МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры полностью)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Программная реализация решения многопродуктовой транспортной задачи»

Обучающийся О.И. Поженский (И.О. Фамилия) _____ (личная подпись)

Руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень, звание, И.О. Фамилия)

Консультант к.т.н., доцент, Т.С. Якушева
(И.О. Фамилия)

Тольятти 2023

Аннотация

Выпускная квалификационная работа посвящена программной реализации решения многопродуктовой транспортной задачи

Ключевые слова: многоиндексная, трипланарная, транспортная задача, метод потенциалов.

Огромное количество возможных вариантов грузоперевозок делает практически невозможным получение оптимального плана путем простого перебора. Применение математических методов для составления плана перевозок позволяет получить оптимальный план за конечное число итераций. Получение оптимального плана перевозок можно условно поделить на два шага. На первом шаге строится первое опорное решение, для построения первого опорного решения в ВКР используется метод северо-западного угла. На втором шаге полученный опорный план последовательно улучшают методом потенциалов или симплексным методом, в ВКР используется метод потенциалов.

Объектом исследования выпускной квалификационной работы являются методы решения задачи линейного программирования.

Предметом исследования является решение многоиндексной транспортной задачи методом потенциалов.

Целью бакалаврской работы является программная реализация решения проблемы оптимизации в многоиндексной транспортной задаче.

В ходе выполнения выпускной работы было проведено исследование методов определения опорного план и методов последовательного улучшения опорного плана, а также разработан программный продукт на языке Java.

Abstract

The graduation thesis is dedicated to the software implementation of solving the multi-product transportation problem.

Keywords: multi-index, triple-planar, transportation problem, potential method.

The vast number of possible transportation options makes obtaining an optimal plan through simple enumeration practically impossible. The use of mathematical methods for transportation planning allows obtaining the optimal plan within a finite number of iterations. The process of obtaining an optimal transportation plan can be conditionally divided into two steps. At the first step, the initial supporting solution is built using the northwest-corner method. At the second step, the obtained supporting plan is sequentially improved using either the method of potentials or the simplex method.

The object of research in the thesis is the methods of solving linear programming problems.

The subject of research is the solution of the multi-index transportation problem using the potential method.

The goal of the bachelor's work is the software implementation of the optimization problem solution in the multi-index transportation problem.

During the graduation work, the methods of determining the supporting plan and methods of sequential plan improvement were studied, and a software product was developed in Java.

Содержание

Введение

Важность транспортировки для поддержания стабильности в любой системе трудно переоценить. В течение многих лет транспорт был повсеместным явлением, почти необходимостью для человеческого существования. Когда дело доходит до торговли, влияние транспорта нельзя игнорировать. Крайне важно, чтобы процесс транспортировки функционировал бесперебойно, поскольку он стал необходимым аспектом современной жизни, как для частных лиц, так и для бизнеса. Эффективная транспортировка товаров играет решающую роль в торговле и коммерции, поскольку она может значительно повлиять на конечную стоимость продукта. Следовательно, минимизация затрат на транспортировку грузов была и продолжает оставаться важнейшей и неотложной задачей в управлении экономикой.

Актуальность ВКР заключается в том, что любой логистической компании приходится выбирать из большого количества имеющихся возможных вариантов перевозок такого, который даст наименьшее количество издержек на транспортировку груза.

Огромное количество возможных вариантов грузоперевозок препятствует получению оптимального плана, при использовании обычного перебора. Применение же математических методов, позволяет получить оптимальный план, перебрав только конечное количество вариантов.

Объектом исследования выпускной квалификационной работы являются методы решения задачи линейного программирования.

Предметом исследования является решение многоиндексной транспортной задачи методом потенциалов.

Целью бакалаврской работы является программная реализация решения проблемы оптимизации в многоиндексной транспортной задаче.

Задачи, которые необходимо решить для достижения указанной цели:

– Изучить и проанализировать методы решения задач линейного программирования транспортного типа;
– Решить аналитически многоиндексную транспортную задачу с использованием оптимизационных методов;
– Реализовать программный модуль алгоритма решения многоиндексной транспортной задачи;
– Протестировать программный модуль на основе задачи, решенной аналитически с использованием оптимизационных методов.

Структура работы включает в себя введение, три раздела, заключение, список литературы.

В первом разделе рассматривается общая постановка задачи линейного программирования.

Во втором разделе рассматриваются формулировка, модель, свойства трипланарной транспортной задачи. Метод построения начального опорного плана трипланарной транспортной задачи методом последовательного распределения. Метод потенциалов, для последовательного улучшения полученного опорного плана. Аналитическое решение трипланарной транспортной задачи.

В третьем разделе описана программная реализация алгоритмов на языке Java.

1 Анализ задач линейного программирования

Изучение оптимизации перевозок продолжало развиваться на протяжении многих лет с середины тридцатых годов двадцатого века и привело к разработке многочисленных математических моделей и алгоритмов, предназначенных для решения широкого спектра транспортных проблем. Классическая транспортная задача, которая касается транспортировки одного вида товаров, была первой и наиболее широко изученной моделью. Прямой и двойной симплексный методы, известные как метод потенциалов и венгерский метод, были разработаны для эффективного решения этой модели. Позже исследования были сосредоточены на многопродуктовых транспортных моделях, позволяющих изучать различные транспортные сценарии, такие как перевозка нескольких грузов, динамические транспортные задачи и многое другое. Эти достижения в оптимизации транспортировки значительно повлияли на способ перемещения товаров и управления ими, повысив эффективность и сократив затраты.

В дополнение к этим методам существуют также специализированные алгоритмы и программные средства, доступные для решения транспортных задач. Например, алгоритм транспортировки – это специализированный алгоритм, который специально разработан для решения транспортных задач. Он основан на симплексном методе и использует модифицированную версию симплексного алгоритма для поиска оптимального решения. Аналогичным образом, существует ряд доступных пакетов транспортного программного обеспечения, которые могут быть использованы для решения транспортных проблем, таких как Пакет планирования перевозок (TPP) и пакет планирования и оптимизации перевозок (TPOP). Одним из реальных примеров транспортных проблем с несколькими индексами является оптимизация маршрутов доставки для логистической компании. Компании необходимо определить оптимальное количество грузовиков для доставки, их маршруты и количество товаров, которые будут перевозиться на каждом грузовике.

Линейное программирование играет важную роль в оптимизации транспортных задач и их решении.

Линейное программирование (LP) – это математические методы, используемые для оптимизации линейной целевой функции с учетом ограничений, представленных линейными уравнениями или неравенствами. Это метод, используемый для нахождения максимального или минимального значения функции с учетом некоторых ограничений. Цель состоит в том, чтобы найти значения переменных решения, которые оптимизируют целевую функцию, удовлетворяя при этом всем ограничениям. Ограничения и целевая функция представлены в виде линейных уравнений или неравенств, отсюда и название "линейное программирование". LP имеет широкий спектр применений в различных областях, включая экономику, финансы, исследования операций и инжиниринг.

Математическая формула, представленная ниже, является наиболее простой целевой функцией в линейном программировании. Цель состоит в том, чтобы минимизировать сумму произведения коэффициентов (cj) и переменных (xj) для всех j от 1 до n. Цель состоит в том, чтобы найти комбинацию переменных, которая приводит к наименьшему значению целевой функции, с учетом ограничений, которые представляют практические ограничения задачи:

$$L = c_1x_1 + c_2x_2 + \cdots + c_nx_n \quad (1)$$

При условиях:

$$\sum_{J=1}^{n} a_{ij}\, x_j \le b_i, \quad i = 1,2, \dots, k, \quad (2)$$

$$\sum_{J=1}^{n} a_{ij}\, x_j = b_i, \quad i = k+1, k+2, \dots, m \quad (3)$$

Свойства задач линейного программирования, таких как двухосная задача с двумя индексами, хорошо задокументированы в многочисленных научных публикациях.

«Целесообразно выразить задачу линейного программирования в векторно-матричном виде. Представим векторы $X = (x_1, x_2, \dots, x_n)^{\text{т}}$, $B = (b_1, b_2, \dots, b_m)^{\text{т}}$, $C = (c_1, c_2, \dots, c_n)^{\text{т}}$, где т – знак транспонирования» [1]. И матрицу

$$A = (A_1 A_2 \dots A_n) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad (4)$$

Векторы-столбцы $A_j, j \in J$, находящиеся в матрице А, называется вектором условий, что в итоге приводит к возможности формулирования задач (1) - (3) как: поиск вектора X*, минимизирующий (5)

$$L(X) = C^{\text{т}} X \quad (5)$$

И удовлетворяющий ограничениям (6)

$$AX = B, \qquad X \geq 0, \quad (6)$$

Учитывая набор чисел $X = \{x_1, x_2, \dots, x_n\}$, которые удовлетворяют ограничениям задачи линейного программирования, оптимальным планом или решением является тот, который максимизирует или минимизирует значение функции (1). «Задача линейного программирования считается разрешимой, если у нее есть хотя бы один оптимальный план, а план $X = \{x_j\}$ называется эталонным планом, если векторы условий, соответствующие его положительным компонентам, линейно независимы» [1].

Кроме того, базовый план должен содержать неотрицательные компоненты и удовлетворять ограничениям задачи.

Критерий оптимальности основан на предположении, что оптимальным решением является то, которое минимизирует затраты, максимизирует выгоду и выполнимо в рамках заданных ограничений. В нашем случае «для оптимальности плана $X = \{x_1, x_2, \dots, x_n\}$ задачи (1) – (3) необходимо и достаточно существование набора $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$, удовлетворяющего условиям» [1] (7)

$$c_j^0 = c_j - \sum_{i=1}^{m} a_{ij}\lambda_i \geq 0, j \in J; \sum_{i=1}^{m} a_{ij}\lambda_i = c_j, j \in \{j: x_j > 0\} \qquad (7)$$

Следует принять во внимание изучение симплексного метода как способа последовательного улучшения базового плана.

Симплексный метод – это метод математической оптимизации, используемый для решения задач линейного программирования. Она основана на идее перехода от одного осуществимого решения к другому, используя простейшие возможные шаги, таким образом, чтобы целевая функция была улучшена. Этот метод включает в себя определенное количество итераций, которые можно описать следующим образом:

Рассмотрим $X^{(q-1)} = \{x_j^{(q-1)}\}$ в качестве плоскости отсчета, полученной на $(q-1)$-й итерации. Ниже приведено содержимое q-й итерации.

«Для проверки оптимальности плана $X^{(q-1)}$ решают систему линейных уравнений (8)

$$\sum_{i=1}^{m} a_{ij}\lambda_i = c_j, j \in R_{q-1} \qquad (8)$$

где $R_{q-1} = \{j: x_j > 0\}$ – множество существенных индексов.

Эта система имеет уникальное решение, которое состоит из m неизвестных, $\{\tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_m\}$. Это связано с линейной независимостью векторов столбцов матрицы [1] $A = \{a_{ij}\}, j \in R_{q-1}$» [2].

Если критерий (7) удовлетворен, план $X^{(q-1)}$ является наилучшим выбором оптимальности, в противном случае этот план является неоптимальным.

«Для того, чтобы уменьшить значение целевой функции (1) в опорном плане $X^{(q-1)}$ следует ввести дополнительную положительную компоненту x_j^*, для которой $c_{j*} = \min\{c_j\}, j \in J$. Для достижения этой цели решим систему линейных уравнений, представленную ниже (9)

$$\sum_{j \in R_{q-1}} A_j \theta_j = -A_{j*} \tag{9}$$

относительно $\theta_j, j \in R_{q-1}$.

В скалярной форме (9) записывается так (10)

$$\sum_{j \in R_{q-1}} a_{ij} \theta_j = -a_{ij*}, i = 1,2, \dots, m \tag{10}$$

Пусть $\theta'_j, j \in R_{q-1}$ – решение системы (10). Найдем (11)

$$\frac{x_{j1}^{(q-1)}}{\theta'_{j1}} = \theta_0^{(q)} = \max\left\{\frac{x_j^{(q-1)}}{\theta'_j}\right\}, j \in R_{q-1}^- \tag{11}$$

Значения компонент нового опорного плана $X^{(q)} = \{x_j^{(q)}\}$ определяются по формуле» [5](12)

$$x_j^{(q)} = x_j^{(q-1)} - \theta_j^{(q)}\theta_0^{(q)}, j \in J \quad (12)$$

где $\theta_j^{(q)} = \begin{cases} 1, j = j^*; \\ \theta_j', j \in R_{q-1}; \\ 0,\ j \notin R_{q-1} \cup j^*; \end{cases}$

«В результате выполненных операций в новом опорном плане $X^{(q)} = \{x^{(q)}\}$ появляется положительная компонента $x_{j*} = \theta_0^{(q)}$ и одновременно становится равной нулю переменная x_{j1}» [2]. При этом $R_q = j^* \cup R_{q-1}/j_1$.

Выше были рассмотрены методы решения задачи линейного программирования и то, как они могут быть решены с помощью симплексного метода. Однако, принимая во внимание специфические характеристики каждой проблемы, можно получить более эффективные и точные решения. Кроме того, будут рассмотрены другие алгоритмы, которые могут быть применены к задачам линейного программирования.

2 Многопродуктовая транспортная задача

2.1 Модель и формулировка трипланарной транспортной задачи

«Трипланарная транспортная задача – это более сложная версия транспортной задачи, где требуется учитывать затраты на перевозку грузов между тремя уровнями. Каждый уровень представляет собой отдельное пространство, и каждый пункт назначения связан с каждым источником на каждом уровне. Так в отличии от более лёгкой двух-индексной задачи в трипланарной рассматриваются не только расположение центров, а также центров потребления, но и тип транспортного средства, доставляющего товар по пунктам. Решение этой задачи требует учета более сложных условий и может быть решено с помощью более сложных методов оптимизации, таких как генетические алгоритмы или искусственные нейронные сети. В данной работе будет рассмотрено решение аналитическим методом, а также методом потенциалов.» [3]

«Будем считать известными следующие неотрицательные величины:

$a_i, i \in I,$ – количество (запас) продукции, находящейся в распоряжении i-го центра производства;

$b_j, j \in J,$ – количество продукции, необходимое j-му центру потребления;

$c_k, k \in K,$ – количество продукции, которое может быть перевезено всеми единицами транспорта k-го типа;

$c_{ijk}, i \in I, j \in J,\ k \in K,$ – стоимость транспортировки единицы продукта из i-го центра производства в j-й центр потребления транспортным средством k-го типа» [3].

«Совокупность чисел $\{c_{ijk}\}, i \in I, j \in J,\ k \in K,$ как правило, называют трехиндексной матрицей» [4] данную матрицу размера $m \times n \times p$, можно

представить в виде трёхмерной числовой решётки, представленной на рисунке 1.

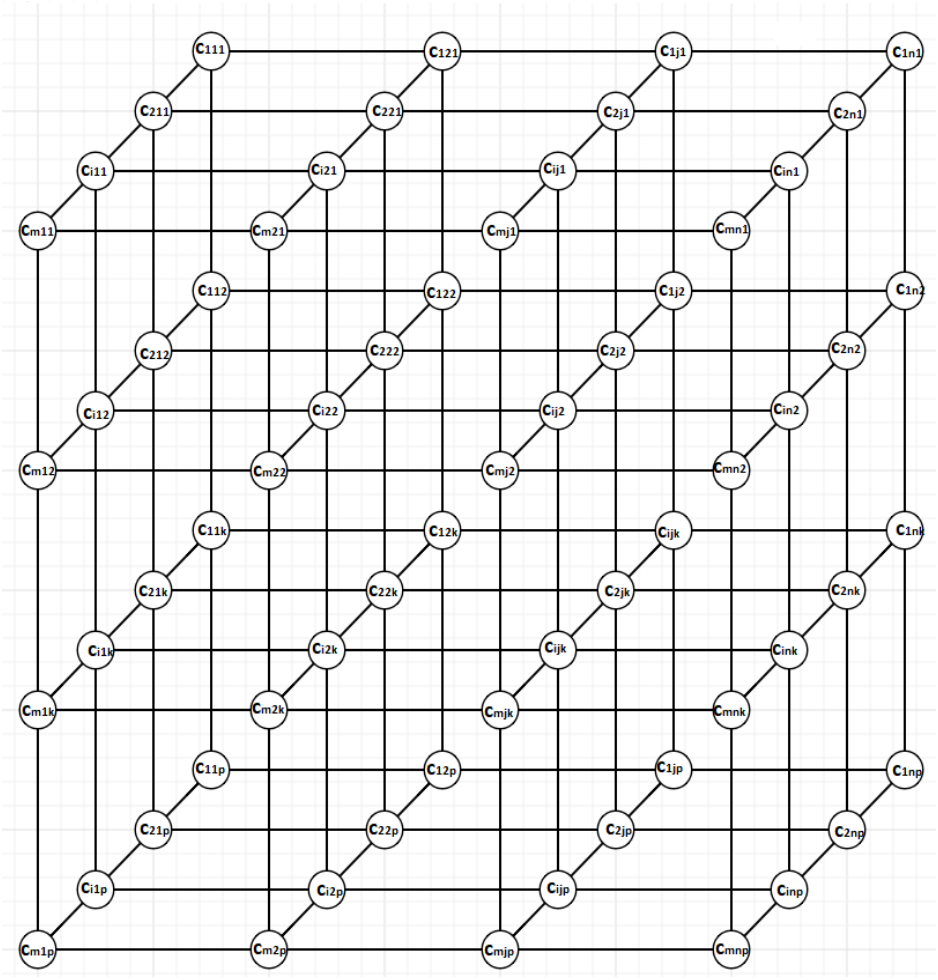


Рисунок 1 – Трехмерная числовая решетка $\{c_{ij}\}$

«Пусть $x_{ijk}, i \in I, j \in J, k \in K$, есть количество продукции, планируемое для перевозки из i-го центра производства в j-й центр потребления транспортными средствами k-го типа. Тогда совокупность чисел $\{x_{ijk}\}, i \in I, j \in J, k \in K$ естественно назвать планом транспортировки, который также представляет собой трехиндексную матрицу. Очевидно, что размеры матриц $\{c_{ijk}\}$ и $\{x_{ijk}\}$ совпадают» [5].

«Элемент x_{ijk} матрицы $\{x_{ijk}\}$ назовем компонентой плана. Заметим, что суммы компонент плана по каждому из сечений матрицы $\{x_{ijk}\}$ имеют вполне определенный смысл. Так, например, сумма элементов матрицы $\{x_{ijk}\}$ по

одномерному сечению $X_{jk}^{i}, x_{jk} = \sum_{i=1}^{m} x_{ijk}$ – это общее количество продукции, доставляемое j-му потребителю транспортными средствами k-го типа» [6].

Другими словами, сумма элементов трехиндексной матрицы по одному измерению называется осевой, тогда как сумма по двум измерениям называется плоскостной.

«План транспортировки должен удовлетворять определенным условиям:

– количество продукции, планируемое для вывоза из каждого центра производства, не должно превышать соответствующий запас,
– количество продукции, планируемое для ввоза каждому из потребителей, должно быть не меньше его потребности,
– общее количество продукции, планируемое для перевозки транспортным средством k-го типа, не должно превышать возможности данных средств» [7].

Формальная запись указанных условий имеет следующий вид (13)

$$\begin{cases} \sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} \le a_i, i \in I; \\ \sum_{i=1}^{m}\sum_{k=1}^{p} x_{ijk} \ge b_j, j \in J; \\ \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} \le c_k, k \in K. \end{cases} \quad (13)$$

В реальных случаях вводится намного больше дополнительных критериев перевозки:

– Затраты на транспортировку товара должны быть сведены к минимуму при обеспечении его качества.

– План транспортировки должен быть составлен таким образом, чтобы в нем учитывались индивидуальные характеристики каждого продукта.

– Доставка товара должна быть произведена в кратчайшие сроки.

– Время погрузки и разгрузки должно быть оптимизировано.

– План транспортировки должен предусматривать безопасную доставку товара.

В конечном итоге можно сделать вывод, что запись формул ограничений имеет вид (14) - (16):

$$\sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} = a_i, i \in I, \quad (14)$$

$$\sum_{i=1}^{m}\sum_{k=1}^{p} x_{ijk} = b_j, j \in J, \quad (15)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} = c_k, k \in K \quad (16)$$

В данном случае, к ограничениям (14) – (16) «требуется добавить требование на неотрицательность для каждой компоненты имеющегося плана» [8](17)

$$x_{ijk} \geq 0 \text{ для всех } i \in I, j \in J, k \in K \quad (17)$$

«Стоимость перевозки x_{ijk} единиц продукта равна $c_{ijk}x_{ijk}$. Суммируя такие произведения по всем трем индексам, получаем общую величину транспортных издержек» [8] (18)

$$L(X) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} c_{ijk} x_{ijk} \qquad (18)$$

«Теперь задача составления плана перевозок может быть сформулирована как задача линейного программирования: найти набор $X^* = \{x^*_{ijk}\}$, минимизирующий функцию (18) и удовлетворяющий условиям (14) – (22)» [9].

«Задача (14) – (18) называется трехиндексной с планарными суммами или трипланарной транспортной задачей и обозначается Т-ЗР. Величины a_i, b_j, c_k, c_{ijk} – параметры задачи Т-ЗР. Набор $\{x_{ijk}\}$ – набор переменных задачи. Параметры полностью определяют задачу» [10].

Рисунок 2 иллюстрирует задачу Т-ЗР с ее ограничениями и матрицей коэффициентов $\{c_{ijk}\}$ в трехмерном пространстве.

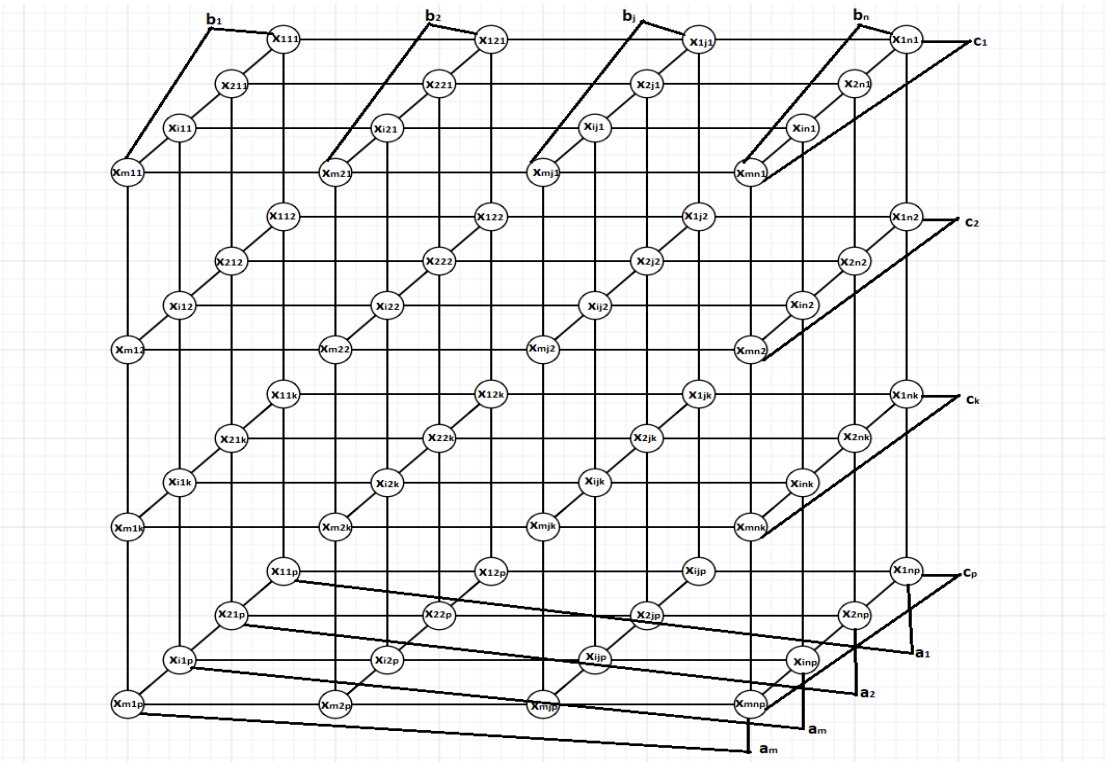


Рисунок 2 – План задачи Т-ЗР, имеющий ограничения

Проблему Т-ЗР можно лучше визуализировать с помощью последовательности блоков в структурной таблице, как показано на рисунке 3, в отличии от объемной геометрической интерпретации констант и переменных.

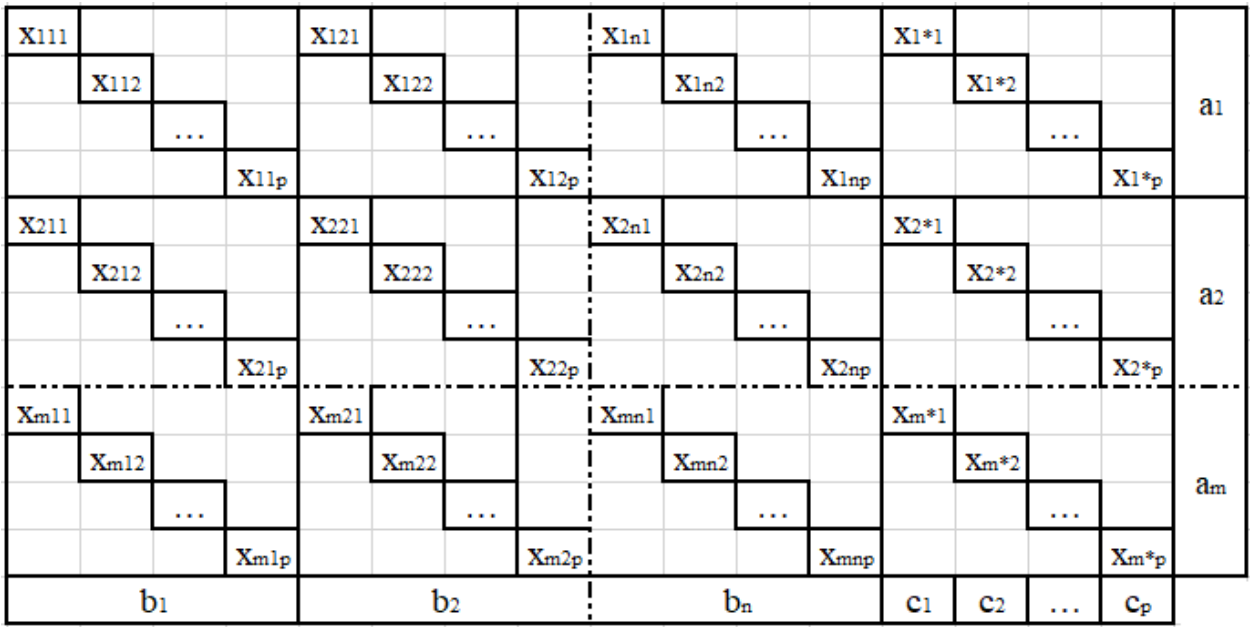


Рисунок 3 – Переменные задачи

«Рассмотрим структуру таблицы 1, называемой таблицей переменных и обозначаемой T_x. Значения переменных x_{ijk} записываются по диагонали каждого блока таблицы. «*», поставленная вместо индекса, означает, что по этому индексу произведено суммирование. Таким образом, $x_{i*k} = \sum_{j=1}^{n} x_{ijk}$.

В нижней строке таблицы записываются свободные члены уравнений (15), в крайнем правом столбце – свободные члены условий (14), в правом нижнем углу – свободные члены условий (15)» [11].

2.2 Свойства задачи Т-ЗР

Перед аналитическим решением следует определить набор переменных, являющихся планом задачи Т-ЗР. «Набор переменных $\{x_{ijk}\}$, удовлетворяющий условиям (14) – (17), называется допустимым планом, или

просто планом задачи. Каждому плану соответствует определенное значение целевой функции (18). Допустимый план, который обеспечивает минимальное значение среди значений целевой функции на множестве всех допустимых планов, называется оптимальным планом задачи. Задача линейного программирования является разрешимой, если существует оптимальный план. Поэтому оптимальный план иногда называют решением задачи» [12].

«Для разрешимости задачи Т-3Р необходимо и достаточно выполнения условия (19)» [12]

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = \sum_{k=1}^{p} c_k = S \qquad (19)$$

«Условие совместимости (19) системы ограничений (14) – (17) называется условием баланса, а задачи, для которых это условие выполняется, сбалансированными транспортными задачами. Таким образом, в сбалансированной трипланарной транспортной задаче общее количество продукции, вывозимое из центров производства, совпадает с общим спросом потребителей и с общей грузоподъемностью выделенного наряда разнотипных транспортных средств» [12].

Далее проанализируем структуру системы ограничений задачи Т-3Р. Перепишем уравнения (14) - (16) для объекта, изображенного на рисунке 4, где m=3, n= 3 и p=2, в более подробной форме.

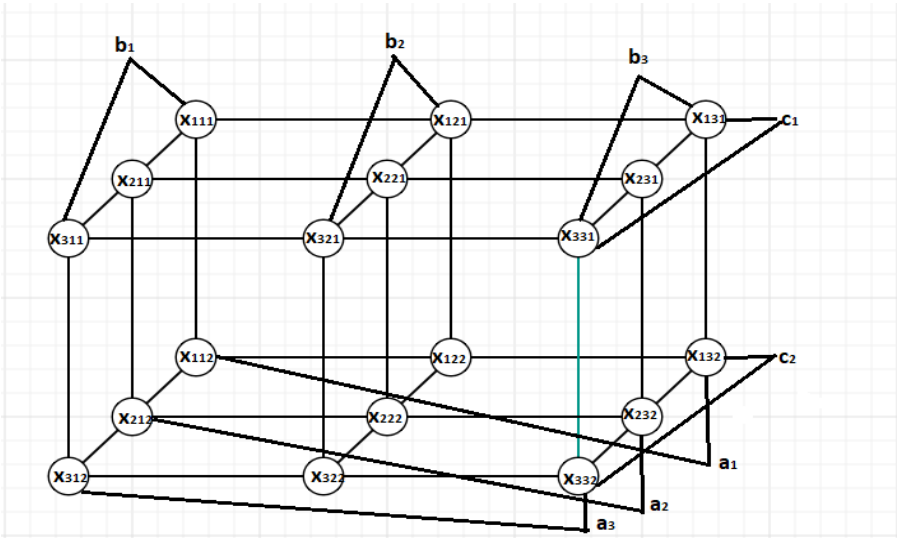


Рисунок 4 – Структурный образец задачи размером $3 \times 3 \times 2$

Далее запишем коэффициенты, представленные ниже (20).

$$\begin{cases} x_{111} + x_{112} + x_{121} + x_{122} + x_{131} + x_{132} = a_1 \\ x_{211} + x_{212} + x_{221} + x_{222} + x_{231} + x_{232} = a_2 \\ x_{311} + x_{312} + x_{321} + x_{322} + x_{331} + x_{332} = a_3 \\ x_{111} + x_{112} + x_{211} + x_{212} + x_{311} + x_{312} = b_1 \\ x_{121} + x_{122} + x_{221} + x_{222} + x_{321} + x_{322} = b_2 \\ x_{131} + x_{132} + x_{231} + x_{232} + x_{331} + x_{332} = b_3 \\ x_{111} + x_{121} + x_{131} + x_{211} + x_{221} + x_{231} + x_{311} + x_{321} + x_{331} = c_1 \\ x_{112} + x_{122} + x_{132} + x_{212} + x_{222} + x_{232} + x_{312} + x_{322} + x_{332} = c_2 \end{cases}, \quad (20)$$

В последствии чего сформируем из них таблицу, представленную на рисунке 5.

		i	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	
		j	1	1	2	2	3	3	1	1	2	2	3	3	1	1	2	2	3	3	R
		k	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
1	E_1		1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	a_1
2			0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	a_2
3			0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	a_3
4	E_2		1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	b_1
5			0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	0	0	b_2
6			0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	1	1	b_3
7	E_3		1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	c_1
8			0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	c_2
λ / G			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

Рисунок 5 – Коэффициенты системы

Пусть система коэффициентов (20) будет определена через П, в свою очередь R будет обозначаться как вектор-столбец свободных членов.

Матрица П – это набор столбцов, каждый из которых является вектором $P_\sigma, \sigma = 1,2,\dots,mnp$, т.е. $\Pi = (P_1, P_2, \dots, P_{mnp})$, а также в трехиндексной нумерации $\Pi = \left(P_{111}, P_{112}, \dots, P_{mnp}\right)$.

Задача Т-ЗР имеет систему ограничений, которая состоит из нескольких групп уравнений типа i, j и k, каждая из которых обозначается соответственно, как (14), (15) и (16). Каждая группа уравнений имеет свои уникальные параметры и ограничения, которые в совокупности определяют возможные решения задачи.

Представленная «матрица $\Pi = (\text{п}_{\lambda\sigma})$, $\lambda \in \{1,2,\dots,m+n+p\}$, $\sigma \in \{1,2,\dots,mnp\}$, имеет размер $(l \times s)$, где $l = m+n+p, s = mnp$, и состоит из нулей и единиц. Множество E_0 номеров строк матрицы П можно разделить на три подмножества E_1, E_2 и E_3 ($\bigcup_{i=1}^{3} E_i = E_0, E_1 \cap E_2 = E_1 \cap E_3 = E_2 \cap E_3 = \emptyset$), порождаемых соответственно группами условий типа i, j и k. Подматрицы Π_1, Π_2 и Π_3, образованные из строк указанных подмножеств, имеют размеры, соответственно» [12] $m \times s, n \times s$ и $p \times s$, следовательно:

$$\sum_{\sigma=1}^{s} \text{п}_{\lambda\sigma} = \begin{cases} np, \lambda \in E_1; \\ mp, \lambda \in E_2; \\ mn, \lambda \in E_3; \end{cases}, \tag{21}$$

$$\sum_{\lambda \in E_p} \text{п}_{\lambda\sigma} = 1, \sigma \in \{1,2,\dots,s\}, p \in \{1,2,3\}, \tag{22}$$

Следует добавить, что, «опорный план задачи Т-ЗР содержит не более чем $m+n+p-2$ положительных компонент. Если опорный план содержит ровно $m+n+p-2$ положительных компонент, то такая задача называется невырожденной» [13].

Для удобства решения задачи (14) - (28) «Трехиндексную матрицу $\{c_{ijk}\}$ и план задачи $\{x_{ijk}\}$ можно записать в виде многомерных векторов $C = (c_{111}, \dots, c_{mnp})$ и $X = (x_{111}, \dots, x_{mnp})$ соответственно» [13]. Тогда решение задачи сводится к поиску вектора X^*, который минимизирует выражение (23). Такой подход позволяет удобно оперировать многомерными данными и сокращает объем вычислений.

$$L(X) = CX \tag{23}$$

а также, «удовлетворяющий ограничениям» (24), (25)

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p} P_{ijk}x_{ijk} = B \tag{24}$$

$$X \geq 0, \tag{25}$$

«Основной метод решения транспортных задач – метод потенциалов – основан на теории двойственности линейного программирования. Сформулируем задачу, двойственную задаче Т-ЗР. Введем вектор двойственных переменных $V = \{u_1, u_2, \dots, u_m, v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_p\}$. Двойственная задача формулируется следующим образом: найти набор $V = \{u_1, u_2, \dots, u_m, v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_p\}$, оптимизирующий функцию» [13] (26)

$$L(V) = \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j + \sum_{k=1}^{p} c_k w_k\,, \tag{26}$$

а также, «удовлетворяющий ограничениям» (27)

$$u_i + v_j + w_k \le c_{ijk}, i \in I, j \in J, k \in K \quad (27)$$

«В задаче Т-ЗР величины $u_i, i \in I, v_j, j \in J, w_k, k \in K,$ называются потенциалами» [13]. Они являются ключевыми параметрами, определяющими характеристики задачи, и используются для ее решения.

Обозначим задачу, полученную из исходной Т-ЗР с помощью процесса двойственности, как $\tilde{\text{Т}}$-ЗР с соответствующими уравнениями (26) и (27). «Если наборы $X = \{x_{ijk}\}$ и $V = \{u_i, v_j, w_k\}$ являются допустимыми планами для задач Т-Зр и $\tilde{\text{Т}}$-ЗР соответственно, то справедливо следующее соотношение» (28)

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p} c_{ijk}x_{ijk} \ge \sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j + \sum_{k=1}^{p} c_k w_k \quad (28)$$

Формула (28) выполнима тогда и только тогда, когда планы X и V являются оптимальными для задач Т-ЗР и $\tilde{\text{Т}}$-ЗР соответственно. Данное соотношение является ключевым в теории двойственности и позволяет связать оптимальные решения прямой и двойственной задач.

«Для оптимальности плана X задачи Т-ЗР необходимо и достаточно существования чисел $u_i, i \in I, v_j, j \in J, w_k, k \in K,$ таких, что (29):

$$u_i + v_j + w_k \le c_{ijk}, i \in I, j \in J, k \in K \quad (29)$$

причем (30):

$$u_i + v_j + w_k = c_{ijk}, (ijk) \in \{(ijk) : x_{ijk} > 0\} \quad (30)$$

Введем следующее преобразование матрицы коэффициентов целевой функции» [13] (31):

$$a_{ijk} = c_{ijk} + \alpha_i + \beta_j + \gamma_k, i \in I, j \in J, k \in K \quad (31)$$

Преобразование, представленное в формуле (31), называется эквивалентным преобразованием, так как оно сохраняет эквивалентность между исходной задачей Т-3Р и ее преобразованной формой. Это преобразование является эффективным инструментом для построения вычислительных алгоритмов, позволяющих решать задачу Т-3Р с высокой точностью и скоростью. Данное преобразование используется в различных областях, включая транспортную логистику, экономику, науку о материалах и другие.

2.3 Составление первоначального ОП транспортной задачи

«Для построения начального опорного плана двухиндексной транспортной задачи обычно применяют так называемый «метод северо-западного узла». Оказывается, что данная процедура обобщение на случай трёх или более индексов» [13] – метод последовательного распределения. Этот метод может быть полезен при решении различных задач, связанных с логистикой, производством, распределением ресурсов и др. Использование данного метода позволяет значительно упростить процесс решения задач и повысить его эффективность.

Метод последовательного распределения – это самый оптимальный вариант построения первоначального опорного плана задачи. Он основан на принципе последовательного распределения грузов между поставщиками и потребителями. Идея заключается в том, что на каждом шаге выбирается наименьшее из неиспользованных количества груза у поставщиков и потребностей у потребителей, и заполняется ими соответствующая ячейка таблицы. Затем уменьшается значение поставщиков и потребностей на

использованное количество груза, повторяется процесс до тех пор, пока не будет заполнен весь опорный план.

«Выберем произвольный элемент $x_{i_1 j_1 k_1}$ матрицы $\{x_{ijk}\}$ и положим его равным $x_{i_1 j_1 k_1} = \min\{a_{i_1}, b_{j_1}, c_{k_1}\}$. Элемент $x_{i_1 j_1 k_1}$ входит в три ограничения: $\sum_{j=1}^{n}\sum_{k=1}^{p} x_{i_1 jk} = a_{i_1}$; $\sum_{i=1}^{m}\sum_{k=1}^{p} x_{ij_1 k} = b_{j_1}$; $\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk_1} = c_{k_1}$. Введем $b'_{j_1} = b_{j_1} - a_{i_1}$ и $c'_{k_1} = c_{k_1} - a_{i_1}$. После определения элемента $x_{i_1 j_1 k_1}$ в сечении $X_{i_1}^{jk}$, все остальные переменные в этом сечении можно установить равными нулю» [13]. Таким образом, это сечение может быть исключено из матрицы $\{x_{ijk}\}$, что приводит к уменьшению ее размерности на одну единицу. «Получив новую усеченную матрицу $\{x_{ijk}\}$ размера $(m-1)\times n\times p$ и новый вектор ограничений $B' = \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_m, b_1, \dots, b_j, \dots, b_n, c_1, \dots, c_{k1}, \dots, c_p\}$» [13], произошла итерация цикла.

Следовательно, «указанная процедура может быть продолжена. Элемент x_{ijk} матрицы назначений, который на очередном шаге вводится в множество ненулевых компонентов плана, будем называть ведущим» [13].

Можно доказать, что общее число шагов алгоритма для получения оптимального плана равно m+n+p-2. Таким образом, «полученный набор удовлетворяет всем ограничениям и, следовательно, является оптимальным планом для задачи Т-ЗР» [13].

2.4 Метод потенциалов для решения трипланарной транспортной задачи

Иногда начальный план распределения ресурсов в задаче Т-ЗР не является лучшим вариантом. Чтобы улучшить его, используется метод потенциалов, который работает в несколько этапов. На первом этапе выполняется проверка на оптимальность полученного плана, и если он проходит этот тест, то его можно считать оптимальным и вычислить целевую функцию задачи на его основе. Однако, если план не проходит тест, то его

можно считать оптимальным и вычислить целевую функцию задачи на его основе. Однако, если план не проходит тест на оптимальность, то переходим ко второму этапу, на котором происходит улучшение опорного плана, чтобы он стал более близким к оптимальному. В итоге, полученный оптимальный план будет иметь более низкое значение целевой функции, чем начальный опорный план, полученный методом последовательного распределения.

Допустим, на $(q-1)$-й итерации имеется опорный план $X^{(q-1)} = \{x_{ijk}^{(q-1)}\}$, а также известна матрица коэффициентов $C^{(q-1)} = \left\{C_{ijk}^{(q-1)}\right\}$.

Индексный элемент – это набор из трех индексов i, j, k, который обозначается как (i, j, k).

Множества значений индексов $I = \{1,2,\dots,m\}$, $J = \{1,2,\dots,n\}$, $K = \{1,2,\dots,p\}$ можно объединить в прямое произведение, которое образует множество всех возможных индексных элементов. Обозначим это множество как E: $E = I \times J \times K$.

Представим, что R_{q-1} – множество, состоящее из индексных элементов, которое равно компонентам плана $X^{(q-1)} = \left\{x_{ijk}^{(q-1)}\right\}$, его аргументы положительны. Следовательно, $R_{q-1} = \{(i, j, k) : x_{ijk}^{(q-1)} > 0\}$.

Множество R_(q-1) будем называть существенным множеством индексных элементов. Если задача невырожденная, то количество элементов в R_(q-1) равно числу m+n+p-2. Рассмотрим, что происходит на q-й итерации.

Перед оптимизацией на первом шаге решается система уравнений (32) для проверки оптимальности плана $X^{(q-1)}$

$$u_i + v_j + w_k = c_{ijk}^{(q-1)}, (i, j, k) \in R_{q-1} \qquad (32)$$

Представленная выше система (32) «состоит из $m + n + p - 2$ уравнений и содержит $m + n + p$ неизвестных. Для получения одного из

решений системы приравняем нулю произвольные две неизвестные с различными индексами, например u_{i_0} и v_{j_0}» [13].

Пусть, что «$\tilde{V} = \{\widetilde{u_1}, \widetilde{u_2}, \dots, \widetilde{u_m}, \widetilde{v_1}, \widetilde{v_2}, \dots, \widetilde{v_n}, \widetilde{w_1}, \widetilde{w_2}, \dots, \widetilde{w_p},\}$ – решение системы» [13] (32). «Числа $\widetilde{u_i}, i \in I, \widetilde{v_j}, j \in J, \widetilde{w_k}, k \in K,$ называются изначальными потенциалами для задачи Т-ЗР» [13].

В соответствии с теоремой (17) «признаком оптимальности опорного плана $X^{(q-1)} = \left\{x_{ijk}^{(q-1)}\right\}$ является выполнение условий:

$$c_{ijk}^{(q)} = c_{ijk}^{(q-1)} - \left(\widetilde{u_i} + \widetilde{v_j} + \widetilde{w_k}\right) \geq 0, (ijk) \in E \qquad (33)$$

Если хотя бы одно из условий (34) не соблюдается, то есть если существует хотя бы один индексный элемент $(i^*j^*k^*)$, для которого значение коэффициента $c_{i^*j^*k^*}^{(q)} < 0$, то план может быть улучшен и осуществляется переход ко второму шагу» [13].

В последствии, на втором шаге «введём в предыдущий план компоненту $x_{i^*j^*k^*}$ для которой» [13]

$$c_{i^*j^*k^*}^{(q)} = \min\left\{c_{ijk}^{(q)}\right\}, (ijk) \in E \qquad (34)$$

Данный шаг реализуется ниже.

«Решим систему линейных уравнений относительно неизвестных переменных θ_{ijk}

$$\sum_{(ijk)\in R_{q-1}} P_{ijk}\,\theta_{ijk} = -P_{i^*j^*k^*} \qquad (35)$$

где P_{ijk} представляет собой вектор-столбец матрицы условий П» [14].

Система уравнений (35) «содержит $m+n+p$ уравнений и $m+n+p-2$ неизвестных. Однако она не является переопределенной, так как уравнения из системы (39) являются следствиями и могут быть отброшены.

Пусть $\{\theta'_{ijk}\}, (ijk) \in R_{q-1}$ – решением системы (35). Введём подмножество индексных элементов: $R^{-}_{q-1} = \{(ijk) : \theta'_{ijk} < 0\}, R^{-}_{q-1}, \subset R_{q-1}$» [15].

Для всех «(ijk), где $\theta'_{ijk} < 0$, вычислим отношения $x^{(q-1)}_{ijk}/\theta'_{ijk}$, затем найдем максимальное из этих отношений и обозначим его как $\theta^{(q)}_0$ с помощью $x^{(q-1)}_{i_1 j_1 k_1}/\theta'_{i_1 j_1 k_1}$» [16]. Следовательно, получим (36)

$$\frac{x^{(q-1)}_{i_1 j_1 k_1}}{\theta'_{i_1 j_1 k_1}} = \theta^{(q)}_0 = \max\left\{\frac{x^{(q-1)}_{ijk}}{\theta'_{ijk}}\right\}, (ijk) \in R_{q-1} \qquad (36)$$

По формуле, представленной ниже, возможно выявить значения компонентов обновлённого опорного плана $X^{(q)} = \left\{x^{(q)}_{ijk}\right\}$

$$x^{(q)}_{ijk} = x^{(q-1)}_{ijk} - \theta^{(q)}_{ijk}\theta^{(q)}_0, \qquad (37)$$

где

$$\theta^{(q)}_{ijk} = \begin{cases} 1, (ijk) = (i^* j^* k^*); \\ \theta'_{ijk}, (ijk) \in R_{q-1}; \\ 0, (ijk) \notin \{R_{q-1} \cup (i^* j^* k^*)\}. \end{cases}, \qquad (38)$$

В соответствии с (37) и (38) «компонента $x^{(q)}_{ijk}$ нового опорного плана $\{x^{(q)}_{ijk}\}$ равна соответствующей компоненте старого плана $\{x^{(q-1)}_{ijk}\}$. Значение переменной $x^{(q)}_{i^* j^* k^*}$, введенной в базис, положим равным $\theta^{(q)}_0$. Значения остальных компонент изменяются (увеличиваются или уменьшаются в

зависимости от знака θ'_{ijk}) на величину $\theta_0^{(q)}\left|\theta'_{ijk}\right|$. В результате проделанных операций в новом опорном плане $X^{(q)} = \{x_{ijk}^{(q)}\}$ появляется новая положительная компонента $x_{i^*j^*k^*}^{(q)} = \left|\theta_0^{(q)}\right| > 0$ и одновременно исключается переменная с индексным элементом $(i_1 j_1 k_1)$, для которого достигается максимум в соотношении (36) $R_q = (i^* j^* k^*) \cup R_{q-1}/(i_1 j_1 k_1)$ » [17].

В последствии идёт цикл из нескольких итераций, который будет ожидать нахождения оптимального решения, представленного в уравнении (37).

2.5 Аналитическое решение трипланарной транспортной задачи

В данном разделе решим трипланарную задачу используя метод последовательного распределения для получения изначального опорного плана, а в последствии улучшая его путём использования метода потенциалов.

Определим набор:

$$X^* = \{x^*_{111}, x^*_{112}, x^*_{121}, x^*_{122}, x^*_{131}, x^*_{132}, x^*_{211}, x^*_{212}, x^*_{221}, x^*_{222}, x^*_{231}, x^*_{232}, \},$$

Который будет оптимизировать (минимизировать) целевую функцию представленную ниже

$$L(X) = 5x_{111} + 32x_{112} + 12x_{121} + 10x_{122} + 11x_{131} + 4x_{211} + 9x_{212} + 21x_{221} + 5x_{222} + 13x_{231} + 8x_{232}$$

А также выполняющий ограничения:

$$\begin{cases} x_{111} + x_{112} + x_{121} + x_{122} + x_{131} + x_{132} = 1 \\ x_{211} + x_{212} + x_{221} + x_{222} + x_{231} + x_{232} = 8 \\ x_{111} + x_{112} + x_{211} + x_{212} = 3 \\ x_{121} + x_{122} + x_{221} + x_{222} = 2 \\ x_{131} + x_{132} + x_{231} + x_{232} = 4 \\ x_{111} + x_{121} + x_{131} + x_{211} + x_{221} + x_{231} = 2 \\ x_{112} + x_{122} + x_{132} + x_{212} + x_{222} + x_{232} = 7 \\ x_{ijk} \geq 0, i = 1,2; j = 1,2,3; k = 1,2 \end{cases}$$

Графически построенный план конкретной трехиндексной трипланарной задачи показан на рисунке 6.

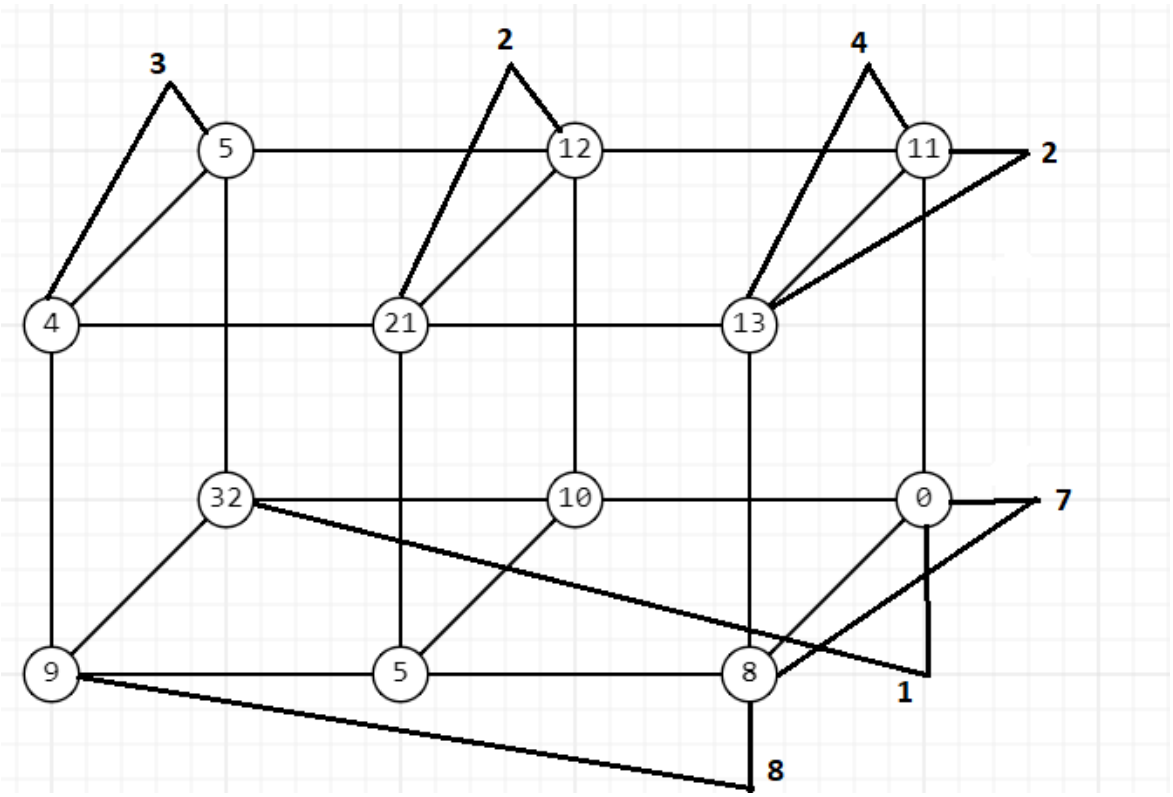


Рисунок 6 – Схема затрат на перевозку, потребителей, поставщиков и транспорта

«На предварительном этапе, при q=0, мы используем метод последовательного распределения для определения начального опорного плана $X^{(0)}$» [18].

«На первом шаге итерации требуется выбрать ведущий элемент x_{111} и определяем $x_{111} = d_1 = \min\{a_1, b_1, c_1\} = \min\{1,3,2\} = 1$. Далее в векторе ограничений переопределим значения:

$$a_1^{(1)} = a_1 - d_1 = 1 - 1 = 0; b_1^{(1)} = b_1 - d_1 = 3 - 1 = 2; c_1^{(1)} = c_1 - d_1 = 2 - 1 = 1.$$

Все остальные элементы переписываются без изменений. Так как $d = a_1$, то $i_2 = 2, j_2 = 1, k_2 = 1$.

Добавим элемент $x_{111} = 1$ в представленную ниже матрицу T_x и присваиваем всем остальным элементам значение 0» [19].

«На втором шаги итерации введём следующий ведущий элемент x_{211}. При этом

$$x_{211} = d_2 = \min\{a_2, b_1, c_1\} = \min\{8,2,1\} = 1.$$

$$a_2^{(2)} = 8 - 1 = 7;\ b_1^{(2)} = 2 - 1 = 1; c_1^{(2)} = 1 - 1 = 0;\ i_3 = 2; j_3 = 1; k_3 = 2.$$

В итоге атрибут $x_{211} = 1$ помещается в матрицу T_x, а остальные элементы становятся 0-м.

Представленные далее расчёты будут приведены без пояснения» [20].

Шаг 3.

$$x_{212} = \min\{7,1,7\} = 1;$$

$$a_2^{(3)} = 7 - 1 = 6;\ b_1^{(3)} = 1 - 1 = 0; c_2^{(3)} = 7 - 1 = 6;\ i_4 = 2; j_4 = 2; k_4 = 2.$$

Шаг 4.

$$x_{222} = \min\{6,2,6\} = 2;$$

$$a_2^{(4)} = 6 - 2 = 4;\ b_2^{(4)} = 2 - 2 = 0; c_2^{(4)} = 6 - 2 = 4;\ i_5 = 2; j_5 = 3; k_5 = 2.$$

Шаг 5.

$$x_{232} = \min\{4,4,4\} = 4;$$

$$a_2^{(5)} = 4 - 4 = 0;\ b_3^{(5)} = 4 - 4 = 0; c_3^{(5)} = 4 - 4 = 0.$$

Предварительный этап по получению изначального «опорного плана $X^{(0)}$ методом последовательного распределения содержит $m + n + p - 2 = 5$ положительных компонентов и является завершённым» [21]. Опорный план представлен на рисунке 7.

	1		2		3		1	2	a_i
1	1		0		0		1		1
		0		0		0		0	
2	1		0		0		1		8
		1		2		4		7	
b_j	3		2		4		2	7	c_k

Рисунок 7 – Начальный опорный план $T_x^{(0)}$

Последующие данные о элементах целевой функции представлены на рисунке 8.

									u_i
	5		12		11				
		32		10		0			
	4		21		13				
		9		5		8			
v_j									w_k

Рисунок 8 – Опорный план нулевой итерации $T_c^{(0)}$

«Выделим элементы $c_{ijk}^{(0)}$ в таблице $T_c^{(0)}$, которые соответствуют положительным переменным $x_{ijk}^{(0)}$» [22]. Значение целевой функции $L\left(X^{(0)}\right)$ будет равно 60, где каждое слагаемое в выражении $1 * 5 + 1 * 4 + 1 * 9 + 2 * 5 + 4 * 8$ соответствует соответствующему элементу $c_{ijk}^{(0)}$ из $T_c^{(0)}$. Множество существенных индексных элементов будет состоять из $\{(1\ 1\ 1), (2\ 1\ 1), (2\ 1\ 2), (2\ 2\ 2), (2\ 3\ 2)\}$ и будет обозначаться как R_0.

Этап решения методом потенциалов. Итерация 1 (q = 1).

На шаге под номером 1 будет происходить проверка оптимальность плана $X^{(0)}$ путем решения системы уравнений:

$$\begin{cases} u_1 + v_1 + w_1 = 5; \\ u_2 + v_1 + w_1 = 4; \\ u_2 + v_1 + w_2 = 9; \\ u_2 + v_2 + w_2 = 5; \\ u_2 + v_3 + w_2 = 8. \end{cases}$$

«Пусть $\widetilde{v_1} = 0$ и $\widetilde{w_1} = 0$. Тогда $\widetilde{u_1} = 5$; $\widetilde{u_2} = 4$; $\widetilde{w_2} = 9 - 4 = 5$; $\widetilde{v_2} = 5 - 4 - 5 = -4$; $\widetilde{v_3} = 8 - 4 - 5 = -1$» [23]. Полученные выше коэффициенты помещены на рисунок 9.

									u_i
	5		12		11				5
		32		10		0			
	4		21		13				4
		9		5		8			
v_j	0		-4		-1		0	5	w_k

Рисунок 9 – Система коэффициентов на первом этапе.

В соответствии с (55) вычислим разности

$$c_{111}^{(1)} = 5 - (5 + 0 + 0) = 0;\ c_{112}^{(1)} = 32 - (5 + 0 + 5) = 22;$$

$$c_{121}^{(1)} = 12 - (5 - 4 + 0) = 11;\ c_{122}^{(1)} = 10 - (5 - 4 + 5) = 4;$$

$$c_{131}^{(1)} = 11 - (5 - 1 + 0) = 7;\ c_{132}^{(1)} = 0 - (5 - 1 + 5) = -9;$$

$$c_{211}^{(1)} = 4 - (4 + 0 + 0) = 0;\ c_{212}^{(1)} = 9 - (4 + 0 + 5) = 0;$$

$$c_{221}^{(1)} = 21 - (4 - 4 + 0) = 21;\ c_{222}^{(1)} = 5 - (4 - 4 + 5) = 0;$$

$$c_{231}^{(1)} = 13 - (4 - 1 + 0) = 10;\ c_{232}^{(1)} = 8 - (4 - 1 + 5) = 0$$

и занесем их на рисунок 10.

									u_i
	0		11		7				-9
		22		4		-9			
	0		21		10				0
		0		0		0			
v_j	0		0		0		0	0	w_k

Рисунок 10 – Система коэффициентов на втором этапе

План «$X^{(0)}$ не является оптимальным, так как $c_{132}^{(1)} = -9 < 0$» [19], следует продолжить работу цикла и перейти на следующий этап.

Шаг 2. Производим повторный проход по алгоритму оптимизации представленный выше. «В соответствии с (38)

$$\min\left\{c_{ijk}^{(1)}\right\} = \min\{0{,}22{,}11{,}4{,}7{,}-9{,}0{,}0{,}21{,}0{,}10{,}0\} = -9 = c_{132}^{(1)}$$

Таким образом, $(i^* j^* k^*) = (1\ 3\ 2)$.

Решим систему уравнений (39), которая в данном случае имеет вид

$$\begin{cases} \theta_{111} = -1; \\ \theta_{111} + \theta_{212} + \theta_{222} + \theta_{232} = 0; \\ \theta_{111} + \theta_{211} + \theta_{212} = 0; \\ \theta_{222} = 0; \\ \theta_{232} = -1; \\ \theta_{111} + \theta_{211} = 0; \\ \theta_{212} + \theta_{222} + \theta_{232} = -1. \end{cases}$$

Решение системы выглядит так: $\theta'_{111} = -1; \theta'_{222} = 0; \theta'_{232} = -1; \theta'_{211} = 1; \theta'_{212} = 0$.

Определяем $\theta_0^{(1)}$ согласно (38), $R_0^- = \{(1\ 1\ 1), (2\ 3\ 2)\}$:

$$\theta_0^{(1)} = \max\left\{\frac{x_{111}^{(0)}}{\theta'_{111}}, \frac{x_{232}^{(0)}}{\theta'_{232}}\right\}, (ijk) \in R_0^- = \{-1, -4\} = -1.$$

Таким образом, $(i_1 j_1 k_1) = (1\ 1\ 1)$.

Вычислим значения компонент нового плана $X^{(1)}$:

$$\theta_{ijk}^{(1)} = \begin{cases} 1, (ijk) = (1\ 3\ 2); \\ \theta'_{ijk}, (ijk) \in R_0; \\ 0, (ijk) \notin R_0 \text{и } (ijk) \neq (\ 1\ 3\ 2). \end{cases}$$

Изменению в старом плане $X^{(0)}$ подвергаются только следующие четыре компоненты:

$$x_{132}^{(1)} = x_{132}^{(0)} - \theta_{132}^{(1)}\theta_0^{(1)} = 0 + 1 = 1; x_{111}^{(1)} = x_{111}^{(0)} - \theta_{111}^{(1)}\theta_0^{(1)} = 1 - 1 = 0;$$

$$x_{211}^{(1)} = x_{211}^{(0)} - \theta_{211}^{(1)}\theta_0^{(1)} = 1 + 1 = 2; x_{232}^{(1)} = x_{232}^{(0)} - \theta_{232}^{(1)}\theta_0^{(1)} = 4 - 1 = 3.$$

Остальные компоненты плана остаются без изменения» [24].

Итоговым значением новой целевой функции является $L\left(X^{(1)}\right) = 0 * 1 + 4 * 2 + 9 * 1 + 5 * 2 + 8 * 3 = 51$, также получен новый опорный план, который будет записан на рисунок 11.

«По итогу ни одно ограничение не нарушилось, а значение целевой функции уменьшилось, следовательно первая итерация завершилась успешно» [25].

	1		2		3		1	2	a_i
1	0		0		0		0		1
		0		0		1		1	
2	2		0		0		2		8
		1		2		3		6	
b_j	3		2		4		2	7	c_k

Рисунок 11 – Опорный план первой итерации $T_x^{(1)}$

$$L\left(X^{(1)}\right) = 51 < L\left(X^{(0)}\right) = 60.$$

Итерация 2 (q = 2).

Шаг 1. Для проверки оптимальности нового опорного плана X^((1)) решим систему уравнений:

$$\begin{cases} u_1 + v_3 + w_2 = -9; \\ u_2 + v_1 + w_1 = 0; \\ u_2 + v_1 + w_2 = 0; \\ u_2 + v_2 + w_2 = 0; \\ u_2 + v_3 + w_2 = 0. \end{cases}$$

Пусть $\widetilde{v_3} = 0$ и $\widetilde{w_2} = 0$. Тогда $\widetilde{u_1} = -9$; $\widetilde{u_2} = 0$; $\widetilde{v_2} = 0$; $\widetilde{v_1} = \widetilde{w_1} = 0$. Вычисленные значения требуется поместить в таблицу $T_c^{(1)}$, затем вычислим разности:

$$c_{111}^{(2)} = 0 - (-9 + 0 + 0) = 9 \geq 0; c_{112}^{(2)} = 22 + 9 = 31 \geq 0;$$

$$c_{121}^{(2)} = 11 + 9 = 20 \geq 0; c_{122}^{(2)} = 4 + 9 = 13 \geq 0; c_{131}^{(2)} = 7 + 9 = 16 \geq 0;$$

$$c_{132}^{(2)} = -9 + 9 = 0 \geq 0; c_{211}^{(2)} = 0 \geq 0; c_{212}^{(2)} = 0 \geq 0; c_{231}^{(2)} = 10 \geq 0; c_{232}^{(2)} = 0 \geq 0.$$

В конечном итоге все значения разностей $c_{ijk}^{(2)}, (ijk) \in E$, неотрицательны, что означает, что опорный план $X^{(1)}$ является оптимальным и совпадает с оптимальным планом $X^{(1)} = X^*$, следовательно, задача решена.

Выводы по 2 разделу.

Во втором разделе изложена математическая модель трипланарной транспортной задачи, которая состоит из целевой функции и ограничений. Решение задачи наглядно проиллюстрировано графически, используя трехмерную числовую решетку для представления трехиндексной матрицы.

В разделе также рассмотрены свойства задачи Т-3Р, определен допустимый план для открытой и сбалансированной транспортных задач, а также сформулирован критерий оптимальности. Для решения трипланарной транспортной задачи представлены методы построения начального опорного плана и метод потенциалов, который последовательно улучшает данный опорный план.

3 Программная реализация программного продукта и его методов

3.1 Определение этапов создания программного продукта

В разделе также рассмотрены свойства задачи Т-ЗР, определен допустимый план для открытой и сбалансированной транспортных задач, а также сформулирован критерий оптимальности.

Создание программного продукта – это долгий и трудоемкий процесс, который включает в себя множество этапов. Каждый этап имеет свои особенности, свои подходы и свои цели. Одним из ключевых этапов является составление программы. Это сложный процесс, который требует глубоких знаний в программировании, технологиях и методологиях разработки.

Первый этап составления программы – это проектирование. Это процесс создания концепции продукта, которая определяет функциональность и основные характеристики программы. На этом этапе определяются требования к продукту, выбираются подходящие технологии и методологии разработки, а также формируется общая архитектура продукта.

Второй этап – выбор подходящих библиотек и фреймворков. На этом этапе программисты выбирают уже готовые компоненты и инструменты, которые позволяют реализовать требуемую функциональность и ускорить процесс разработки. На выбор библиотек и фреймворков влияют множество факторов, включая качество кода, удобство использования, производительность и т.д.

Третий этап – внедрение паттернов программирования. Паттерны программирования – это стандартные решения для типичных задач в программировании. Использование паттернов программирования позволяет ускорить процесс разработки и упростить поддержку продукта в будущем. На этом этапе разработчики выбирают наиболее подходящие паттерны для решения конкретных задач.

Четвертый этап – подключение базы данных. База данных – это ключевой компонент многих программных продуктов. Она позволяет хранить и обрабатывать большие объемы данных, а также обеспечивает безопасность и надежность работы продукта. На этом этапе программисты выбирают подходящую базу данных, проектируют ее структуру и подключают ее к программе.

Одним из основных инструментов помогающих выполнить все указанные выше шаги является управление большого проекта, а также добавление специальной структуры разработки, одной из которых является система Agile. Пример Agile алгоритма представлен на рисунке 12.

Рисунок 12 – Циклы методологии Agile

Agile – это способ разработки программного обеспечения, который основывается на гибкости и адаптивности.

Идея заключается в том, чтобы быстро создавать версии продукта, проверять их и получать обратную связь от пользователей и клиентов. Agile разделен на короткие циклы, которые длительностью 2-4 недели, в течение которых создается рабочая версия продукта. Каждый цикл должен заканчиваться выпуском рабочей версии продукта, которую можно протестировать и оценить. Agile позволяет быстро адаптироваться к

изменяющимся условиям, учитывать новые требования и корректировать планы.

Эта методология также ставит на первое место командную работу и взаимодействие с заказчиком, что позволяет уменьшить количество ошибок и повысить качество продукта. Основная цель Agile – получение рабочего и полезного продукта в кратчайшие сроки. Agile методология помогает ускорить разработку программного обеспечения, снизить риски и затраты, улучшить качество и повысить удовлетворенность клиентов.

Кроме того, Agile позволяет лучше контролировать процесс разработки и быстро реагировать на изменения в условиях рынка или требованиях заказчика.

В итоге, разработка программного продукта является сложным и многопроходным процессом, который требует сочетания глубоких знаний, обширного опыта и творческого мышления. Однако, следует отметить, что успешное завершение этого процесса невозможно без правильно составленной программы, которая является главным фактором обеспечения успеха в данной области.

3.2 Поиск подходящих инструментов разработки

Одно из важнейших требований к реализации приложения, является простота, понимая, точность и безопасность операций – данные условия могут быть выполненным благодаря языку Java, одному из актуальнейших языков в данный момент, пользующимся популярностью во многих крупных IT-компаниях.

Java – это язык программирования, созданный компанией Sun Microsystems в 1995 году, который стал очень популярным. Он используется для создания различных приложений, включая веб-приложения, мобильные приложения, настольные приложения и игры.

Одним из преимуществ языка Java является его платформонезависимость. Это означает, что программы, написанные на Java, могут работать на любой операционной системе, поддерживающей виртуальную машину Java (JVM). Это упрощает процесс разработки приложений, потому что программистам не нужно создавать различные версии приложений для разных платформ.

Java также обладает объектно-ориентированным подходом, где все является объектом, что делает язык более гибким и понятным для разработчиков. Это также означает, что код можно переиспользовать и модифицировать более эффективно.

Библиотеки и фреймворки – еще одно преимущество Java, которые упрощают процесс разработки приложений. Например, в Java есть множество библиотек для работы с базами данных, сетевыми протоколами и графическим интерфейсом. Благодаря этим инструментам разработчики могут создавать приложения быстрее и с меньшими затратами.

Java также известна своей безопасностью. В языке есть механизмы для контроля доступа к данным и обеспечения безопасности виртуальной машины. Это делает Java хорошим выбором для создания приложений, которые должны быть безопасными и надежными.

Кроме того, Java является одним из самых распространенных языков программирования в мире. Это означает, что есть огромное сообщество разработчиков, которые могут помочь друг другу и делиться опытом.

Java является мощным и гибким языком программирования, который предлагает целый ряд преимуществ, позволяющих разработчикам эффективно создавать приложения в различных областях. Эти преимущества включают платформонезависимость, объектно-ориентированный подход, богатые наборы библиотек и фреймворков, высокий уровень безопасности и распространенность в различных сообществах разработчиков.

Более того, эти характеристики Java позволяют программистам создавать приложения, которые могут работать на разных операционных

системах, облегчают переиспользование кода и повышают качество и надежность создаваемых программных продуктов.

3.3 Настройка окружающей среды разработки

Структура программы состоит из нескольких компонентов, включая подключенную библиотеку iostream и ряд функций, каждая из которых выполняет определенную задачу. Основные компоненты программы:

- перечисление данных программы – требуемые элементы, содержащие в себе информацию о изготовителях и потребителях;
- метод вычисления опорного плана – программная реализация циклов, получающих изначальный не оптимальный результат;
- показательные методы – методы выводящие результат работы в консоль приложения;
- метод решения систем – реализация алгоритмов, рассмотренных во втором разделе;
- метод проверки – анализирует конечность и полноту результата работы программы;
- метод завершения программы – обобщает полученные данные в один блок и останавливает программу;
- метод нахождения разности – позволяет определить необходимый результат для вычисления итогового целевого значения программы.

Каждая из этих функций является необходимой для успешного выполнения программы и выполняет свою роль в процессе решения трипланарной транспортной задачи.

Программа начинается с объявления глобальных переменных или констант, которые будут использоваться в различных функциях программы. Это позволяет упростить код, уменьшить количество повторяющихся выражений и обеспечить однообразность данных.

Параметры количества потребителей, поставщиков и количества транспортных средств будут введены из основного конструктора, представленного на рисунке 13.

```
//количество поставщиков
final int m = 2;

//количество потребителей
final int n = 3;

//количество транспортных средств
final int p = 2;
```

Рисунок 13 – Глобальные переменные

В языке Java для объявления константы используется ключевое слово final, которое указывает на то, что значение переменной не может быть изменено после ее инициализации. Следующим шагом определяются массивы количества запасов a[m], который представлен как номер поставщика. Массив b[n] – количество потребителей, имеющий корреляцию с номером поставщика в программе, а также последовательность элементов c[p] – сумма груза, которое способно транспортировать определённый вид перевозок. Объединяющей матрицей является задающая стоимость перевозок матрица T[m][n][p], представленная на рисунке 14.

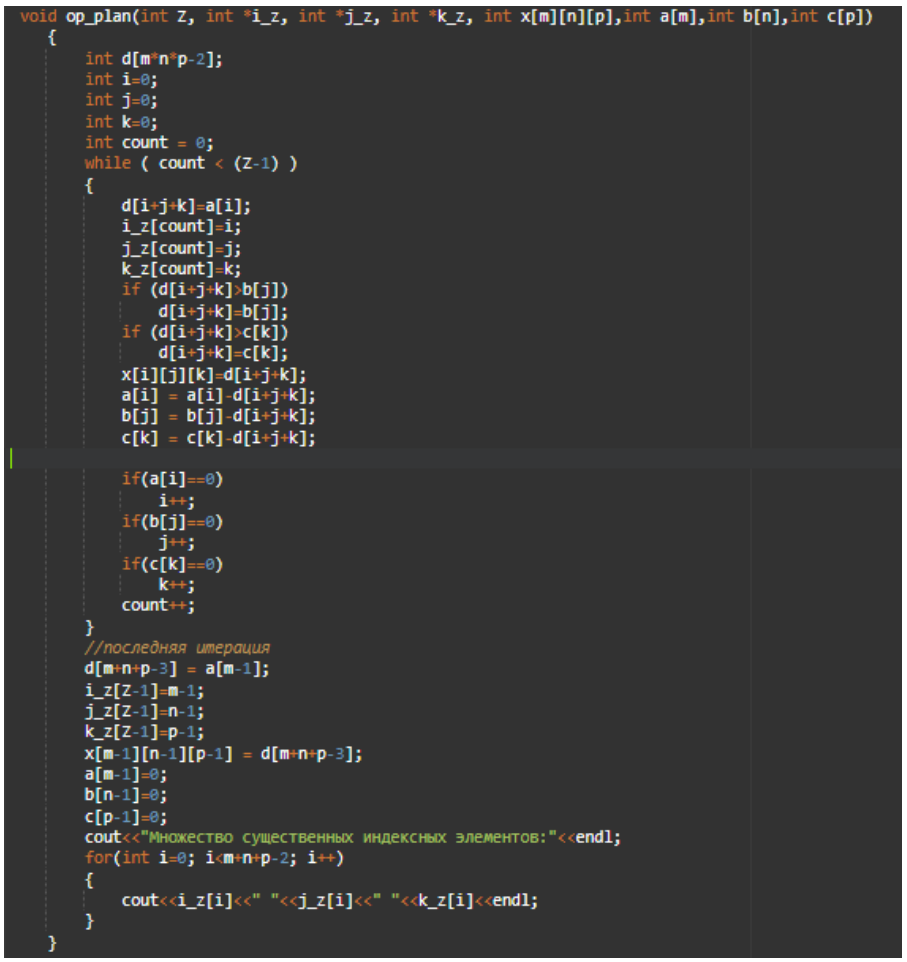
```
// объявление массивов и переменных
int m = 2; // количество поставщиков
int n = 3; // количество потребителей
int p = 2; // количество транспортных средств

// количество (запас) продукции, находящейся в распоряжении m-го центра производства
int[] a = {1, 8};
// количество продукции, необходимое n-му центру потребления
int[] b = {3, 2, 4};
// количество продукции, которое может быть перевезено всеми единицами транспорта p-го типа
int[] c = {2, 7};

// стоимость транспортировки единицы продукта из m-го центра производства
//в n-й центр потребления транспортным средством p-го типа.
int[][][] T = {
{{5, 32}, {12, 10}, {11, 0}},
{{4, 9}, {21, 5}, {13, 8}}
};
```

Рисунок 14 – Исходные данные

3.4 Реализация метода последовательного распределения

Метод последовательного распределения является одним из способов решения этой задачи. Он заключается в последовательном рассмотрении ячеек таблицы стоимости грузоперевозок и нахождении минимального элемента в каждой строке или столбце.

Составленная функция построения опорного плана методом последовательного распределения, который в лучшей степени помогает решить многопродуктовую транспортную задачу, метод PlanOP (рисунок 15).



```
void op_plan(int Z, int *i_z, int *j_z, int *k_z, int x[m][n][p],int a[m],int b[n],int c[p])
  {
      int d[m*n*p-2];
      int i=0;
      int j=0;
      int k=0;
      int count = 0;
      while ( count < (Z-1) )
      {
          d[i+j+k]=a[i];
          i_z[count]=i;
          j_z[count]=j;
          k_z[count]=k;
          if (d[i+j+k]>b[j])
              d[i+j+k]=b[j];
          if (d[i+j+k]>c[k])
              d[i+j+k]=c[k];
          x[i][j][k]=d[i+j+k];
          a[i] = a[i]-d[i+j+k];
          b[j] = b[j]-d[i+j+k];
          c[k] = c[k]-d[i+j+k];

          if(a[i]==0)
              i++;
          if(b[j]==0)
              j++;
          if(c[k]==0)
              k++;
          count++;
      }
      //последняя итерация
      d[m+n+p-3] = a[m-1];
      i_z[Z-1]=m-1;
      j_z[Z-1]=n-1;
      k_z[Z-1]=p-1;
      x[m-1][n-1][p-1] = d[m+n+p-3];
      a[m-1]=0;
      b[n-1]=0;
      c[p-1]=0;
      cout<<"Множество существенных индексных элементов:"<<endl;
      for(int i=0; i<m+n+p-2; i++)
      {
          cout<<i_z[i]<<" "<<j_z[i]<<" "<<k_z[i]<<endl;
      }
  }
```

Рисунок 15 – Метод PlanOP

Один из важнейших компонентов программного продукта, метод генерации опорного плана PlanOP выполняется линейно. Данное свойство сложности алгоритма позволяет ускорить программу, а также сделать линейной зависимость между временем выполнения и количеством данных при её использовании.

Следующий метод ShowLFunc является показательным и выводит данные связанные с результатом работы метода в строку консоли, представленной на рисунке 16.

```java
public static int L_foo(int[][][] x, int[][][] T) {
    int L = 0;
    for (int k = 0; k < x.length; k++) {
        for (int i = 0; i < x[0].length; i++) {
            for (int j = 0; j < x[0][0].length; j++) {
                L += x[i][j][k] * T[i][j][k];
            }
        }
    }
    System.out.println("Значение целевой функции при таком плане = " + L + "\n");
    return L;
}
```

Рисунок 16 – Метод ShowLFunc

Данная функция используется последовательно шаг за шагом до окончания алгоритма выводя результирующее значение $L(X)$.

3.5 Реализация метода потенциалов

Блок схема алгоритма потенциалов сгенерирована благодаря использованию инструмента проектирования UML-таблиц. Данный инструмент позволяет быстро и с высокой точностью изобразить все детали и секции, изображённые на рисунке 17.

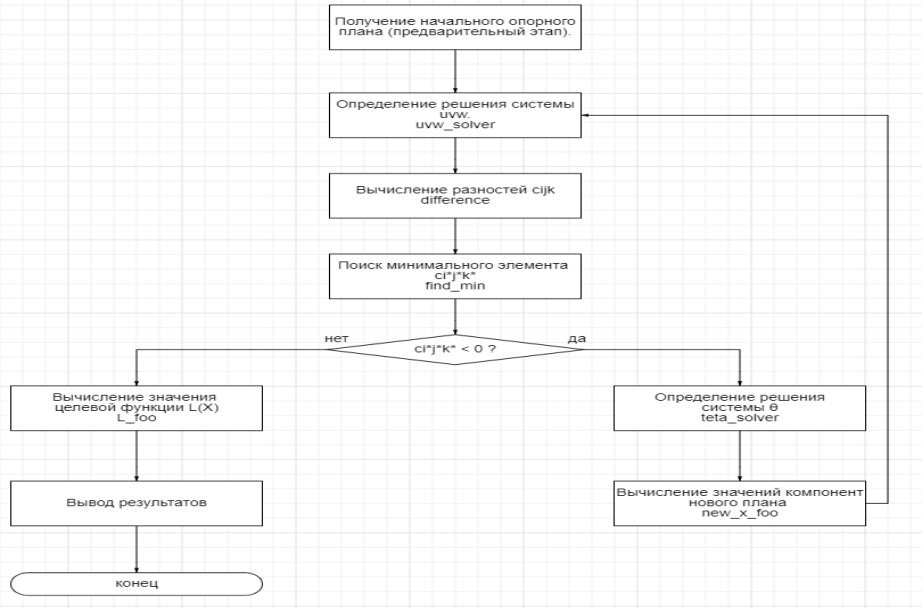


Рисунок 17 – UML-диаграмма алгоритма

Следующий метод SolvRolUVW позволяет оптимизировать заданную функцию (36). (рисунок 18)

```
public void uvw_solver(int[] u, int[] v, int[] w, int[][][] T, int[] i_z, int[] j_z, int[] k_z, int q, int[][][] Tc) {
    int count = 0;
    int line = 0;
    int col = 0;
    int height = 0;
    while (count < n + p + m - 2) {
        while (count < m) {
            line = i_z[count];
            col = j_z[count];
            height = k_z[count];
            if (q == 1) u[line] = T[line][col][height];
            else u[line] = Tc[line][col][height];
            count++;
        }

    while (count < p + m - 1) {
        line = i_z[count];
        col = j_z[count];
        height = k_z[count];
        if (q == 1)
            w[height] = T[line][col][height] - u[line];
        else
            w[height] = Tc[line][col][height] - u[line];
        count++;
    }
    line = i_z[count];
    col = j_z[count];
    height = k_z[count];
    if (q == 1)
        v[col] = T[line][col][height] - u[line] - w[height];
    else
        v[col] = Tc[line][col][height] - u[line] - w[height];
    count++;
}
}
```

Рисунок 18 – Метод SolvRolUVW

В последствии программа переходит к этапу определения наименьшей значимости из имеющихся полученных элементов с использованием метода MF представленной на рисунке 19.

```
public int find_min (int[][][]Tc, int min, int[]l, int[]c, int[]h)
{
  for (int i = 0; i < m; i++)
    {
      for (int j = 0; j < n; j++)
    {
      for (int k = 0; k < p; k++)
        {
          if (min > Tc[i][j][k])
        {
          min = Tc[i][j][k];
          l[0] = i;
          c[0] = j;
          h[0] = k;
        }
        }
    }
    }
  System.out.println ("Tc min = " + min);
  System.out.println ("ijk* = " + l[0] + " " + c[0] + " " + h[0]);
  return min;
}
```

Рисунок 19 – Метод MF

Методом Optima выводит данные на проверку, а также определяет оптимальность плана работая с индексами. (рисунок 20).

```
public void check_opt (int k_opt, int[][][]Tc)
{
  System.out.println ("Проверка на оптимальность");
  int[] i_z2 = new int[k_opt];
  int[] j_z2 = new int[k_opt];
  int[] k_z2 = new int[k_opt];
  int line2;
  int col2;
  int height2;
  System.out.print("План не является оптимальным, так как Tc[");
  for (int i = 0, index = 0; i < m; i++)
    {
      for (int j = 0; j < n; j++)
    {
      for (int k = 0; k < p; k++)
        {
          if (Tc[i][j][k] < 0)
        {
          i_z2[index] = i;
          line2 = i_z2[index];
          j_z2[index] = j;
          col2 = j_z2[index];
          k_z2[index] = k;
          height2 = k_z2[index];
          System.out.println (i + "][" + j + "][" + k + "] < 0");
          index++;
        }
        }
    }
    }
}
```

Рисунок 20 – Метод Optima

Далее начинает работу метод TSolution (рисунок 21), который в цикле фор проверяет лимит шагов и выводит результат решённой системы (38).

```
public int
teta_solver (int[]i_z, int[]j_z, int[]k_z, int[][][]teta, int teta0, int col2,
         int line2, int[][][]x)
{
  String[][][]sign = new String[m][n][p];
  int line;
  int col;
  int height;


  for (int i = 0; i < m + n + p - 2; i++)
    {

      line = i_z[i];

      col = j_z[i];

      height = k_z[i];

      if (i % 2 != 0)
    {

      sign[line][col][height] = "+";

      if (sign[line - 1][col][height].equals ("-"))
        {

          teta[line][col][height] = teta[line - 1][col][height] * (-1);

        }

    }

      if ((i % 2 == 0) && ((col == col2) || (line == line2)))
    {


      sign[line][col][height] = "-";

      teta[line][col][height] = -1;

      if (teta0 == 0)

        teta0 = teta[line][col][height] * x[line][col][height];

      if (teta0 < (teta[line][col][height] * x[line][col][height]))

        teta0 = teta[line][col][height] * x[line][col][height];
```

Рисунок 21 – Метод TSolution

Для формирования цикла пересчета используется трехмерный массив s[m][n][p] типа данных $string$. Данный массив заполняется с помощью функции TSolution (рисунок 21).

Метод XFinder (рисунок 22) используя многослойный цикл определяет новый x элемент оперируя выстой, колонкой и изменённым параметром Teta.

```
public void new_x_foo (int[][][]teta, int[][][]x, int teta0, int line2, int col2,
       int height2, int[][][]T, int[]i_z, int[]j_z, int[]k_z)
{
  int count = 0;
  for (int i = 0; i < m; i++)
    {
      for (int k = 0; k < p; k++)
    {
      for (int j = 0; j < n; j++)
        {
          if (teta[i][j][k] != 0)
        {
          x[i][j][k] = x[i][j][k] - teta[i][j][k] * teta0;
        }
          if ((i == line2) && (j == col2) && (k == height2))
        {
          x[i][j][k] = x[i][j][k] - teta0;
          i_z[0] = line2;
          j_z[0] = col2;
          k_z[0] = height2;
        }
          System.out.print (x[i][j][k]);
          count++;
        }
      System.out.println ();
      if (k != p - 1)
        System.out.print (" ");
    }
    }
}
```

Рисунок 22 – Метод XFinder

Итерационный процесс, упомянутый в оригинальном тексте, предполагает последовательное улучшение плана до достижения оптимального результата. Каждая итерация может изменять один или несколько параметров плана, и после каждой итерации происходит проверка на оптимальность плана с помощью функции Optima.

Если функция Optima показывает, что текущий план является оптимальным, то программа использует функцию ShowLFunc для вычисления значения целевой функции $L(X)$. Целевая функция $L(X)$ может использоваться

для оценки качества плана и принятия решения о его дальнейшем улучшении или принятии в качестве окончательного результата.

После вычисления значения целевой функции $L(X)$ оптимальный план выводится в консоль, а программа завершает свою работу.

Для проверки работы программы на примере, необходимо использовать тестовые данные, которые были использованы для тестирования алгоритма в предыдущем решении. Это позволит убедиться в корректности работы программы и правильности вычисления оптимального плана. На рисунке 23 представлена установка базовых параметров и вывод изначальных матриц.

```
Исходные данные:
Количество поставщиков = 2
Количество потребителей = 3
Количество транспортных средств = 2
Таблица стоимости грузоперевозок T:
5 12 11
 32 10 0
4 21 13
 9 5 8
Множество существенных индексных элементов:
0 0 0
1 0 0
1 0 1
1 1 1
1 2 1
Опорный план:
1 0 0
 0 0 0
1 0 0
 1 2 4

Оптимизация плана грузоперевозок методом потенциалов. Итерация (1):
Значение целевой функции при таком плане = 60

Решение системы uvw
u[0] = 5
u[1] = 4
w[0] = 0
w[1] = 5
v[0] = 0
v[1] = -4
v[2] = -1
Вычисление разностей
Tc[0][0][0] = 0
Tc[0][0][1] = 22
Tc[0][1][0] = 11
Tc[0][1][1] = 4
Tc[0][2][0] = 7
Tc[0][2][1] = -9
Tc[1][0][0] = 0
Tc[1][0][1] = 0
Tc[1][1][0] = 21
Tc[1][1][1] = 0
Tc[1][2][0] = 10
Tc[1][2][1] = 0
Проверка на оптимальность
План не является оптимальным, так как Tc[0][2][1] < 0
Tc min = -9
i*j*k* = 0 2 1
Новый план:
000
 001
200
 123
```

Рисунок 23 – Вывод подготовительного этапа работы

После проведение расчётных операций в основном телецикла, программа переходит к выведению всех итоговых значений на экран, представленных на рисунке 24 в последствии выходя из цикла.

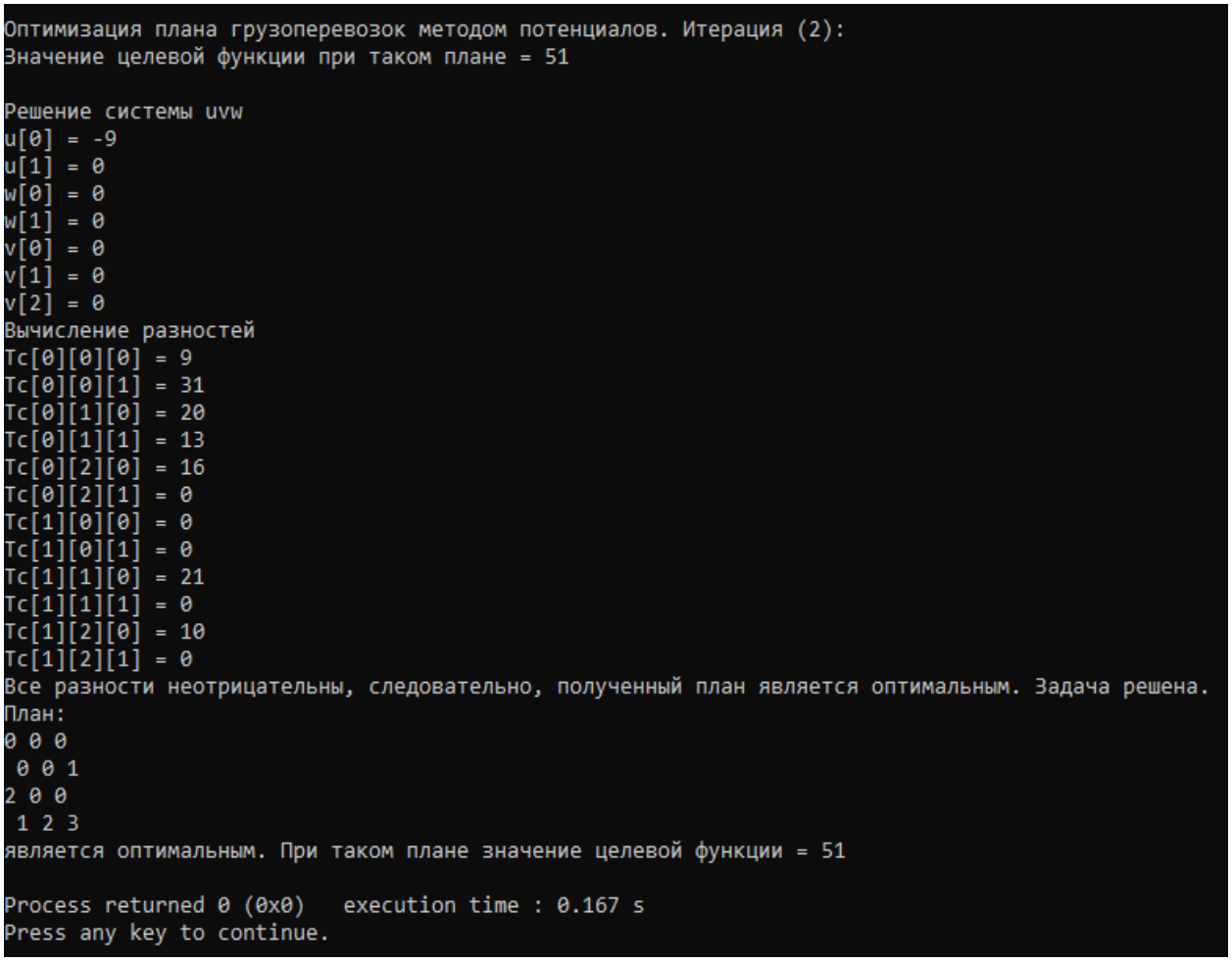


Рисунок 24 – Финальный шаг работы программы

В этом разделе было создано программное обеспечение, используя определенные алгоритмы и структуры данных. Также проведены тесты для проверки правильности работы программы.

Проведя успешное тестирование, можно с уверенностью сказать, что программа готова к использованию в транспортной компании. В конечном итоге получилось достичь целей и задач, которые были поставлены в этом разделе, что является важным успехом в разработке программного обеспечения.

3.6 Решение трипланарной транспортной задачи в Microsoft Excel

Excel – это универсальный помощник в офисном деле от Microsoft. Он содержит множество инструментов для работы с таблицами, графиками, диаграммами, и текстовыми документами. Кроме того, одной из важных функций программы является решение логистических задач, таких как транспортная задача, связанная с перевозкой товаров из одной точки в другую. Для того чтобы определить оптимальный план перевозок учитываются ограничения по времени и стоимости. С помощью Экселя можно решать задачи создания и заполнения таблиц, строить графики и прогнозировать результаты. Легко вычислить затраты на логистику и прогнозировать будущие расходы, используя различные модели прогнозирования.

Конечно, решение транспортных задач с помощью программного инструмента может вызвать определенные трудности, включая ограниченность функциональности программы и сложность решения задач с большим количеством переменных. В этом случае могут быть полезны другие специализированные программные продукты с более точными алгоритмами.

Перед тем, как устанавливать методы решения, требуется заполнить таблицу данными, представленными на рисунке 25.

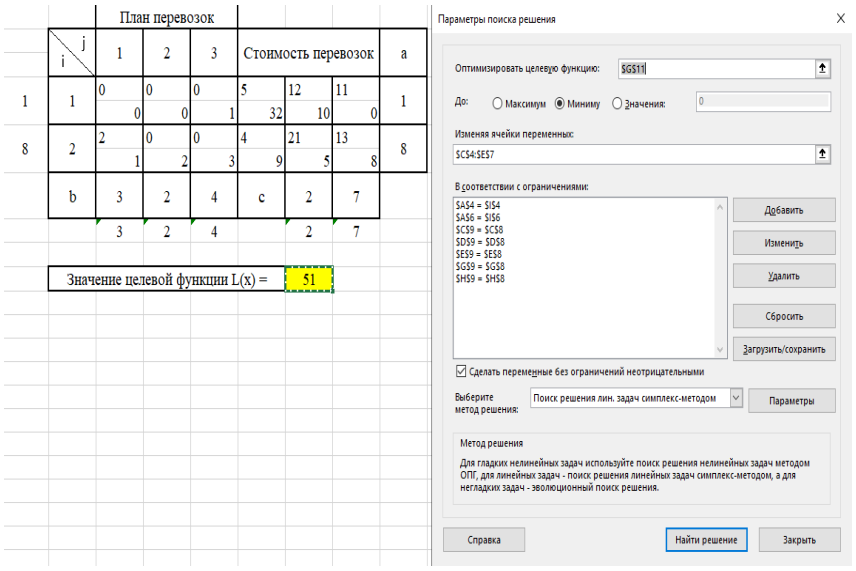


Рисунок 25 – Метод вычисления «Поиск решения»

Когда используется функция "Поиск решения", требуется указать ячейку, в которой нужно получить наименьшее значение функции $L(x)$, используя формулу =SUMPRODUCT(C4:E7,F4:H7), которая вычисляет стоимость перевозок учитываю две матрицы, представленные выше.

По нажатию на кнопку "Найти решение", симплекс-метод начнет подбирать значения каждой ячейки, чтобы получить наименьшее значение функции L(x). Требуется обрати внимание, что этот метод не работает быстро для больших трипланарных транспортных задач, где нужно вводить все данные вручную. Этот процесс может занять некоторое время, к тому же даже маленькая ошибка в ячейке может повлиять на всю задачу.

3.7 Анализ эффективности программного продукта

В современном мире компьютеры охватывают все сферы жизни. Все больше используются электронные устройства для различных задач. И, конечно же, не обойдется без программирования. В настоящее время существует множество методов для создания алгоритмов, которые способны решать различные задачи. Однако, чем более сложным становится задание, тем более трудоемким становится процесс его выполнения и приходится упрощать сложность алгоритмов, путём написания более оптимального кода под конкретную задачу.

Алгоритм может иметь различную сложность, которая измеряется в количестве времени, необходимом для его выполнения, или объеме памяти, которую он занимает. В связи с технологическим развитием, объем памяти, которую занимает алгоритм, перестал быть критическим ресурсом. Поэтому анализ сложности алгоритма обычно ориентирован на скорость его работы.

Время выполнения алгоритма может отличаться в зависимости от устройства, на котором он запущен. Однако сложность алгоритма можно выразить в элементарных шагах – количестве действий, необходимых для его

выполнения. Количество шагов, которые необходимы для выполнения любого алгоритма, остается неизменным независимо от устройства, на котором алгоритм запущен. Эту идею можно представить в виде Big O-нотации.

Один из способов сократить затраты времени и ресурсов компании – использовать в основе программы простые алгоритмы получения результата, позволяющие найти наилучший путь доставки товаров быстрее, при наличии большого набора данных. Данная программа позволит сэкономить средства компании и ускорить свою работу повысив свою эффективность. Также данную задачу позволяет решить модульность проекта, в значительной степени сокращающая время работы приложения.

Программный продукт, реализованный в данной ВКР, содержит в основе алгоритма линейную сложность O(n), представленную синим графиком на рисунке 26.

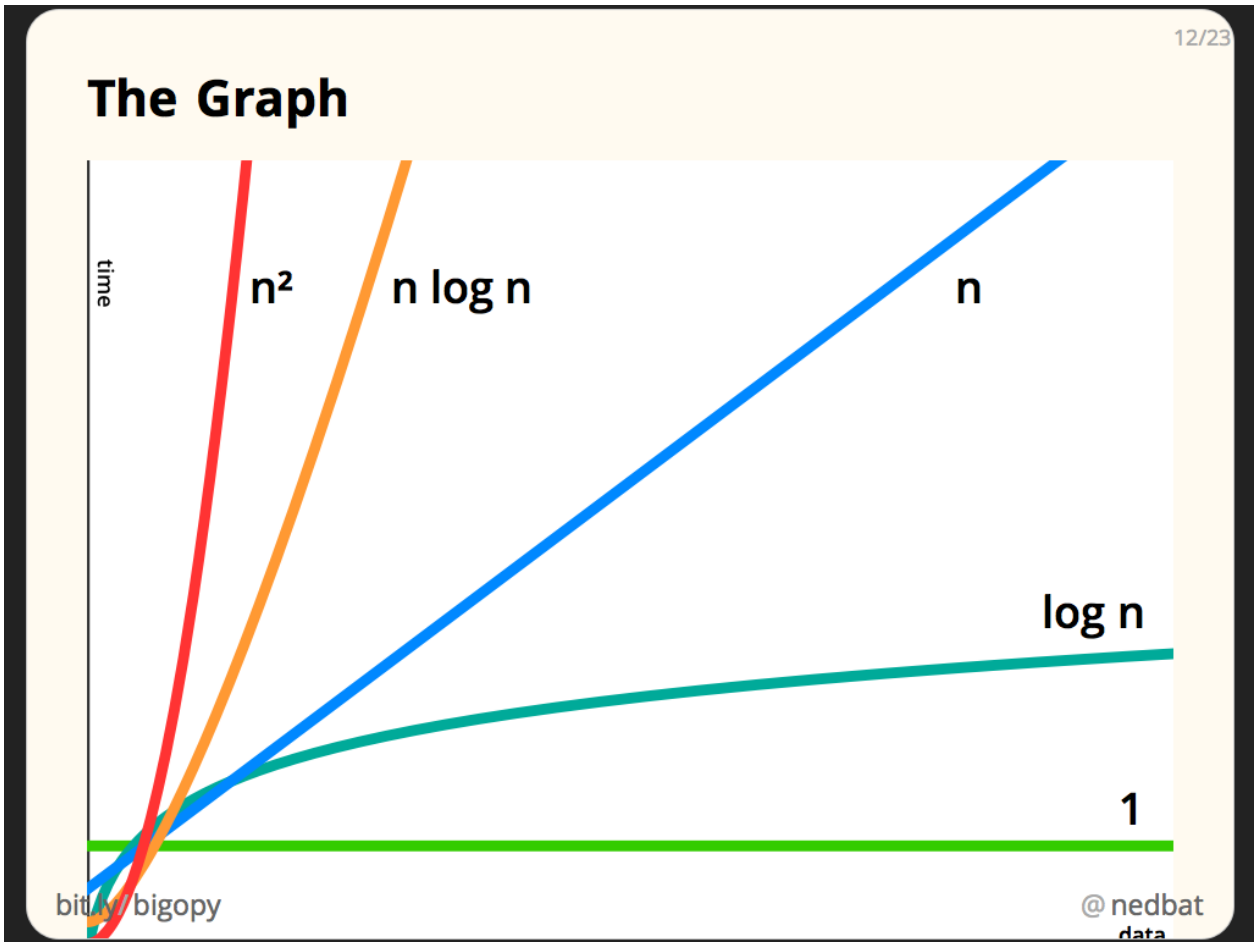


Рисунок 26 – Метод вычисления «Поиск решения»

С другой стороны, стандартная реализация просчёта наименьшего пути затрат Excel, представленная красным графиком на рисунке 25, через построение сложных таблиц и объединения нескольких формул в одну, имеет квадратичную сложность, связанную с трудностью изменения данных на новые, при условии реальной задачи в которой может присутствовать до миллиона различных значений, что делает данное решение не оптимальным по множеству факторов, таких как:

- квадратичные алгоритмы выполняют большее количество операций при увеличении размера входных данных. Линейные алгоритмы, напротив, выполняют операции пропорционально размеру входных данных. Поэтому алгоритмы с линейной сложностью лучше масштабируются для обработки больших объемов данных;
- квадратичные алгоритмы требуют больше памяти для хранения и обработки данных, что делает их менее эффективными и увеличивает расходы на хранение информации. Линейные алгоритмы обрабатывают данные последовательно, что позволяет сократить расходы на память;
- квадратичные алгоритмы, как правило, сложнее отлаживать и тестировать из-за большого количества итераций в циклах. Линейные алгоритмы не имеют этой проблемы, поскольку они выполняются только один раз для каждого элемента.

В отличии от алгоритма линейной сложности, который имеет:

- быстрое выполнение: линейные алгоритмы имеют время выполнения, которое зависит от количества элементов, которые нужно обработать. С увеличением количества элементов время выполнения увеличивается линейно;
- интуитивная понятность: линейные алгоритмы обычно имеют интуитивно понятную структуру, что упрощает их понимание и использование;

– низкая вероятность ошибок: линейные алгоритмы обычно имеют меньше шансов на ошибки, так как они проходят по всем элементам последовательно и не имеют сложных ветвлений.

С помощью тестов были проведены экспериментальные запуски программы с различным количеством параметров в соотношении с изменением времени при работе, таблица временных затрат представлена Таблице 1.

Таблица 1 – Зависимость количества совокупных элементов программы

Сложность	Размер (Кол-во элементов)					
	10	900	5700	10050	30090	100500
O(n), с	0,00001	0,0009	0,0057	0,01005	0,03009	0,1005
$O(n^2)$, с	0,0001	0,81	32,49	101	905	10010

По таблице можно сделать вывод, что постоянные расчёты наиболее оптимальны с использованием линейной сложности алгоритма.

К тому же, написание программы на высокоуровневом языке, таком, как Java, позволяет быстро расширить его функционал и добавлять новые данные.

Java – один из самых используемых языков программирования в мире. Он прост в освоении и может быть использован для создания программ с различными целями – от мобильных приложений до серверных приложений.

Кроме того, данный алгоритм легко обновляется и дополняется. Это означает, что при необходимости можно быстро добавлять новый функционал, не переписывая весь код, что значительно экономит время и усилия разработчиков, а также экономических средств компании.

Выводы по 3 разделу.

В третьем разделе нашей работы были разработаны методы на основе сложного алгоритма для решения задач оптимизации и линейного программирования. Был рассмотрен классический метод симплекс, методы на основе симплекс-таблиц, методы внутренней точки, а также методы перебора

и прямой суммы. Для каждого метода были подробно описаны принципы работы и алгоритмы.

В итоге, алгоритм линейной сложности, написанный на Java, является отличным способом сократить временные и ресурсные траты компании. Он обладает высокой скоростью работы, прост в обновлении и расширении, а также обладает проверкой на валидность данных, что предотвращает появление ошибок в работе программы. Использование данного алгоритма является актуальным и эффективным решением в IT-сфере программирования и алгоритмических задач.

Заключение

В выпускной квалификационной работе проанализированы теоретические сведения линейного программирования.

В рамках проделанной работы разработан и протестирован на производительность программный продукт для решения многопродуктовой транспортной задачи.

Для создания программного продукта осуществлена декомпозиция многопродуктовой транспортной задачи на блоки, по которым составлены UML-диаграммы классов, описывающих алгоритм работы программы.

Для реализации текущего веб-приложения были выбраны наиболее актуальные и подходящие инструменты. При выборе архитектуры приложения, выбор был сделан в пользу объектно-ориентированной архитектуры. Данная архитектура выбрана в связи с тем, что она обладает простой возможностью масштабирования, гибкостью при проектировании различных модулей и способностью повторно использовать различные объекты. Основным инструментом разработки послужил язык Java и его сторонние библиотеки позволившие сократить сложность алгоритма программы.

В ВКР проведен анализ эффективности алгоритма программного продукта, определены положительные характеристики, снижающие временные и экономические затраты программы на обработку большого количества реальных массивов данных.

Разработанный программный продукт, показал высокую эффективность при работе с большим набором переменных, по сравнению со стандартным решением подсчёта наименьших издержек в программе Excel, которая требует большого количества времени на обработку данных и сложную структуру добавления новых блоков в таблицу.

Список используемой литературы

1. Афраймович Л.Г. Минимизация затрат при распределении однородного ресурса в иерархических системах с двусторонними ограничениями // КоГраф 2002. Материалы докладов всероссийской конференции. – Нижний Новгород. 2002. С. 100-110.

2. Булавский В.А., Звягина Р.А., Яковлева М.А. Численные методы линейного программирования. Специальные задачи. – М.: Наука. 1977. С. 50-55.

3. Афраймович Л.Г. Метод решения целочисленных многоиндексных транспортных задач с декомпозиционной структурой // Доклады Одесского семинара по дискретной математике. № 11. 2011. 210 с.

4. Раскин Л.Г., Кириченко И.О. Многоиндексные задачи линейного программирования. – М.: Радио и связь. 1982. С. 12-17.

5. Рублев В.С., Смирнов А.В. Задача целочисленного сбалансирования трехмерной матрицы и алгоритмы ее решения // Моделирование и анализ информационных систем. 2010. Т. 17. № 2. С. 57-78.

6. Рублев В.С., Смирнов А.В. NP-полнота задачи сбалансирования трехмерной матрицы // Доклады Академии наук. 2010. Т. 435. № 3. С. 50–60.

7. Рублев В.С., Чаплыгина Н.Б. Выбор критерия оптимизации в задаче о равномерном назначении // Дискретная математика. 2005. Т. 17. Вып. 4. 29 с.

8. Сергеев С.И. Новые нижние границы для трипланарной задачи назначения. Использование классической модели // Автоматика и телемеханика. 2008. № 12. С. 105-120.

9. Сергеев С.И. Улучшенные нижние границы для решения квадратичной задачи назначения // Автоматика и телемеханика. 2004. № 11. С. 100-110.

10. Смирнов А.В. Задача целочисленного сбалансирования трехмерной матрицы и сетевая модель // Моделирование и анализ информационных систем. 2009. Т. 16. № 3. 100 с.

11. Схрейвер А. Теория линейного и целочисленного программирования: в 2 т. – М.: Мир. 1991. С. 117-123.

12. Триус Е.Б. Задачи математического программирования транспортного типа. М.: Советское радио. 1967. С. 228 с.

13. Филлипс Д., Гарсиа-Диас А. Методы анализа сетей. – М.: Мир. 1984. С. 99-110.

14. Финкельштейн Ю.Ю. Приближенные методы и прикладные задачи дискретного программирования. М.: Наука. 1976. 122 с.

15. Кравцов М.К., Кравцов В.М., Лукшин Е.В. О нецелочисленных вершинах многогранника трехиндексной аксиальной задачи о назначениях // Дискретная математика. 2001. Т. 13. Вып. 2. С. 57-64.

16. Кравцов М.К., Лукшин Е.В. О нецелочисленных вершинах многогранника трехиндексной аксиальной транспортной задачи // Автоматика и телемеханика. 2004. № 3. С. 102–120.

17. Кропанов В.А., Рублев В.С. Равномерное назначение работ минимальной стоимости // Дискретная математика. 2001. Т. 13. Вып. 4. 2001. 57 с.

18. Литвак Б.Г., Раппопорт А.М. Задачи линейного программирования, допускающие сетевую постановку // Экономика и математические методы. 1970. Т. 6. Вып. 4. С. 100-110.

19. Малашенко Ю. Е., Новикова Н. М. Потоковые задачи анализа уязвимости многопродуктовых сетей. – М.: ВЦ АН СССР. 1989. С. 101-110.

20. Меламед И.И., Сигал И.Х. Вычислительное исследование алгоритмов решения бикритериальных задач дискретного программирования // Журнал вычислительной математики и математической физики. 2000. Т. 4. № 11. 30 с.

21. Ahuja R.K., Magnati T.L., Orlin J.B. Network flows: theory, algorithms, and applications. Prentice Hall. 1993. С. 122-128.

22. Alighanbari M., How J.P. Cooperative task assignment of unmanned aerial vehicles in adversarial environments // Proceedings of the American Control Conference. 2005. V. 7. P. 4661–4666. C. 32-38.

23. Amaldi E., Pfetsch M.E., Leslie E.T. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs // Mathematical Programming, 2003. V. 95. N 3. 68 c.

24. Arbib C., Pacciarelli D., Smriglio S. A. A three-dimensional matching model for perishable production scheduling // Discrete Applied Mathematics. 1999. V. 92. C. 79-90.

25. Balas E., Saltzman M.J. An algorithm for the three-index assignment problem // Operations Research. 1991. V. 39. C. 123–134.