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

Кафедра «Прикладная математика и информатика»
(наименование)

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

Компьютерные технологии и математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Программная реализация решения трехиндексной транспортной задачи декомпозиционным методом

Обучающийся А. А. Ахвердян _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Выпускная квалификационная работа посвящена программной реализации решения трехиндексной транспортной задачи.

Ключевые слова: трехиндексная, транспортная задача, программная реализация, декомпозиционный метод, симплекс-метод.

Решение транспортных задач является важным современным направлением исследований, результаты которого востребованы во многих областях практической деятельности.

Применение программных средств для решения трехиндексных транспортных задач во многих случаях обеспечивает оптимальные логистические решения, что потенциально может позволить сэкономить на транспортных расходах.

Объектом исследования данной бакалаврской работы является трехиндексная транспортная задача.

Предметом исследования является декомпозиционный метод решения трехиндексной транспортной задачи.

Цель работы – разработка программной реализации решения трехиндексной транспортной задачи декомпозиционным методом.

В ходе выполнения выпускной работы было проведено исследование предметной области, проектирование программного обеспечения, разработка и тестирование программного обеспечения.

Выпускная квалификационная работа представлена на 41 странице, включает 16 иллюстраций, 2 таблицы, список используемой литературы, состоящий из 20 источников.

Abstract

The graduate qualification work is devoted to the software implementation of the solution of the three-index transport problem.

Key words: three-index, transport problem, software implementation, decomposition method, simplex method.

The solution of transport problems is an important modern direction of research, the results of which are in demand in many areas of practical activity.

Application of software tools for solving three-index transport problems in many cases provides optimal logistic solutions, which can potentially save on transport costs.

The object of research of this bachelor's thesis is a three-index transport problem.

The subject of the research is the software implementation of the solution of the three-index transport problem.

The purpose of the work is to develop a software implementation of the solution of the three-index transport problem.

In the course of the graduate work was carried out research of the subject area, software design, software development and testing.

The graduate qualification work is presented on 41 pages, includes 16 illustrations, 2 tables, list of used literature consisting of 20 sources.

Содержание

Введение

Проблема программной реализации решения трехиндексных транспортных задач является важным современным направлением исследований, результаты которого востребованы во многих областях практической деятельности.

«Примерами транспортных задач являются задачи распределения ресурсов в иерархических системах: задача объемно-календарного планирования, задача формирования портфеля заказов, задача переработки газового конденсата, задача распределения мощностей каналов передачи данных, транспортная задача с промежуточными пунктами. [9]. Многоиндексные задачи транспортного типа возникают также в области статистики и в смежной области защиты статистических данных [17], [11] в задаче сбалансирования многоиндексных матриц» [1].

Это подтверждает актуальность направления многоиндексных транспортных задач в целом.

Декомпозиционный метод решения транспортных задач эффективен для крупных транспортных задач с большим количеством узлов и дуг, которые было бы сложно решить с помощью других методов. Этот метод хорошо подходит для распределенных сред, где вычислительная нагрузка может быть разделена между несколькими процессорами или компьютерами. Также декомпозиционным методом можно эффективно решать транспортные задачи с нерегулярными транспортными сетями, в которых присутствуют разрывы и ограничения на маршрутах.

Этим определяется актуальность работ в области декомпозиционных методов решения транспортных задач.

В соответствии с темой работы «Программная реализация решения трехиндексной транспортной задачи декомпозиционным методом» объектом исследования является трёхиндексная транспортная задача.

Предмет исследования – декомпозиционный метод решения трёхиндексной транспортной задачи.

Цель работы – разработка программной реализации решения трехиндексной транспортной задачи декомпозиционным методом.

Задачи работы:

Выполнить описание трехиндексной транспортной задачи.

Провести анализ существующих решений трехиндексной транспортной задачи.

Разработать модель программной реализации решения трехиндексной транспортной задачи декомпозиционным методом.

Разработать программу реализующую решение трехиндексной транспортной задачи декомпозиционным методом.

Провести вычислительный эксперимент.

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

В первом разделе «Анализ трехиндексной транспортной задачи» выполнено описание трехиндексной транспортной задачи, исследованы существующие методы решения транспортных задач на основании чего сформулированы требования к программной реализации декомпозиционного метода решения трехиндексной транспортной задачи.

Во втором разделе «Программная реализация решения трехиндексной транспортной задачи» выполнена постановка задачи и выбор средств разработки, разработаны алгоритмы решения задачи, описана реализация программных модулей.

В третьем разделе «Тестирование программной реализации решения трехиндексной транспортной задачи» выполнено планирование и проведение вычислительных экспериментов, в результате которых подтверждена функциональность разработанного программного решения.

1 Анализ трехиндексной транспортной задачи

1.1 Описание трехиндексной транспортной задачи

«Линейное программирование – математическая дисциплина, посвященная теории и методам решения экстремальных задач на множествах n-мерного пространства, задаваемых системами линейными уравнений и неравенств» [5].

Транспортная задача линейного программирования получила в настоящее время широкое распространение в теоретических разработках и практическом применении на транспорте и в промышленности. Особенно большое значение она имеет в деле рационализации постановок важнейших видов промышленной и сельскохозяйственной продукции, а также оптимального планирования грузопотоков и работы различных видов транспорта.

Кроме того, к задачам транспортного типа сводятся многие другие задачи линейного программирования - задачи о назначениях, сетевые, календарного планирования.

Однородный груз сосредоточен у m поставщиков в объемах a_1, a_2, ,..., a_m . Данный груз необходимо доставить n потребителям в объемах b_1, ,b_2 ,...,b_n . Известны c_{IJ} , j=1,2,…,n – стоимости перевозки единицы груза от каждого I-го поставщика каждому j-му потребителю. Требуется составить такой план перевозок, при котором запросы всех потребителей полностью удовлетворены и суммарные затраты на перевозку всех грузов минимальны.

На рисунке 1 изображена транспортная модель в виде сети с m исходными пунктами и n пунктами назначения. Исходным пунктам и пунктам назначения соответствуют вершины сети.

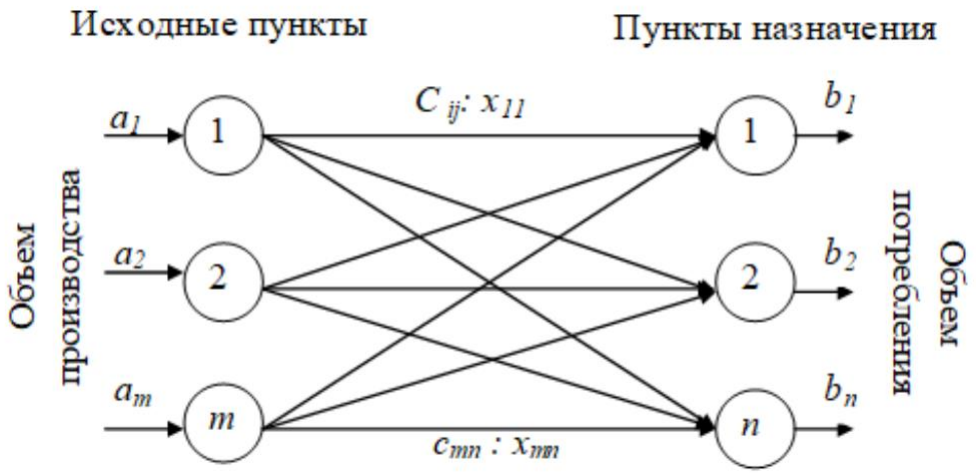

Рисунок 1 – Модель двухиндексной транспортной задачи

В транспортных задачах под поставщиками и потребителями понимаются различные промышленные и сельскохозяйственные предприятия, заводы, фабрики, склады, магазины и т.д. Однородными считаются грузы, которые могут быть перевезены одним видом транспорта. Под стоимостью перевозок понимаются тарифы, расстояния, время, расход топлива и т.п.

В транспортной задаче предполагается, что суммарные запасы поставщиков равны суммарным запросам потребителей. Такая задача называется задачей с правильным балансом, а ее модель – закрытой. Если же это равенство не выполняется, то задача называется задачей с неправильным балансом, а ее модель – открытой.

Имеет место широкий класс практических задач, которые предусматривают использование многоиндексных транспортных задач линейного программирования. Примерами таких задач являются задачи распределения ресурсов в иерархических системах. Многоиндексные задачи транспортного типа возникают также в области статистики и в смежной области защиты статистических данных.

«Многоиндексные задачи о назначениях (подкласс многоиндексных транспортных задач целочисленного линейного программирования) возникают в теории расписаний при планировании изготовления скоропортящейся продукции, при планировании прохождения практики студентами, при планировании учебы клинических ординаторов по отделениям, при составлении расписания занятий, при планировании спортивных матчей и др. [12]; в области технического анализа данных при сопровождении объектов в многосенсорных системах [20]; в военной области при назначении военной техники на цели [17]. Известны результаты, посвященные исследованию вопросов сводимости задач линейного и целочисленного линейного программирования к многоиндексным транспортным задачам [15], что также расширяет область применения методов решения многоиндексных задач» [1].

