором будет создан новый узел» – задается минимальное количество примеров, которое возможно в узле. Если примеров, которые попадают в данный узел, будет меньше заданного – узел считается листом (т.е. дальнейшее ветвление прекращается).
2. «Строить дерево с более достоверными правилами в ущерб сложности» – установка данного флажка включает специальный алгоритм, который, усложняя структуру дерева, увеличивает достоверность результатов классификации. Сброс данного флажка хотя и приводит к упрощению дерева, снижает достоверность результатов классификации.
3. «Уровень доверия, используемый при отсечении узлов дерева». Значение этого параметра задается в процентах и должно лежать в пределах от 0 до 100. Эти значения выбираются из списка. Чем больше уровень доверия, тем более ветвистым получается дерево, и, соответственно, чем меньше уровень доверия, тем больше узлов будет отсечено при его построении.

Задание 1

1. Разработать сценарии построения дерева решений и проведения анализа «что - если».
2. По таблице (например, продаж) создать таблицу транзакций с полями (например, Менеджер, Организация, Вид товара). Таблицу получить путем слияния соответствующих полей из разных таблиц и последующей группировки.
3. Разработать сценарии построения дерева решений с представлением правил, наиболее популярных наборов и анализа «что - если» с входными полями (например, Менеджеры Организация) и выходным полем (например, Вид товара).
4. Создать отчеты по всем разработанным сценариям.
5. Продемонстрировать проект преподавателю с использованием тестовых наборов данных и защитить работу.

Контрольные вопросы

1. С какой целью проводится нормализация значений полей?
2. Для чего используется обучающая выборка? Из каких множеств она состоит?
3. Какие критерии используются для выбора параметров обучения?
4. Какие требования предъявляются к исходным данным при построении дерева решений?
5. Поясните смысл расчетных полей при анализе «что - если».

Контрольные вопросы

Теоретическая часть.

1. Понятие решения. Множество решений, оптимальное решение. Показатель эффективности решения
2. Математические модели, принципы их построения, виды моделей
3. Задачи: классификация, методы решения, граничные условия
4. Общий вид и основная задача линейного программирования. Симплекс-метод
5. Транспортная задача. Методы нахождения начального решения транспортной задачи.
6. Метод потенциалов
7. Общий вид задач нелинейного программирования.
8. Графический метод решения задач нелинейного программирования. Метод множителей Ла-

гранжа

9. Основные понятия динамического программирования: шаговое управление, управление операцией в целом, оптимальное управление, выигрыш на данном шаге, выигрыш за всю операцию, аддитивный критерий, мультипликативный критерий
10. Простейшие задачи, решаемые методом динамического программирования
11. Методы хранения графов в памяти ЭВМ.
12. Задача о нахождении кратчайших путей в графе и методы ее решения
13. Задача о максимальном потоке и алгоритм Форда-Фалкерсона
14. Системы массового обслуживания: понятия, примеры, модели
15. Основные понятия теории марковских процессов: случайный процесс, марковский процесс, граф состояний, поток событий, вероятность состояния, уравнения Колмогорова, финальные вероятности состояний
16. Схема гибели и размножения
17. Метод имитационного моделирования.
18. Единичный жребий и формы его организации. Примеры задач
19. Понятие прогноза.
20. Количественные методы прогнозирования: скользящие средние, экспоненциальное сглаживание, проектирование тренда. Качественные методы прогноза
21. Предмет и задачи теории игр.
22. Основные понятия теории игр: игра, игроки, партия, выигрыш, проигрыш, ход, личные и случайные ходы, стратегические игры, стратегия, оптимальная стратегия
23. Антагонистические матричные игры: чистые и смешанные стратегии
24. Методы решения конечных игр: сведение игры $m \times n$ к задаче линейного программирования, численный метод - метод итераций
25. Область применимости теории принятия решений.
26. Принятие решений в условиях определенности, в условиях риска, в условиях неопределенности
27. Критерии принятия решений в условиях неопределенности.
28. Дерево решений.

Варианты практических заданий

Установить различные профили загрузки для ресурса Подсобник.

Задание 1

Разработать функциональную модель декомпозиции учета движения материалов на складе фирмы

Задание 2

Разработать функциональную модель работы информационной системы приемной комиссии института

Задание 3

Разработать функциональную модель декомпозиции работы информационно-справочной службы фирмы

Задание 4

Разработать функциональную модель работы информационной системы городского бюро медико-социальной экспертизы

Задание 5

Разработать функциональную модель декомпозиции работы информационной системы туристической фирмы

Задание 6

Разработать функциональную модель работы офиса продаж оператора сотовой связи

Задание 7

Разработать функциональную модель декомпозиции работы отдела бухгалтерии предприятия

Задание 8

Разработать функциональную модель работы переговорного пункта

Задание 9

Разработать функциональную модель декомпозиции работы регистратуры центральной районной больницы поселка городского типа

Задание 10

Разработать функциональную модель декомпозиции работы отдела кадров предприятия

Задание 11

Разработать функциональную модель работы учебного отдела вуза

Задание 12

Разработать функциональную модель декомпозиции работы деканата факультета вуза

Задание 13

Разработать в среде RationalRose модель информационной системы страховой компании

Задание 14

Разработать в среде RationalRose модель информационной системы пункта проката видеофильмов

Задание 15

Разработать в среде RationalRose модель информационной системы начисления сдельной заработной платы

Задание 16

Разработать в среде RationalRose модель информационной системы учета транспортных перевозок

Задание 17

Разработать в среде RationalRose модель информационной системы кассы автостанции

Задание 18

Разработать в среде RationalRose модель информационной системы учета заявок клиентов торговой фирмы

Задание 19

Разработать в среде RationalRose модель информационной системы приемной комиссии института

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

| № п.п. | Содержание изменения | Дата, номер протокола заседания кафедр- ры, подпись зав.кафедрой |
|-------------------|-----------------------------|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |