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)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(наименование профиля, специализации)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка программного обеспечения для решения проблемы целочисленности в многоиндексной транспортной задаче»

Обучающийся М.В. Жукова
(И.О. Фамилия) (личная подпись)

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

Консультант к.п.н., доцент, А.В. Егорова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Аннотация

Выпускная квалификационная работа посвящена решению проблемы целочисленности в многоиндексной транспортной задачи.

Ключевые слова: многоиндексная, транспортная задача, трипланарная, метод Гомори, симплекс-метод.

Решение проблемы целочисленности в многоиндексной транспортной задаче является важным современным направлением исследований, результаты которого востребованы во многих областях практической деятельности.

Применение методов решения многоиндексных транспортных задач с ограничением на получение целочисленного решения во многих случаях обеспечивает оптимальные логистические решения, что потенциально может позволить сэкономить на транспортных расходах.

Объектом исследования данной бакалаврской работы являются математические методы решения многоиндексной транспортной задачи.

Предметом исследования является алгоритм и программная реализация обеспечения целочисленного решения многоиндексной транспортной задачи.

Цель работы - разработка программного обеспечения для обеспечения целочисленности решения многоиндексной транспортной задачи.

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

Выпускная квалификационная работа представлена на 58 страницах, включает 26 иллюстраций, 5 таблиц, список используемой литературы, состоящий из 20 источников, приложение.

Abstract

The graduate qualification work is devoted to solving the problem of integrability in the multi-index transport problem.

Key words: multi-index, transport problem, triplanar, Gomori method, simplex method.

The solution of the problem of integrability in a multi-index transport problem is an important modern direction of research, the results of which are in demand in many fields of practical activity.

The application of methods for solving multi-index transportation problems with the constraint of obtaining an integer solution in many cases provides optimal logistic solutions, which can potentially save on transportation costs.

The object of the research of this bachelor's work is mathematical methods of solving a multi-index transportation problem.

The subject of the research is an algorithm and software implementation to provide an integer solution of the multi-index transport problem.

The aim of the work is to develop software to ensure the integer solution of the multi-index transport problem.

During the performance of the graduate work was carried out a study of the subject area, software design, development and testing the software.

The graduate qualification work is presented on 58 pages, includes 26 illustrations, 5 tables, the list of the used literature consisting of 20 sources, the appendix.

Содержание

Введение

Решение проблемы целочисленности в многоиндексной транспортной задаче является важным современным направлением исследований, результаты которого востребованы во многих областях практической деятельности. Доктор физико-математических наук Л.Г. Афраймович в работе «Потоковые методы решения многоиндексных задач транспортного типа» определяет область прикладного применения таких задач следующим образом: «Существует широкий класс прикладных задач, формализуемых в виде многоиндексных задач (целочисленного) линейного программирования транспортного типа. Примерами таких 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 целочисленного сбалансирования многоиндексных матриц» [1].

В деятельности предприятия АО «ВАЗИНТЕРСЕРВИС» большую роль играет логистика транспортных поставок запасных частей к автомобилям. Применение методов решения многоиндексных транспортных задач с ограничением на получение целочисленного решения в этом случае обеспечивает оптимальные логистические решения, что потенциально может позволить сэкономить на транспортных расходах. Этим определяется актуальность работы.

Целью бакалаврской работы является обеспечение целочисленного решения многоиндексной транспортной задачи и его программная реализация.

Объектом исследования данной бакалаврской работы являются математические методы решения многоиндексной транспортной задачи.

Предметом исследования является алгоритм и программная реализация обеспечения целочисленного решения многоиндексной транспортной задачи.

Цель работы - разработка программного обеспечения для обеспечения целочисленности решения многоиндексной транспортной задачи

Задачи, которые необходимо решить для достижения указанной цели:

– исследование предметной области;

– проектирование программного обеспечения;

– разработка программного обеспечения.

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

В первой главе рассматриваются постановка и методы решения задачи линейного программирования с учетом целочисленности получаемого решения, а также класс транспортных задач, сводимых к задачам линейного программирования.