«Многоиндексные задачи линейного программирования транспортного типа относятся к классу задач линейного программирования, который согласно [11] является полиномиально разрешимым. Таким образом, для решения многоиндексных задач линейного программирования транспортного типа могут быть применены общие методы исследований задач линейного программирования – симплекс метод [13], алгоритм Кармаркара [16].

Существует ряд работ, посвященных непосредственно методам решения многоиндексных задач линейного программирования транспортного типа. Наиболее изученным является класс двухиндексных задач. В многоиндексном случае (число индексов не менее трех) наибольшее внимание уделено двум классам задач: многоиндексные аксиальные задачи и многоиндексные планарные задачи» [1].

«Особый интерес представляет решение многоиндексных задач целочисленного линейного программирования транспортного типа, относящихся к классу задач целочисленного линейного программирования» [8].

Примером полиномиально разрешимого класса задач N-кратного целочисленного программирования является класс трехиндексных планарных задач, в котором два из трех индексов принимают фиксированное количество значений. При отсутствии дополнительных ограничений на параметры для решения многоиндексных задач целочисленного линейного программирования транспортного типа применимы лишь экспоненциальные по верхней оценке вычислительной сложности общие методы целочисленного линейного программирования, например, метод динамического программирования, метод ветвей и границ, метод отсечения Гомори» [1].

«Среди целочисленных многоиндексных транспортных задач наиболее изученным является класс многоиндексных задач о назначениях. Особое внимание уделяется двум классам многоиндексных задач о назначениях: класс аксиальных задач о назначениях и класс планарных задач о назначения» [1].

Одним из наиболее распространенных вариантов постановки трехиндексной транспортной задачи является следующий.

Имеется m поставщиков (A_1, $A_{2, \ldots,}$ A_m), n потребителей (B_1, $B_{2, \ldots,}$ B_n) и k различных продуктов (C_1, $C_{2, \ \ldots,}$ C_k). Объём поставок продукта C_t поставщиком A_i обозначим через a_{it}, объём потребности в продукте C_t у потребителя B_j – через b_{jt}, общий объём перевозок от поставщика A_i к потребителю B_j через c_{ij}, стоимость перевозки единицы продукта C_t от поставщика A_i к потребителю B_j через d_{ijt}. Задача минимизации общей стоимости перевозки запишется так:

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{t=1}^{k} d_{ijt}x_{ijt} \rightarrow min.$$

При этом для каждой пары $(i, t) \in N_m \times N_k$ должно выполняться

$$\sum_{j=1}^{m} x_{ijt} = a_{it.}$$

Для каждой пары $(j, t) \in N_n \times N_k$ должно выполняться

$$\sum_{i=1}^{m} x_{ijt} = b_{jt.}$$

Для каждой пары $(i, j) \in N_m \times N_n$ должно выполняться

$$\sum_{t=1}^{k} x_{ijt} = c_{ij.}$$

Для каждой тройки $(i, j, t) \in N_m \times N_n \times N_k$ должно выполняться

$$x_{ijt} \geq 0.$$

На исходные данные накладывается ряд ограничений.

Условие баланса поставок и потребления означает, что сумма поставки по каждому продукту должна быть равна суммарному потреблению этого продукта.

Условие баланса по объемам перевозок означает, что пропускная способность коммуникации (i, j) задана и фиксирована для совокупности продуктов.

Условия баланса:

$$\sum_{i=1}^{m} a_{it} = \sum_{j=1}^{n} b_{jt}, t \in N_t,$$

$$\sum_{t=1}^{k} a_{it} = \sum_{j=1}^{n} c_{ij}, i \in N_i,$$

$$\sum_{t=1}^{k} b_{jt} = \sum_{i=1}^{n} c_{ij}, j \in N_j.$$

В такой постановке трехиндексная транспортная задача относится к виду аксиальных задач о назначениях.

1.2 Методы решения транспортных задач

Классическим методом решения линейных транспортных задач является симлекс-метод – наиболее универсальный метод решения задач линейного программирования.

«Симплекс-метод позволяет эффективно найти оптимальное решение, избегая простой перебор всех возможных угловых точек. Основной принцип метода: вычисления начинаются с какого-то «стартового» базисного решения, а затем ведется поиск решений, «улучшающих» значение целевой функции. Это возможно только в том случае, если возрастание какой-то переменной приведет к увеличению значения функционала» [5].

«Необходимые условия для применения симплекс-метода:

– задача должна иметь каноническую форму;

– у задачи должен быть явно выделенный базис.

Базисный вектор имеет размерность (m+1), где m – количество уравнений в системе ограничений» [5].

Алгоритм симплекс-метода показан на рисунке 2.

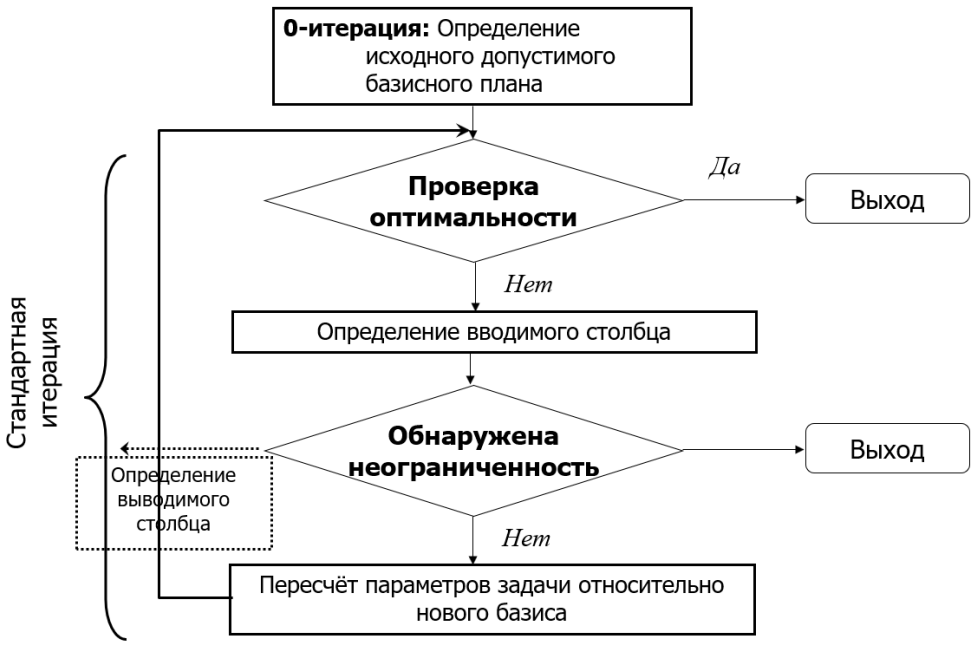

Рисунок 2 – Алгоритм симплекс-метода

Алгоритм симплекс-метода (рисунок 2) состоит из следующих шагов.

«Выбираем переменную, которую будем вводить в базис. Это делается в соответствии с указанным ранее принципом: мы должны выбрать переменную, возрастание которой приведет к росту функционала. Выбор происходит по следующему правилу: если задача на минимум – выбираем максимальный положительный элемент в последней строке, если задача на максимум – выбираем минимальный отрицательный.

Такой выбор соответствует принципу: если задача на минимум, то чем большее число вычитаем – тем быстрее убывает функционал; для максимума наоборот – чем большее число добавляем, тем быстрее функционал растет » [5].

«Выбираем переменную, которую будем вводить в базис. Для этого нужно определить, какая из базисных переменных быстрее всего обратится в нуль при росте новой базисной переменной. Алгебраически это делается так:

вектор правых частей почленно делится на ведущий столбец, среди полученных значений выбирают минимальное положительное (отрицательные и нулевые ответы не рассматривают).

Ищем элемент, стоящий на пересечении ведущих строки и столбца. Такой элемент называется ведущим элементом» [5].

«Вместо исключаемой переменной в первом столбце (с названиями базисных переменных) записываем название переменной, которую мы вводим в базис.

Далее начинается процесс вычисления нового базисного решения. Он происходит с помощью метода Жордана-Гаусса.

Новая Ведущая строка = Старая ведущая строка / Ведущий элемент

Новая строка = Новая строка – Коэффициент строки в ведущем столбце * Новая Ведущая строка.

После этого проверяем условие оптимальности. Если полученное решение неоптимально – повторяем весь процесс снова.

Условие оптимальности полученного решения – если задача на максимум – в строке функционала нет отрицательных коэффициентов (т.е. при любом изменении переменных значение итогового функционала расти не будет), если задача на минимум – в строке функционала нет положительных коэффициентов (т.е. при любом изменении переменных значение итогового функционала уменьшаться не будет)» [5].

Метод потенциалов для решения трехиндексной транспортной задачи [4] - это итеративный метод, состоящий в последовательном повторении шести шагов:

На первом шаге выполняется инициализация, в ходе которой необходимо найти начальное допустимое решение транспортной задачи с помощью метода северо-западного угла или любого другого эвристического метода и присвоить потенциалам поставщиков и потребителей нулевые значения.

На втором шаге делается расчет оценочной стоимости – для каждой ячейки трехмерной транспортной таблицы рассчитывается соответствующая стоимость перевозки одной единицы товара от поставщика к потребителю.

Затем выполняется поиск улучшающего цикла, такого замкнутого цикла, состоящего из непустых ячеек, для которых оценочная стоимость отрицательна.

На найденном цикле выполняется расчёт минимального потока – минимальное количество единиц товара, которые можно переместить.

На следующем шаге выполняют обновление решения, которое состоит в вычитании минимального потока из потоков по ребрам, направленным в положительном направлении и добавлении минимального потока к ребрам направленным в отрицательном направлении.

На завершающем шестом шаге выполняется проверка оптимальности. Если нет улучшающих циклов, то текущее решение является оптимальным. Если улучшающие циклы существуют, то выполняется переход к шагу поиска улучшающего цикла.

По данным [4] при решении трехиндексных транспортных задач метод потенциалов обеспечивает:

- быструю сходимость;
- учет динамики трехиндексных транспортных задач;
- простоту реализации.

Метод последовательной модификации функционала при решении трехиндексных транспортных задач [10] также является итеративным – предусматривает последовательное приближение к оптимальному решению за несколько циклов.

В начале выполняется инициализация, в ходе которой необходимо определить исходное базовое допустимое решение z(0) и определить модифицирующий функционал по формуле

$$F(z) = \sum\sum\sum c_{ijk} z_{ijk} + \alpha \sum\sum (i-j)^2 z_{ij,}$$

где $\alpha > 0$ – штрафной параметр

Первый шаг итерации. Для k = 1, 2, ..., n необходимо попытаться получить решение z(k) решив модифицированную транспортную задачу с функционалом F(z).

Если z(k) выполнимо и $\alpha = 0$, то решение является оптимальным. Значение объективной функции при z(n) рассчитывается по формуле:

$$F(z) = \sum\sum\sum c_{ijk} z_{ijk}\,(n).$$

В противном случае необходимо обновить α по формуле

$$\alpha = \alpha \cdot \beta,$$

где $\beta > 1$ – коэффициент увеличения параметра

и перейти к первому шагу итерации.

Метод последовательной модификации функционала позволяет решить трехиндексные транспортные задачи путем преобразования их в серию двухиндексных транспортных задач, но при этом может потребоваться много итераций для достижения оптимального решения. Также критичным для эффективности метода является выбор коэффициентов α и β.

Декомпозиционный метод решения трехиндексных транспортных задач [2], [3] предусматривает декомпозицию (разделение) трехиндексной задачи на некоторое количество более простых.

Для трехиндексной линейной транспортной задачи, описанной формулами (1) – (8) декомпозиционный метод решения запишется следующим образом.

Метод решения. Решая задачи по схеме [4], заметим, что каждая переменная x_{ijk} присутствует ровно в трех ограничениях Выпишем ограничения, содержащие x_{111}:

$$\sum_{j=1}^{n} x_{1j1} = a_{11},$$

$$\sum_{i=1}^{m} x_{i11} = b_{11},$$

$$\sum_{t=1}^{k} x_{11t} = c_{11},$$

К ограничениям 12 – 14 добавим целевую функцию

$$Z_{111} = d_{111}x_{111} + \sum_{j=2}^{n} \frac{d_{1i1}}{3} x_{1j1} + + \sum_{i=2}^{m} \frac{d_{i11}}{3} x_{i11} + \sum_{t=2}^{k} \frac{d_{11t}}{3} x_{11t} \to min.$$

Решаем эту задачу, затем находим выражение оптимальных решений через коэффициенты системы. Далее по схеме [4] формируем три независимые задачи с одним ограничением, где подбираются три коэффициента критерия оптимальности при связывающей переменной. Их сумма равняется исходному коэффициенту, а оптимумы задач с одним

ограничением в сумме должны быть равны оптимуму задачи с тремя ограничениями.

Задача (12) – (15) можно аналогично выписать три задачи с ограничениями (12) – (14) соответственно, так что объединение решений этих трех задач совпадает с решением задачи (12) – (15). Обозначим через $d1111$ коэффициент в целевой функции при $x111$, относящийся к ограничению 9, d_{111}^{2} коэффициент в целевой функции при x_{111}, относящийся к ограничению 10, d_{111}^{3} коэффициент в целевой функции при $x111$, относящийся к ограничению 11. Сумма новых коэффициентов равна

Вторая задача с тремя ограничениями имеет вид

$$\sum_{j=1}^{n} x_{1j2} = a_{12},$$

$$\sum_{i=1}^{m} x_{i12} = b_{12},$$

$$\sum_{t=1}^{k} x_{11t} = c_{11}.$$

$$Z_{112} = d_{111}^{3} x_{111} + d_{112} x_{111} \sum_{j=2}^{n} \frac{d_{1i2}}{3} x_{1j2} + \sum_{i=2}^{m} \frac{d_{i11}}{3} x_{i12} + \\ + \sum_{t=2}^{k} \frac{d_{11t}}{3} x_{11t} \to min.$$

Таких задач с тремя ограничениями будет всего m x k x n, и с решением каждой очередной задачи сумма целевых функций всех m+k+n задач с одним ограничением в их оптимальных решениях возрастает. При этом сумма ограничена сверху значением целевой функции в оптимальном решении исходной задачи (1) – (5). За конечное число циклов будет достигнут предел, который совпадает с оптимальным решением задачи (1) – (5).

Метод декомпозиции эффективен в случаях, когда имеют место трехиндексные транспортные задачи большого объема, задачи с разными типами ограничений, такими как неотрицательность, целочисленность и ограничения по пропускной способности, задачи с несколькими целями, в которых целевая функция представлена системой уравнений.

При этом метод декомпозиции относительно более медленный – для достижения решения необходимо большое количество итераций. Также к недостаткам метода относится негарантированная оптимальность и сложность управления вычислительным процессом для больших задач.

Потоковые методы [1] решения многоиндексных задач транспортного типа основаны на преобразовании многоиндексной транспортной задачи в эквивалентную задачу о максимальном потоке с одним индексом, решении задачи о максимальном потоке и разделении полученного потока на потоки по каждой индексной группе.

Потоковые методы эффективны для решения транспортных задач с большим количеством индексов. При этом необходимо отметить, что преобразование транспортной задачи в задачу о максимальном потоке может быть сложным для больших задач. При этом теряется информация об

индексной структуре транспортной задачи. Поэтому для задач со сложными индексными структурами и ограничениями потоковые методы не применяются. Потоковые методы оптимально использовать на больших задачах с простой структурой. При этом обеспечивается высокая вычислительная эффективность.

Достоинства и недостатки рассмотренных методов решения трехиндексных транспортных задач показаны в таблице 1.

Таблица 1 – Достоинства и недостатки различных методов решения трехиндексных транспортных задач

Метод	Достоинства	Недостатки
Метод потенциалов	Гарантированная оптимальность. Эффективен для задач с большим количеством ограничений	Сложность применения. Долгое время расчета
Метод последовательной модификации функционала	Производительность. Возможность применения для задач с нулевыми элементами в матрице затрат	Негарантированная оптимальность. Сложность в реализации
Декомпозиционный метод	Подходит для решения крупных задач	Негарантированная оптимальность. Сложность в реализации Требуется координация между подзадачами
Потоковые методы	Самая высокая производительность из рассмотренных методов Применимость для задач с большим количеством индексов.	Негарантированная оптимальность. Треднореализуем на задачах со сложной структурой

В целом декомпозиционный метод наряду с другими является эффективным средством решения трехиндексных транспортных задач.

На основании изложенного сформулированы требования к программной реализации декомпозиционного метода решения трехиндексной транспортной задачи.

Программа должна обеспечивать:

- удобный ручной ввод матриц исходных данных;

- контроль условия баланса поставок и потребления при вводе исходных данных;
- контроль условия баланса по объёмам перевозок при вводе исходных данных;
- контроль условия неотрицательности при вводе исходных данных;
- использование современных инструментов решения задач линейного программирования для решения декомпозиционных задач;
- возможность визуализации результата расчета;
- возможность загрузки исходных данных из текстового файла.

Выводы по разделу:

Выполнено описание трехиндексной транспортной задачи, в ходе которого определен тип задачи, описаны исходные данные, целевая функция и ограничения.

Рассмотрены основные методы решения трехиндексных транспортных задач. По результатам сравнительного анализа метода потенциалов, метода последовательной модификации функционала, декомпозиционного метода и потоковых методов показано, что декомпозиционный метод наряду с другими является эффективным средством решения трехиндексных транспортных задач.

Сформулированы требования к осуществлению программной реализации декомпозиционного метода решения трехиндексной транспортной задачи.

2 Программная реализация решения трехиндексной транспортной задачи

2.1 Постановка задачи и выбор средств разработки

В соответствии с требованиями к программной реализации декомпозиционного метода решения трехиндексной транспортной задачи, сформулированными в подразделе 1.2 архитектура программного решения состоит из трех модулей:

- модуль ввода;
- модуль расчета;
- модуль вывода.

В качестве языка программирования выбран Python [18].

Для реализации требований по удобству ввода применена библиотека пользовательского интерфейса tkinter [7].

Для решения использован модуль scipy.optimize.linprog библиотеки scipy [19] специализированный для решения оптимизационных задач линейного программирования. Преобразование исходных данных выполнено с использованием библиотеки numpy. Это библиотека открытым исходным кодом, которая, среди прочих, предоставляет функции для работы с многомерными массивами.

Непосредственно для решения задачи использована библиотека pulp, предназначенная для решения задач линейного программирования. Pulp упрощает решение задач линейного программирования, поскольку содержит встроенные инструменты поддержки низкого уровня алгоритмов решения.

Для визуализации результатов расчета использована библиотека matplotlib. Это библиотека визуализации Python, которая используется для создания статических, анимированных и интерактивных визуализаций на основе результатов расчетов.

2.2 Разработка алгоритмов решения задачи

В соответствии со сформулированными требованиями к модулям программного решения разработана UML диаграмма компонентов, на которой показано, что компонент «Ввод данных» предоставляет компоненту «Решение задачи» интерфейс для получения матриц с исходными данными. При этом при вводе данных может быть использована проверка баланса.

Результаты компонента «Решение задачи» передаются в компонент «Вывод решения», который выводит результат в текстовом виде в окне пользовательского интерфейса и в компонент «Визуализация решения», который строит столбчатую диаграмму по результатам решения.

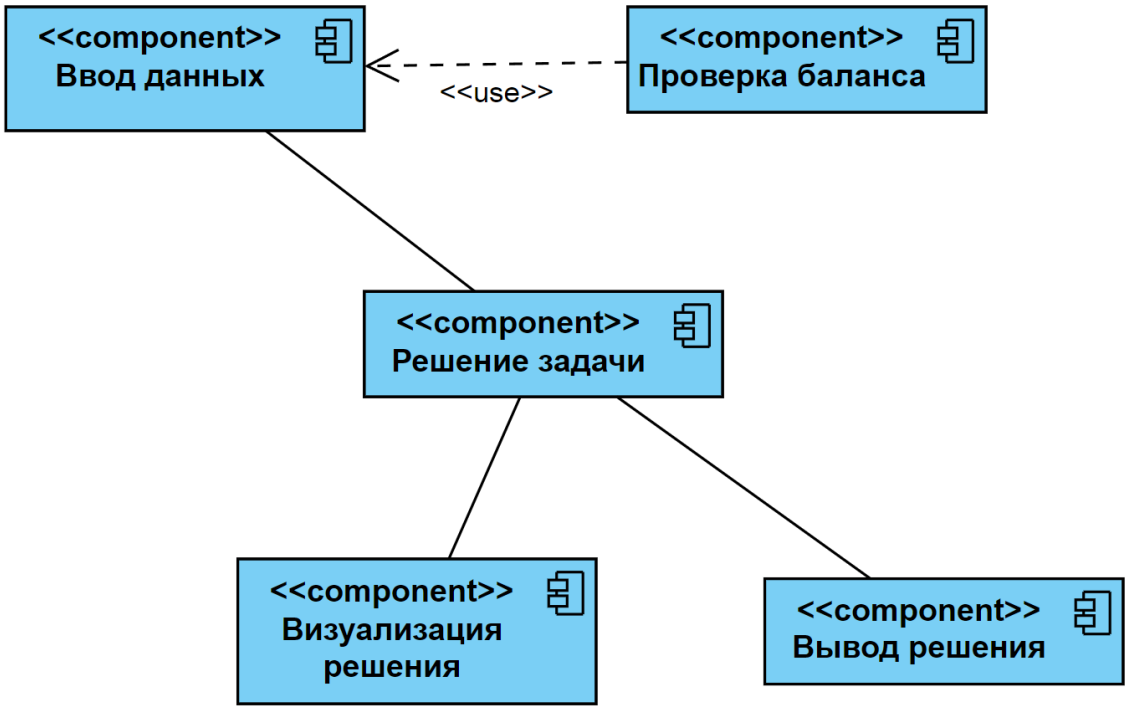

Рисунок 3 – Диаграмма компонентов модулей программного решения

Алгоритм работы компонента ввода данных, показанный на рисунке 4, предусматривает последовательный ввод размера задачи – количества

производителей, потребителей и продуктов, ввод матриц поставки и потребления с проверкой баланса поставки и потребления , ввод матрицы стоимости перевозки, формирование массивов – исходных данных для модуля расчета.

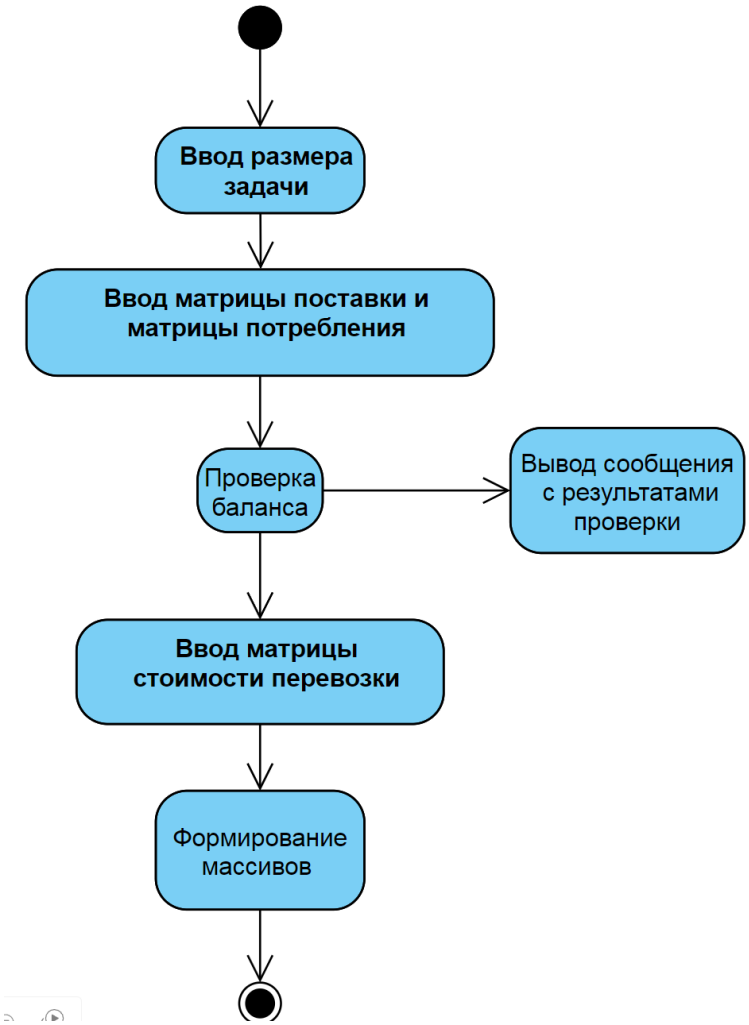

Рисунок 4 – Алгоритм работы модуля ввода в виде диаграммы активности

Алгоритм работы модуля расчета показанный на рисунке 5, предусматривае разбиение задачи на более простые блоки, решение транспортной здачи для каждого блока, сборку (агрегирование) общего решения из частного и уточнением расчета путем повторного решения задач для тх блоков, в которых условие баланса после агрегирования нарушается.

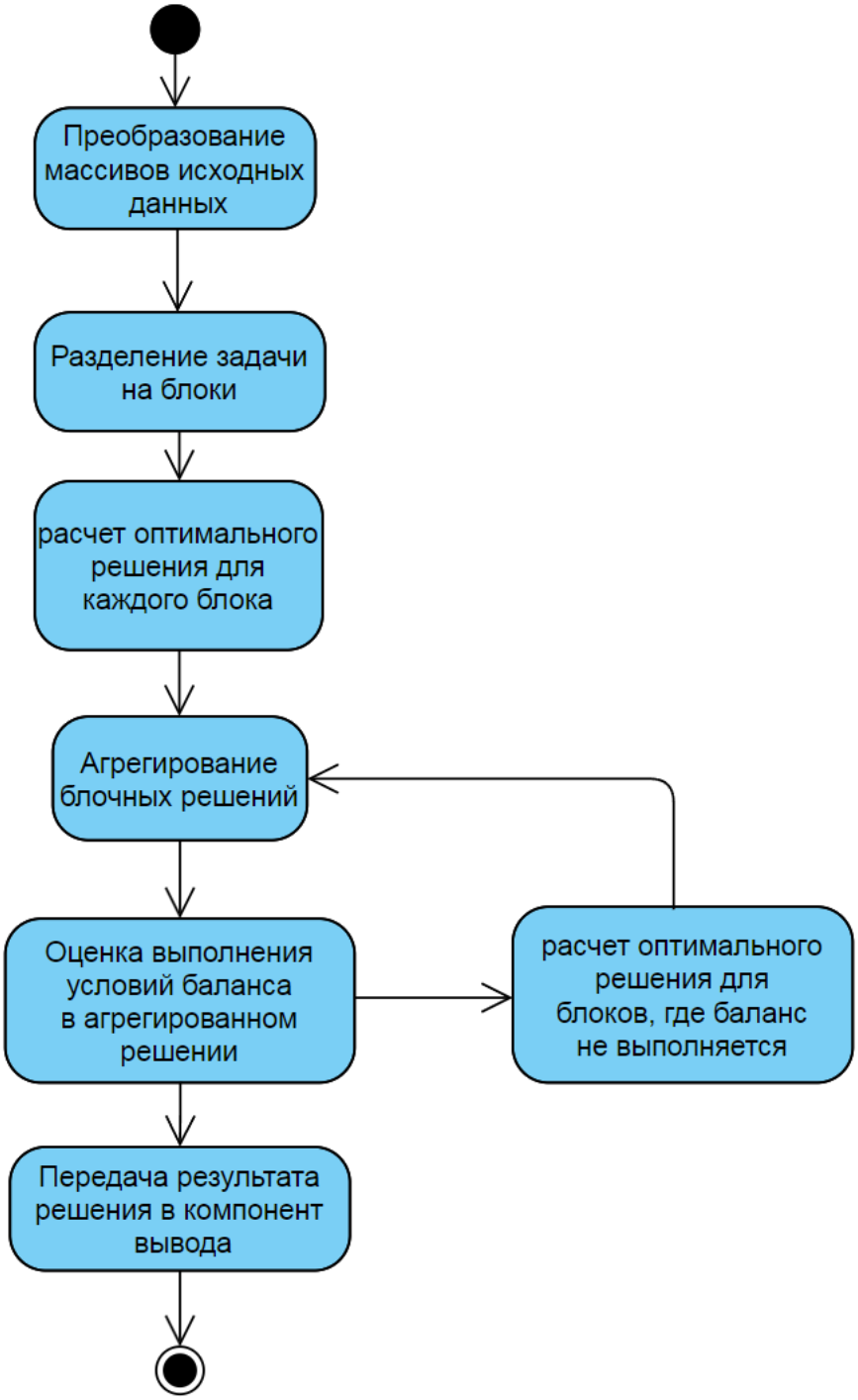

Рисунок 5 – Алгоритм работы модуля расчета в виде диаграммы активности

Этот процесс повторяется до тех пор, пока не будет получено агрегированное решение все элементы которого соответствуют условию баланса. Это решение будет оптимальным.

2.3 Реализация программных модулей

Модуль ввода данных последовательно предлагает пользователю ввести размер задачи, ввести матрицы поставки и потребления с возможностью проверки баланса введенных данных, ввести стоимость перевозки и после этого, приступить к расчету.

Фрагмент программного кода, отвечающего за формирование окна ввода размера задачи:

```
import tkinter as tk
from tkinter import filedialog, messagebox
# Создать главное окно
root = tk.Tk()
root.title("Размер задачи")
 # Создание поля ввода для количества поставщиков
num_suppliers_label = tk.Label(root, text="Количество поставщиков:")
num_suppliers_entry = tk.Entry(root)
# Создание поля ввода для количества
# Создание поля ввода для количества потребителей
num_consumers_label = tk.Label(root, text="Количество потребителей:")
num_consumers_entry = tk.Entry(root)
# Создание поля ввода для количества продуктов
num_products_label = tk.Label(root, text="Количество продуктов:")
num_products_entry = tk.Entry(root)
# Создание кнопки "Ручной ввод"
manual_entry_button = tk.Button(root, text="Ручной ввод")
manual_entry_button.config(command=open_supply_and_demand_window
)
# Создание кнопки "Загрузить из файла"
load_from_file_button = tk.Button(root, text="Загрузить из файла")
load_from_file_button.config(command=load_from_file)
```

```
# Размещение виджетов на окне
num_suppliers_label.grid(row=0, column=0)
num_suppliers_entry.grid(row=0, column=1)
num_consumers_label.grid(row=1, column=0)
num_consumers_entry.grid(row=1, column=1)
num_products_label.grid(row=2, column=0)
num_products_entry.grid(row=2, column=1)
manual_entry_button.grid(row=3, column=0)
load_from_file_button.grid(row=3, column=1)
# Запуск главного цикла приложения
root.mainloop()
```

Вид окна вода размера задачи показан на рисунке 6

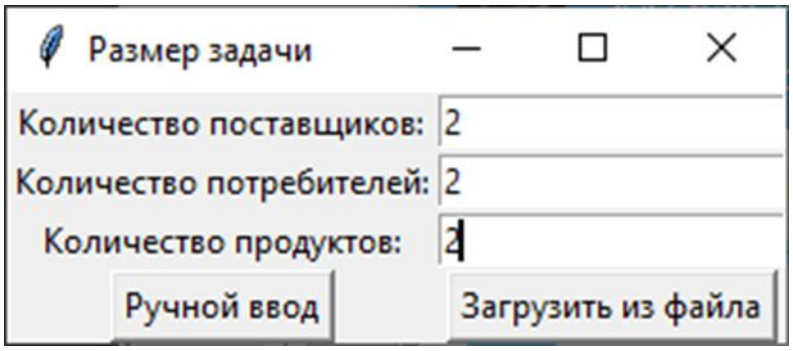

Рисунок 6 – Окно ввода размера задачи

При нажатии кнопки «Ручной ввод» открывается стандартное окно для указания файла и затем выполняется считывание данных из выбранного текстового файла с заполнением матриц поставки, потребления и емкости каналов передачи продуктов.

Код процедуры обработчика кнопки «Загрузить из файла»

```
# Определение функции для загрузки данных из файла
def load_from_file():
    # Открыть диалоговое окно выбора файла
```

```
    filename = filedialog.askopenfilename(title="Выберите файл с данными задачи")
    # Считать данные из файла
    with open(filename, "r") as file:
        # Считать количество поставщиков, потребителей и продуктов
        line = file.readline()
        num_suppliers, num_consumers, num_products = map(int, line.split())
    # Заполнить поля ввода данными из файла
    num_suppliers_entry.delete(0, tk.END)
    num_suppliers_entry.insert(0, num_suppliers)
    num_consumers_entry.delete(0, tk.END)
    num_consumers_entry.insert(0, num_consumers)
    num_products_entry.delete(0, tk.END)
    num_products_entry.insert(0, num_products)
```

При нажатии кнопки «Ручной ввод» выполняется открытие окна ввода матриц поставки и потребления. Этому действию соответствует следующий код:

```
# Определение функции для перехода к окну ввода поставки и потребления
def open_supply_and_demand_window():
    # Получить количество поставщиков, потребителей и продуктов из полей ввода
    num_suppliers = int(num_suppliers_entry.get())
    num_consumers = int(num_consumers_entry.get())
    num_products = int(num_products_entry.get())
    # Создать новое окно для ввода поставки и потребления
    supply_and_demand_window = tk.Tk()
    supply_and_demand_window.title("Ввод поставки и потребления")
    # Создание таблицы для ввода матрицы поставки
    supply_table = tk.Frame(supply_and_demand_window)
```

```
supply_table.pack()
# Создание заголовков столбцов матрицы поставки
product_headers = ["Про" + str(i + 1) for i in range(num_products)]
for i, header in enumerate(product_headers):
    tk.Label(supply_table, text=header).grid(row=0, column=i + 1)
# Создание заголовков строк матрицы поставки
supplier_headers = ["Пос" + str(i + 1) for i in range(num_suppliers)]
for i, header in enumerate(supplier_headers):
    tk.Label(supply_table, text=header).grid(row=i + 1, column=0)
# Создание полей ввода для матрицы поставки
supply_entries = []
for i in range(num_suppliers):
    for j in range(num_products):
        entry = tk.Entry(supply_table)
# Создание таблицы для ввода матрицы потребления
demand_table = tk.Frame(supply_and_demand_window)
demand_table.pack()
# Создание заголовков столбцов матрицы потребления
product_headers = ["Про" + str(i + 1) for i in range(num_products)]
for i, header in enumerate(product_headers):
    tk.Label(demand_table, text=header).grid(row=0, column=i + 1)
# Создание заголовков строк матрицы потребления
consumer_headers = ["Пот" + str(i + 1) for i in range(num_consumers)]
for i, header in enumerate(consumer_headers):
    tk.Label(demand_table, text=header).grid(row=i + 1, column=0)
# Создание полей ввода для матрицы потребления
demand_entries = []
for i in range(num_consumers):
    for j in range(num_products):
        entry = tk.Entry(demand_table)
```

```
demand_entries.append(entry)
```

Окно ввода матриц поставки и потребления показано на рисунке 7.

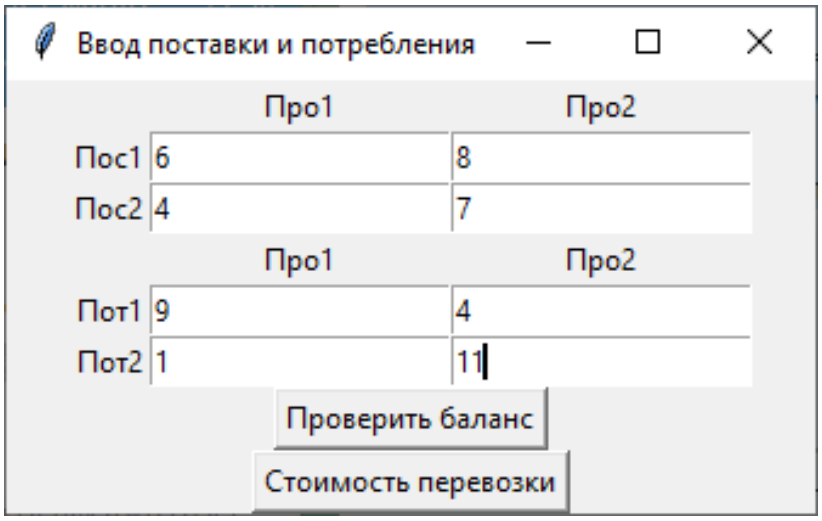

Рисунок 7 – Окно ввода матриц поставки и потребления

При нажатии кнопки «Проверить баланс» выполняется проверка баланса между поставкой и потреблением с выводом результата в отдельное окно.

Данное действие реализуется следующим кодом:

```
# Создание кнопки "Проверить баланс"
check_balance_button = tk.Button(supply_and_demand_window, text="Проверить баланс")
check_balance_button.pack()
# Определение функции для проверки баланса
def check_balance():
    # Получить значения из полей ввода
    supply_values = [float(entry.get()) for entry in supply_entries]
    demand_values = [float(entry.get()) for entry in demand_entries]
    # Проверить баланс для каждого продукта
    balance_results = []
```

```
            for i in range(num_products):
                supply_sum = sum(supply_values[i::num_products])
                demand_sum = sum(demand_values[i::num_products])
                if supply_sum == demand_sum:
                    balance_results.append("ОК")
                else:
                    balance_results.append("НАРУШЕНИЕ")
            # Создать окно "Баланс поставки и потребления"
            balance_window = tk.Tk()
            balance_window.title("Баланс поставки и потребления")
            # Вывести результаты проверки баланса
            for i, product in enumerate(product_headers):
                tk.Label(balance_window,                    text=f"{product}:
{balance_results[i]}").pack()
        # Привязать функцию проверки баланса к кнопке
        check_balance_button.config(command=check_balance)
```

Окно с результатами проверки баланса показано на рисунке 8.

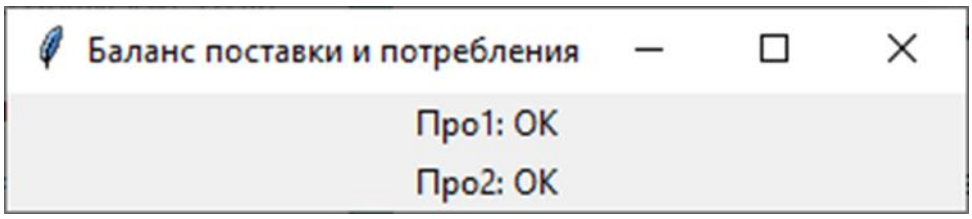

Рисунок 8 – Окно проверки баланса поставки и потребления

При нажатии кнопки «Стоимость перевозки» выполняется переход в окно (рисунок 9) для ввода соответствующих данных, после чего кнопкой «Выполнить расчет» запускается расчетный модуль.

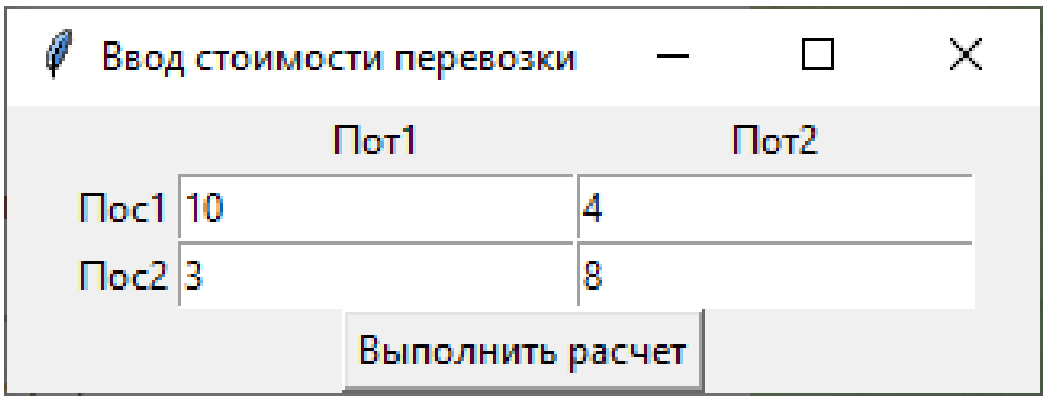

Рисунок 9 – Окно каналов перевозки

Результаты расчета выводятся в окне «Минимизация стоимости перевозки», которое показано на рисунке 10.

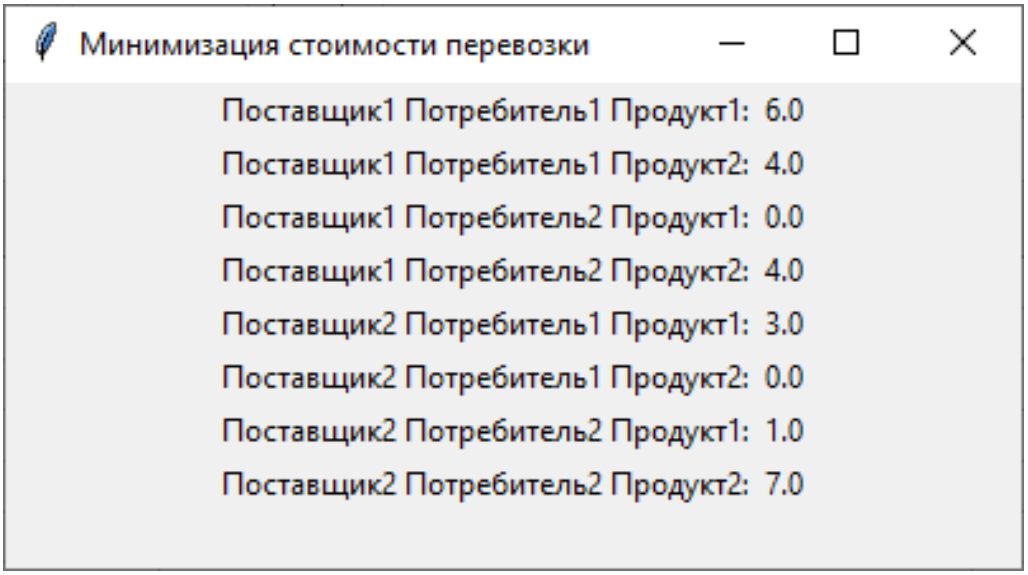

Рисунок 10 – Окно вывода результатов

Визуализация результатов выполняется столбчатой диаграммой в отдельном окне (рисунок 11).

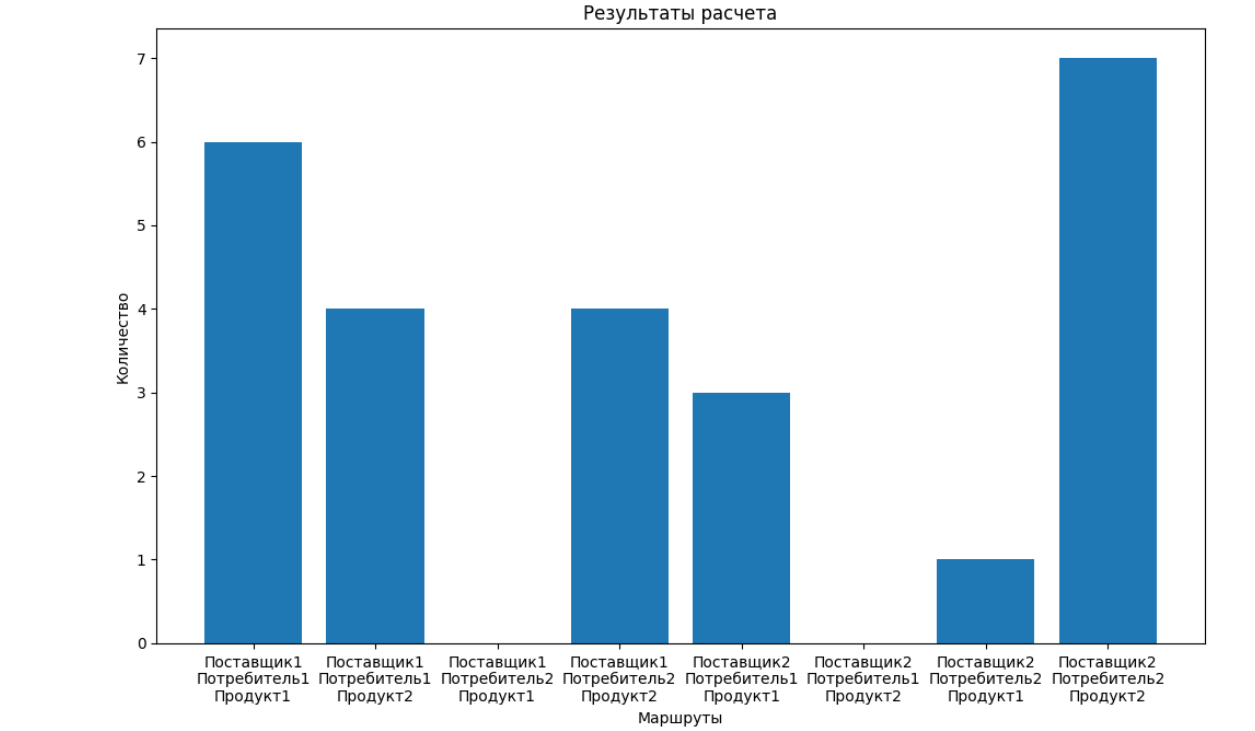

Рисунок 11 – Окно визуализации результатов

Программное решение разработано и работоспособно. На следующем этапе необходимо провести его тестирование.

Выводы по разделу:

Во втором разделе выполнен выбор средств разработки, разработаны схемы и алгоритмы, реализовано программное решение.

3 Тестирование программной реализации решения трехиндексной транспортной задачи

3.1 Планирование вычислительных экспериментов

Цель проведения вычислительного эксперимента – исследовать работоспособность программы решения трехиндексной транспортной задачи

В качестве варьируемых параметров выбраны число производителей, число потребителей, число продуктов. Пределы варьирования от 2 до 3.

Полный перебор всех вариантов варьируемых параметров составляет 8 экспериментов.

Эксперимент 1 включает 2 поставщика, 2 потребителя, 2 продукта.

Эксперимент 2 включает 2 поставщика, 2 потребителя, 3 продукта.

Эксперимент 3 включает 2 поставщика, 3 потребителя, 2 продукта.

Эксперимент 4 включает 3 поставщика, 2 потребителя, 2 продукта.

Эксперимент 5 включает 2 поставщика, 3 потребителя, 3 продукта.

Эксперимент 6 включает 3 поставщика, 2 потребителя, 3 продукта.

Эксперимент 7 включает 2 поставщика, 3 потребителя, 3 продукта.

Эксперимент 8 включает 3 поставщика, 3 потребителя, 3 продукта.

Значения матриц поставки, потребления и стоимости доставки задавались случайным образом с соблюдением требований баланса.

В ходе эксперимента оценивались такие параметры как время выполнения, наличие ошибок при выполнении программы, соответствие результата расчета ожидаемому (по количеству строк вывода).

Время выполнения оценивалось по информации, выводимой в терминал.

В ходе эксперимента использовано следующее оборудование и программное обеспечение:

Операционная система Майкрософт Windows 10 Корпоративная LTSC версия 10.0.17763 Сборка 17763

Имя системы DESKTOP-7HVU9C0

Процессор Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz, 2601 МГц, ядер: 2, логических процессоров: 4

Установленная оперативная память (RAM) 8,00 ГБ

Среда разработки IDE Visual Studio Code Version: 1.89.1

Язык программирования Python 3.12.3:

Подключенные модули и библиотеки: fonttools 4.51.0, matplotlib 3.8.4, numpy 1.26.4, pandas2.2.2, pillow 10.3.0, PuLP 2.8.0, scipy 1.13.0

3.2 Проведение вычислительных экспериментов

Результаты эксперимента показаны в таблице 2

Таблица 2 – Результаты вычислительного эксперимента

Поставщиков	Потребителей	Продуктов	Ожидаемый результат	Фактический результат	Время, с	Сообщения об ошибке
			Список 8 строк	Список 8 строк		нет
			Список 12 строк	Список 12 строк		нет
			Список 12 строк	Список 12 строк		нет
			Список 12 строк	Список 12 строк		нет
			Список 18 строк	Список 18 строк		нет
			Список 18 строк	Список 18 строк		нет
			Список 18 строк	Список 18 строк		нет
			Список 27 строк	Список 27 строк		нет

По результатам вычислительного эксперимента программа работает без сбоев и ошибок.

Пример экспериментального расчета для эксперимента 8 (3 поставщика, 3 потребителя, 3 продукта)

Окно ввода размера задачи показано на рисунке 12

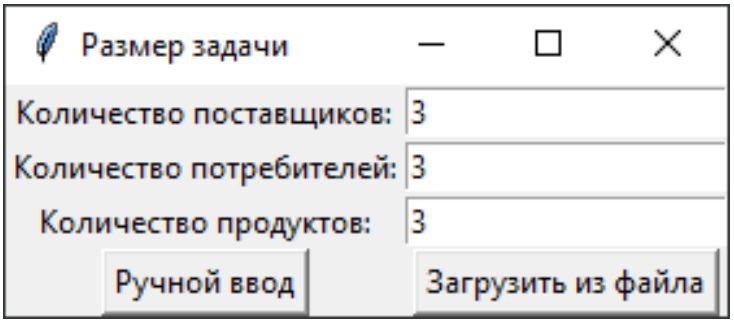

Рисунок 12 – Окно ввода размера задачи

Окно ввода матриц поставки и потребления показано на рисунке 13

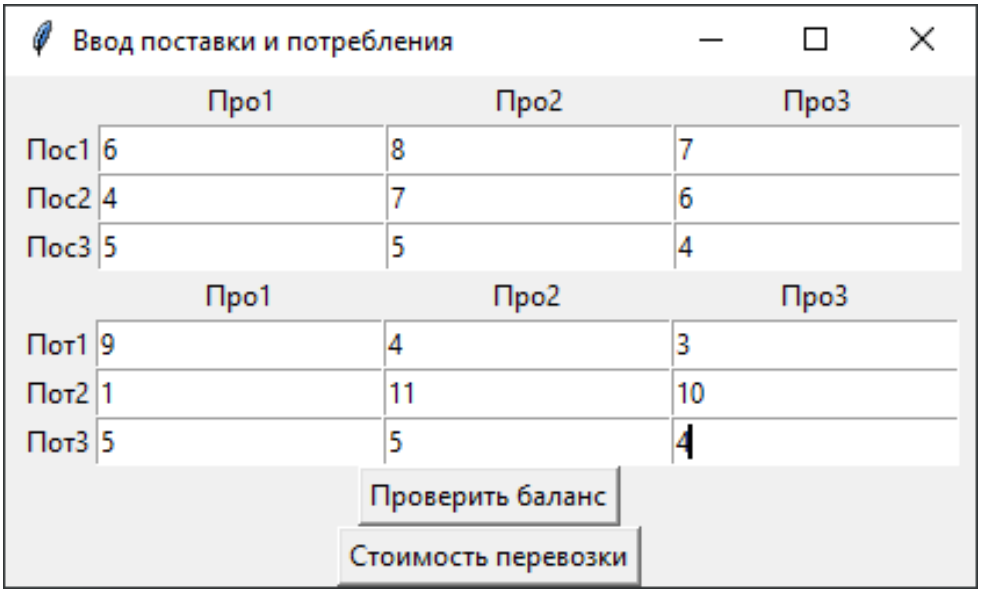

Рисунок 13 – Окно ввода матриц поставки и потребления

Окно ввода матрицы стоимости перевозки показано на рисунке 14.

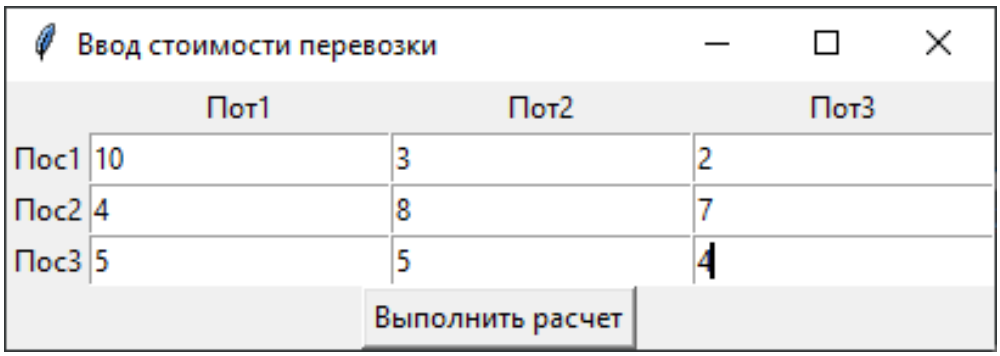

Рисунок 14 – Окно ввода матрицы стоимости перевозки

Фрагмент вывода в терминал, по которому определено время расчета показан на рисунке 15.

```
At line 2 NAME          MODEL
At line 3 ROWS
At line 17 COLUMNS
At line 43 RHS
At line 56 BOUNDS
At line 58 ENDATA
Problem MODEL has 42 rows, 9 columns and 126 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 0 (-42) rows, 0 (-9) columns and 0 (-126) elements
Empty problem - 0 rows, 0 columns and 0 elements
Optimal - objective value 0
After Postsolve, objective 0, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 0 - 0 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.14   (Wallclock seconds):       0.14
```

Рисунок 15 – Фрагмент вывода в терминал

Окно вывода результатов расчета показано на рисунке 16.

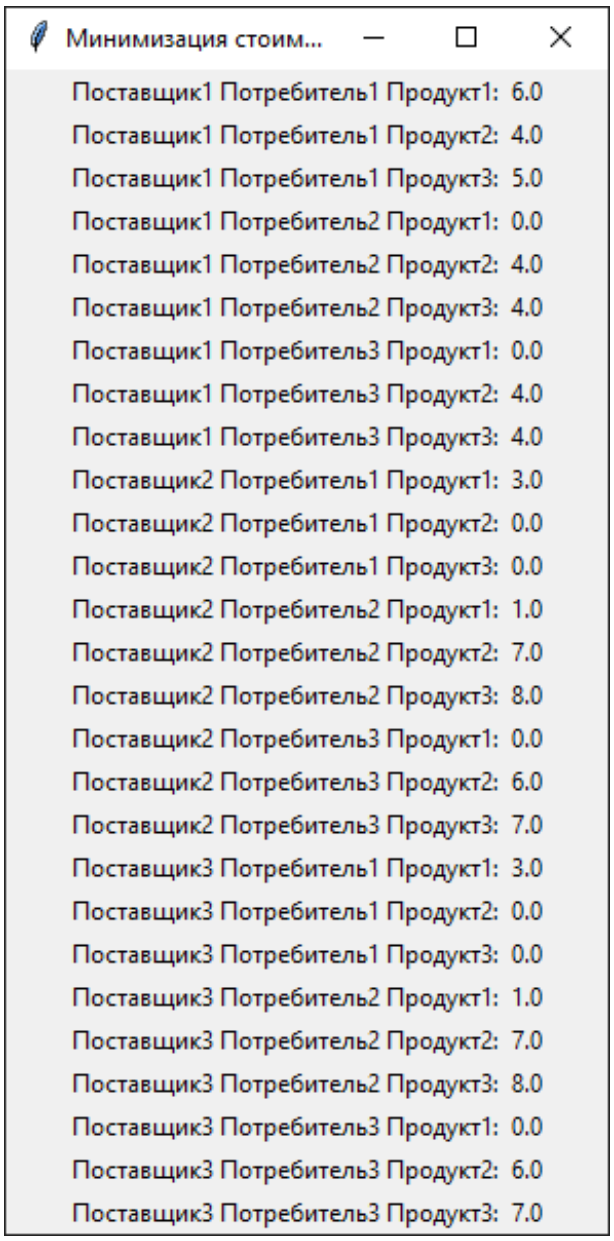

Рисунок 16 – Окно вывода результатов расчета

Выводы по разделу.

В третьем разделе выполнено тестирование разработанного программного решения. По результатам тестирования ошибок и сбоев в работе программы не выявлено.

Заключение

В результате анализа трехиндексной транспортной задачи» выполнено описание трехиндексной транспортной задачи как более расщиренного случая двухиндексной задачи. В результате анализа установлен тип задачи, исходные данные, ограничения на исходные данные, целевая функция и граничные условия решения.

Проведено исследование современных подходов и методов решения трехиндексных транспортных задач. По результатам сравнительного анализа показано, что декомпозиционный метод наряду с другими является эффективным средством решения трехиндексных транспортных задач.

Сформулированы требования к программной реализации декомпозиционного метода решения трехиндексной транспортной задачи.

В ходе программной реализации проведен выбор средств разработки – определен язык программирования и библиотеки, позволяющие реализовать требуемую функциональность и удобство пользовательского интерфейса.

Разработана общая модель взаимодействия модулей программного решения и разработаны алгоритмы работы модуля ввода и модуля расчета.

Программное решение по расчету трехиндексных транспортных задач реализовано в виде программы на языке программирования Python.

В ходе тестирования разработанной программы реализован план численного эксперимента с тремя варьируемыми параметрами – число производителей, число потребителей, число продуктов с оценкой времени расчета, соответствия результата расчета ожидаемому и отсутствию сбоев и ошибок в работе программы.

По результатам тестирования ошибок или сбоев в работе программы для решения трехиндексной транспортной задачи не выявлено.

В практическом плане разработанная программа может быть использована для решения трехиндексных транспортных задач методом декомпозиции.

Список используемой литературы и используемых источников

1. Афраймович Л. Г. Потоковые методы решения многоиндексных задач транспортного типа. Диссертация на соискание ученой степени доктора физико-математических наук [Электронный ресурс]. URL: http://www.ccas.ru/avtorefe/0002d.pdf. (дата обращения: 30.04.2024).
2. Ванг Л., Тизик А. П., Цурков В. И. Декомпозиционный алгоритм для линейной трехиндексной транспортной задачи/ Известия РАН. Теория и системы управления, 2019, № 6, стр. 57-62
3. Ванг Л. П., Есенков А. С., Тизик А. П., Торчинская Э. В. Декомпозиционный метод решения трехиндексных транспортных задач/ Известия РАН. Теория и системы управления, 2018, № 5, стр. 91-97
4. Криволапов Ю.А. Метод потенциалов для решения трехиндексной транспортной задачи. М.:ВИМИ, 1990. деп №Д08221.
5. Подробный разбор симплекс-метода [Электронный ресурс]. URL: https://habr.com/ru/post/474286/ (дата обращения: 30.04.2024).
6. Прилуцкий М.Х. Распределение однородного ресурса в иерархических системах древовидной структуры // Труды международной конференции «Идентификация систем и задачи управления SICPRO’2000». – М.: ИПУ им. В.А. Трапезникова РАН. 2000. С. 2038–2049.
7. Руководство по Tkinter [Электронный ресурс]. URL: https://metanit.com/python/tkinter/?ysclid=lw6kmbg95700050158 (дата обращения: 30.04.2024).
8. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. Учебное пособие. – М.: Физмалит. 2007
9. Смирнов А.В. Задача целочисленного сбалансирования трехмерной матрицы и сетевая модель // Моделирование и анализ информационных систем. 2009. Т. 16. № 3. С. 70–76.

10. Тизик А.П., Цурков В.И. Метод последовательной модификации функционала для решения транспортной задачи // АиТ. 2012. V.l. С.148 – 158.

11. Хачиян Л.Г. Полиномиальный алгоритм в линейном программировании // Доклады АН СССР. 1979. Т. 244. № 5. С. 1093–1096.

12. Briskorn D., Drexl A., Spieksma F.C.R. Round robin tournaments and three index assignment // 4OR: a Quarterly Journal of Operations Research. 2010. V. 8. P. 365–374.

13. Dantzig G.B. Linear programming and extensions. – Princeton, NJ: Princeton University Press. 1963

14. De Loera J., Onn S. All Rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables // Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science. 2004. V. 3064. P. 338–351

15. De Loera J., Onn S. The complexity of three-way statistical tables // SIAM Journal on Computing. 2004. V. 33. P. 819–836.

16. Karmarkar N. A new polynomial time algorithm for linear programming // Combinatorica. 1984. V. 4. P. 373–395

17. Krokhmal P., Murphey R., Pardalos P., Uryasev S., Zrazhevski G. Robust decision making: addressing uncertainties / Butenko et al. (Eds.). Cooperative Control: Models, Applications and Algorithms. Kluwer Academic Publishers. 2003. P. 165–185

18. Python 3.12.3 documentation [Электронный ресурс]. URL: https://docs.python.org/3.12/ (дата обращения: 30.04.2024).

19. SciPy documentation [Электронный ресурс]. URL: https://docs.scipy.org/doc/scipy/ (дата обращения: 30.04.2024).

20. Storms P.P.A., Spieksma F.C.R. An LP-based algorithm for the data association problem in multitarget tracking // Computers and Operation Research. 2003. V. 30. N 7. P. 1067–1085