Во второй главе выполнено проектирование программного обеспечения для решения проблемы целочисленности. Решение состоит в получении специальными методами целочисленного решения таких трипланарных транспортных задач, которые не могут быть решены целочисленно обычными методами. Разработан прототип программного обеспечения в Excel.

В третьей главе проведены разработка программного обеспечения, тестирование и выполнен вычислительный эксперимент по исследованию зависимости времени решения от размера задачи с применением разработанного программного обеспечения.

1 Анализ транспортных задач линейного программирования

1.1 Описание транспортной задачи

«Линейное программирование – математическая дисциплина, посвященная теории и методам решения экстремальных задач на множествах n- мерного пространства, задаваемых системами линейными уравнений и неравенств» [8].

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.

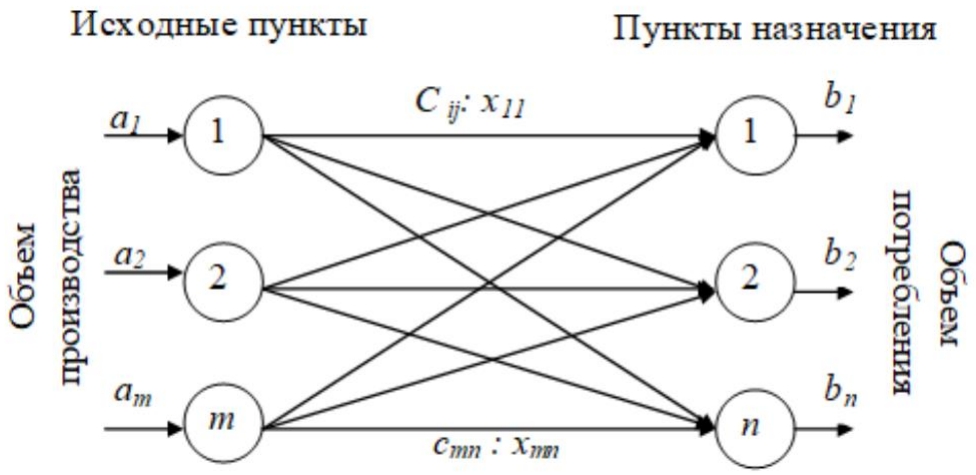


Рисунок 1 – Модель двухиндексной транспортной задачи

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.

1.2 Многоиндексные задачи линейного программирования

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. [13]; в области технического анализа данных при сопровождении объектов в многосенсорных системах [20]; в военной области при назначении военной техники на цели [19]. Известны результаты, посвященные исследованию вопросов сводимости задач линейного и целочисленного линейного программирования к многоиндексным транспортным задачам [16], что также расширяет область применения методов решения многоиндексных задач» [1].

«Многоиндексные задачи линейного программирования транспортного типа относятся к классу задач линейного программирования, который согласно [12] является полиномиально разрешимым. Таким образом, для решения многоиндексных задач линейного программирования транспортного типа могут быть применены общие методы исследований задач линейного программирования – симплекс метод [15], алгоритм Кармаркара [18].

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» [10].

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].

1.3 Методы решения транспортных задач

Исходя из обзора в подразделе 1.2 в качестве методов для решения транспортной задачи выбраны симплекс-метод, как наиболее универсальный метод решения задач линейного программирования. Данный метод не обеспечивает целочисленности. В качестве метода целочисленного решения выбран метод Гомори – он структурно близок к симплекс-методу. Такое сочетание методов решения позволит сравнить целочисленные и нецелочисленные решения многоиндексной транспортной задачи.

«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» [8].

«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» [8].

Алгоритм симплекс-метода показан на рисунке 2.

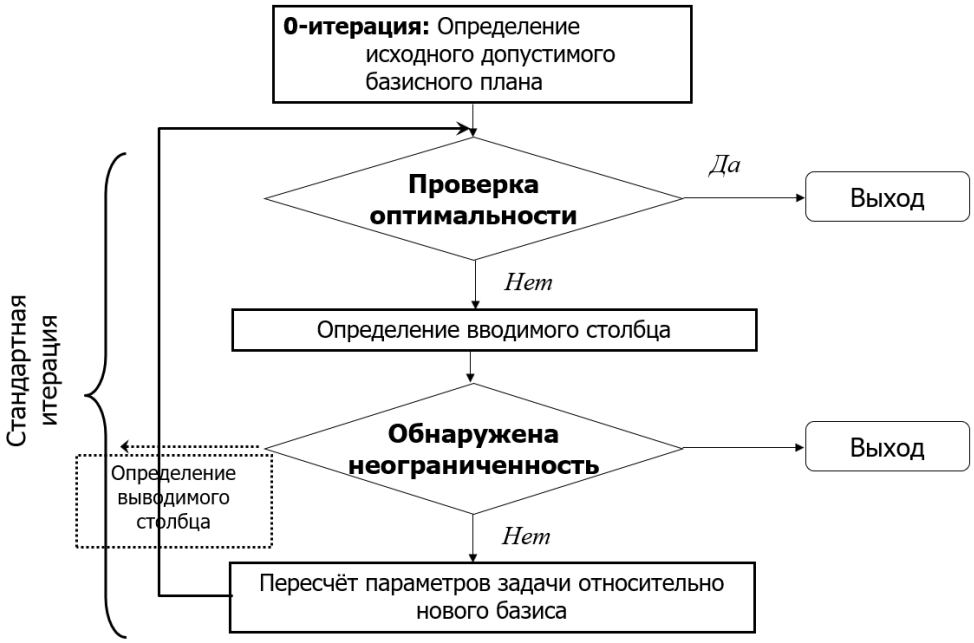


Рисунок 2 – Алгоритм симплекс-метода

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 » [8].

«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» [8].

«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)» [8].

Сущность метода отсечения, предложенного Р. Е. Гомори, состоит в том, что сначала задача решается без условия целочисленности. Если полученный план целочисленный, задача решена. В противном случае к ограничениям задачи добавляется новое ограничение, обладающее следующими свойствами: оно должно быть линейным; должно отсекать найденный оптимальный нецелочисленный план; не должно отсекать ни одного целочисленного плана. Далее задача решается с учетом нового ограничения. После этого в случае необходимости добавляется еще одно ограничение и т. д.

Геометрически (рисунок 3) добавление каждого линейного ограничения отвечает проведению прямой (гиперплоскости), которая отсекает от многоугольника (многогранника) решений некоторую его часть вместе с оптимальной точкой с нецелыми координатами, но не затрагивает ни одной из целых точек этого многогранника.

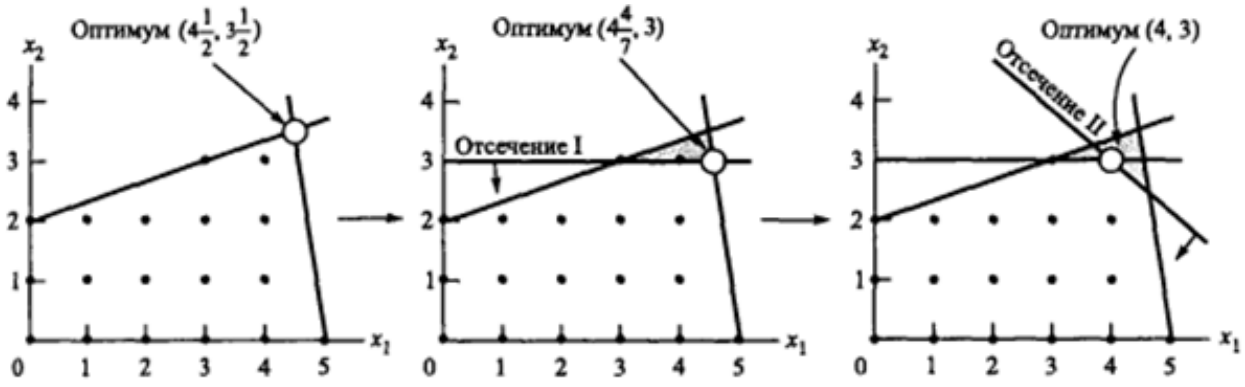


Рисунок 3 – Графическая интерпретация отсечений по методу Гомори

В результате новый многогранник решений содержит все целые точки, заключавшиеся в первоначальном многограннике решений, и соответственно полученное при этом многограннике оптимальное решение будет целочисленным (на рисунке 3 пространство допустимых решений задачи целочисленного линейного программирования представлено точками).

Искомое отсечение строится на основании дробных составляющих коэффициентов производящей строки. По этой причине его называют дробным отсечением. Если же в оптимальном плане задачи некоторая переменная имеет нецелое значение, то решается новая с учетом дополнительного неравенства. Под дробной частью числа k понимается разность k – [k], где [k] – целая часть числа k. Под целой частью числа k понимается наибольшее целое число, не превосходящее k.

Выводы по разделу:

Рассмотрены основные виды транспортных задач и определены их отличительные черты в классе задач линейного программирования.

Изучены виды и области применения многоиндексных транспортных задач.

Рассмотрены методы решения транспортных задач, среди которых выделены методы обеспечения целочисленности решения. На основании сравнительного анализа выбрана трипланарная задача и метод Гомори.

2 Проектирование ПО для решения проблемы целочисленности в многоиндексной транспортной задаче

2.1 Постановка задачи и выбор средств разработки

Выбрана трипланарная задача, в которой целочисленность решения обеспечивается методом Гомори.

Задача: отправители имеют к отправке некоторое целое число грузов, получатели ожидают получения некоторого целого числа грузов, имеется некоторое количество транспортных средств и набор цен на перевозку единицы груза от определенного отправителя определенному получателю определенным транспортом.

Задача является закрытой и сбалансированной – сумма грузов к отправке равна сумме грузов к получению, количество отправителей, получателей и транспортных средств – одинаковое.

Для решения крупных задач приложение должно обеспечивать генерацию исходных данных.

Для проведения сравнительных исследований приложение должно иметь возможность решения двухиндексной задачи (с одним транспортным средством).

Средства разработки – MS Excel с надстройкой «Поиск решения» для предварительной оценки решений и C++ для готового приложения.

2.2 Разработка алгоритмов решения задачи

Метод Гомори предусматривает сначала решение симплекс-методом и если получен нецелочисленный результат, то решение продолжается методом Гомори.

Очевидно, что решение проблемы целочисленности это получение целочисленного решения из нецелочисленного. Для этого применяют

специализированные целочисленные методы (Гомори), в случае, когда обычные методы (симплекс-метод) дают нецелочисленное решение.

Если обычные методы решения сразу дают целочисленный результат, это значит, что в такой задаче проблемы целочисленности нет.

Работа направлена на решение проблемы целочисленности, поэтому нельзя использовать задачи, в которых проблемы целочисленности нет. Это такие задачи, которые решаются нацело обычными методами.

Нужно использовать задачи, в которых проблема целочисленности есть. Это такие задачи, которые не решаются нацело обычными методами. В таких задачах использование, например, симплекс-метода, дает нецелочисленное решение.

Специально получать нецелочисленное решение нужно для того, чтобы потом решить проблему целочисленности. А затем, сравнив целочисленное и нецелочисленное решения, показать, что проблема решена.

Выполнено исследование задачи в простой постановке на модели-прототипе в Excel. Простая постановка — это вектор получателей, вектор отправителей и массив цен перевозки. Проведены расчеты при различных размерах задачи и различных сочетаниях исходных данных. Во всех исследованных случаях задача решается нацело симплекс-методом. Это значит, что проблемы целочисленности нет. Поэтому задачу в простой постановке нельзя использовать в работе, направленной на решение проблемы целочисленности.

Усложнение задачи путем введения дополнительного ограничения по грузоподъёмности согласовано с заказчиком ООО «ВАЗИНТЕРСЕРВИС». Если задать это ограничение произвольно, то слишком большая грузоподъёмность просто не будет работать, а слишком маленькая не позволит решить задачу. Определить величину ограничения, целесообразную в свете решаемой задачи, можно как долю от максимального количества груза на соответствующее транспортное средство, полученного при решении задачи в простой постановке (без ограничения по грузоподъёмности).

Алгоритм решения задачи представлен на рисунке 4.

Рисунок 4 – Общий алгоритм решения задачи

Транспортная задача с дополнительным ограничением по грузоподъёмности не решается нацело симплекс-методом. Значит проблема целочисленности есть. Поэтому задачу в такой постановке можно использовать в работе, направленной на решение проблемы целочисленности.

Задача решается симплекс-методом, затем, на основании результатов вводятся дополнительные ограничения и задача решается с их учетом симплекс-методом и, когда получено нецелочисленное решение, методом Гомори.

Общий алгоритм решения задачи при выборе коэффициента грузоподъёмности, обеспечивающего нецелые значения в векторе грузоподъёмности, гарантирует решение проблемы целочисленности методом Гомори на третьем расчете с получением массива Gruz3.

2.3 Исследование прототипа приложения в MS Excel

В соответствии с постановкой задачи самый простой вариант задачи, решаемой разрабатываемым приложением – двухиндексная с размером 2. В такой задаче два отправителя, два получателя и одно транспортное средство. Обозначим данную задачу как 2x2.

Лист с исходными данными и настройками поиска решения для первого расчета показан на рисунке 5.

Результаты первого расчета показаны на рисунке 6

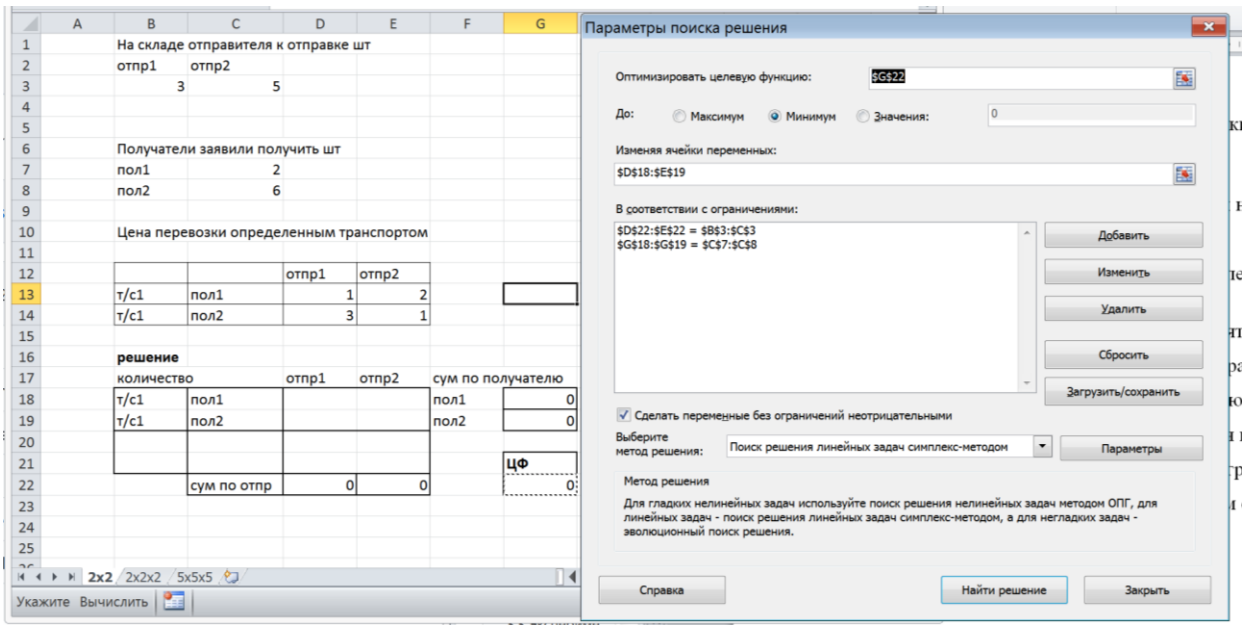


Рисунок 5 – Исходные данные и настройки поиска решения для первого расчета

	A	B	C	D	E	F	G
15							
16		**решение**					
17		количество		отпр1	отпр2	сум по получателю	
18		т/с1	пол1	2	0	пол1	2
19		т/с1	пол2	1	5	пол2	6
20							
21							**ЦФ**
22			сум по отпр	3	5		10

Рисунок 6 – Результаты первого расчета

По результатам первого расчета вводятся дополнительные ограничения по грузоподъёмности транспорта. Ограничения рассчитываются от максимального количества на соответствующем транспорте, полученного в результате решения. При этом используется коэффициент грузоподъёмности. В данном случае при коэффициенте грузоподъёмности 0,72 и максимальном количестве 5 отправленном вторым отправителем второму получателю получаем ограничение грузоподъёмности 3,6 и вносим ограничение в настройки поиска решения. Формула расчета грузоподъемности (рисунок 7)

обеспечивает грузоподъемность больше или равную единице. Это необходимо, потому что при переходе к трехиндексной задаче могут быть транспортные средства с максимальным количеством 1 после первого расчета, и, получив простым умножением на коэффициент, грузоподъёмность меньше единицы, они заведомо исключатся из третьего целочисленного расчета.

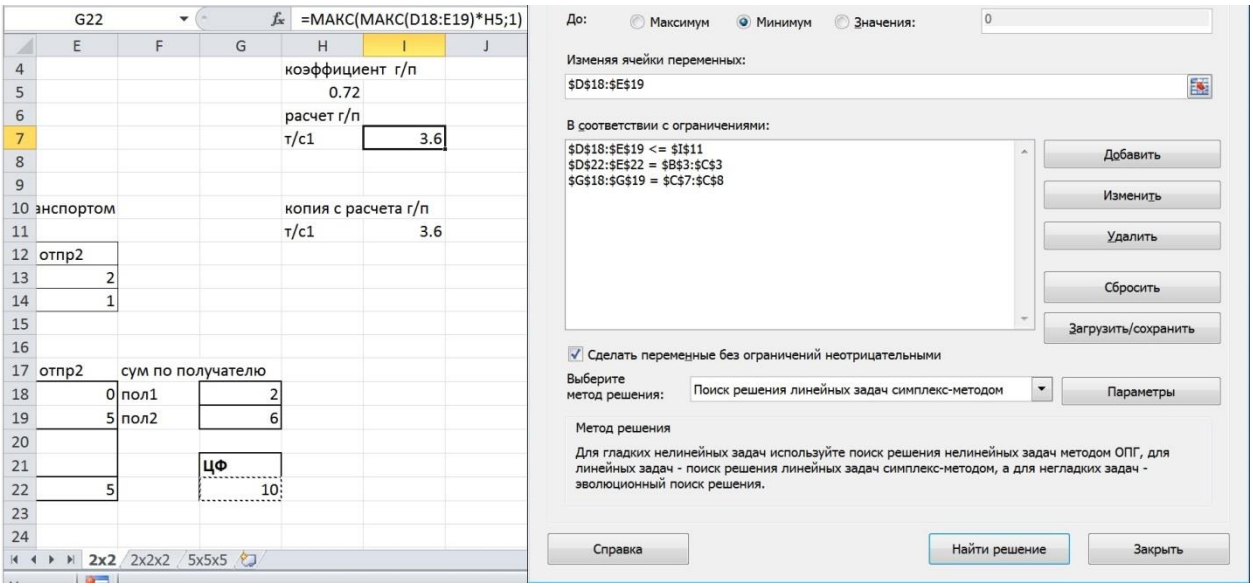


Рисунок 7 – Исходные данные и настройки поиска решения для второго расчета

Результаты второго расчета показаны на рисунке 8

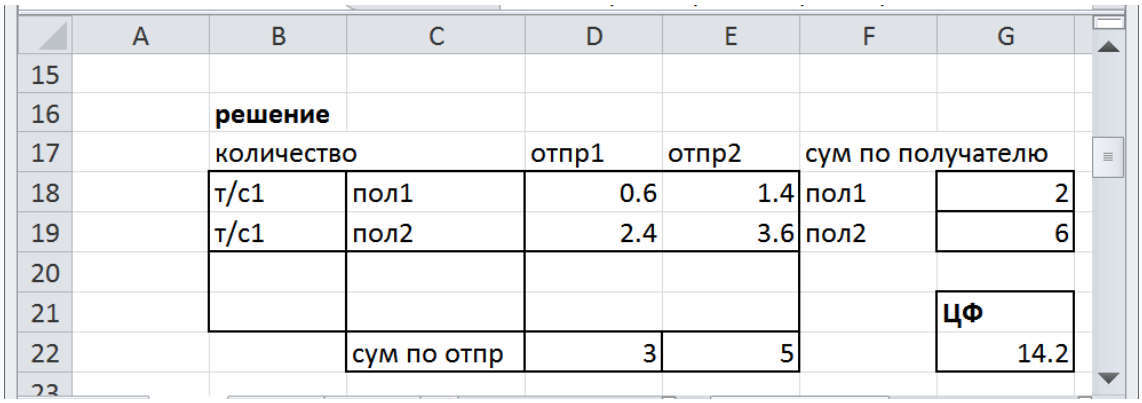


Рисунок 8 – Результаты второго расчета

На третьем заключительном расчете добавляем ограничение по целочисленности (рисунок 9) и получаем окончательное решение (рисунок 10)

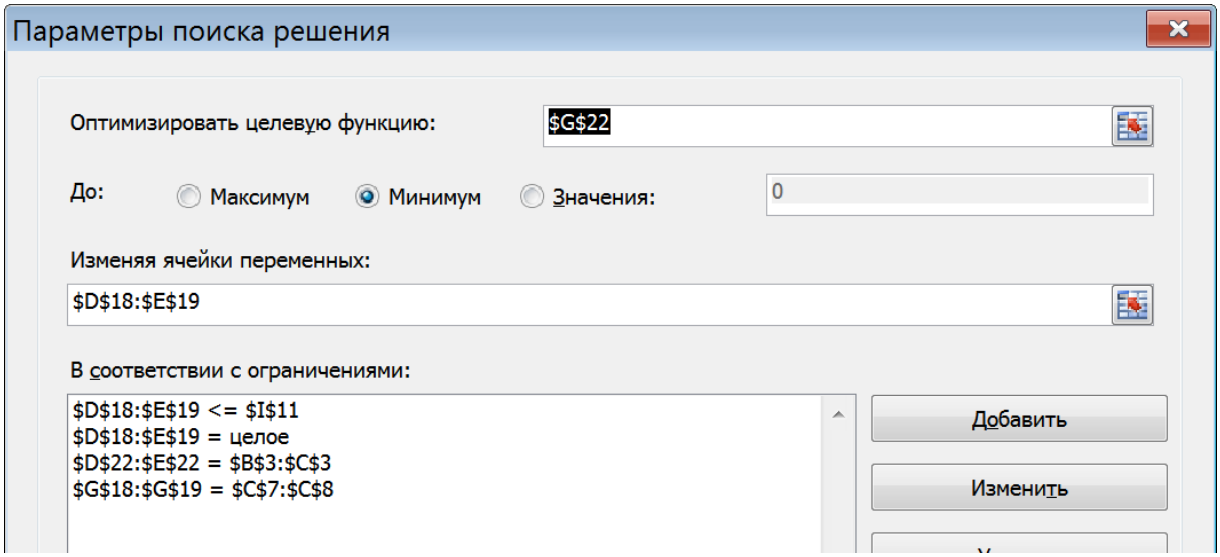


Рисунок 9 – Настройки поиска решения для третьего расчета

	A	B	C	D	E	F	G
15							
16		**решение**					
17		количество		отпр1	отпр2	сум по получателю	
18		т/с1	пол1	0	2	пол1	2
19		т/с1	пол2	3	3	пол2	6
20							
21							**ЦФ**
22			сум по отпр	3	5		16

Рисунок 10 – Результаты третьего расчета

Результаты расчетов, представленные на рисунках, показывают, что все ограничения выполнены, а значение целевой функции увеличивается при переходе от первого расчета к расчету с ограничением грузоподъемности и при переходе от второго расчета к целочисленному расчету с ограничением грузоподъемности, составляя, соответственно 10, 14,2 и 16.

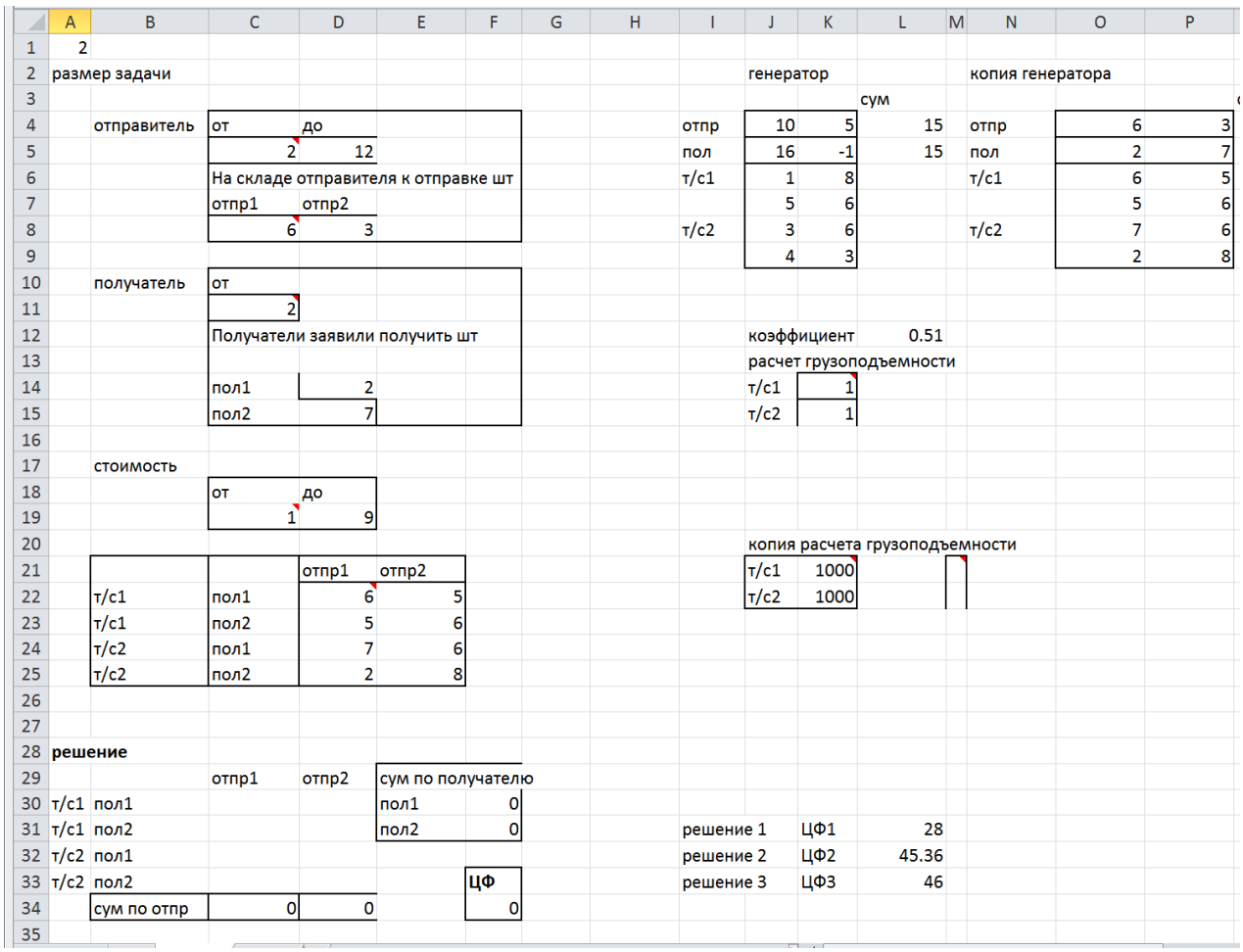
Трипланарная задача минимального размера предусматривает перевозку от двух отправителей двум получателям двумя транспортными средствами. Обозначим данную задачу как 2x2x2.

Стоимость перевозки каждым транспортным средством от каждого отправителя каждому поставщику – индивидуальна, следовательно, в рассматриваемом случае будет 8 стоимостей.

Для автоматизации заполнения исходных данных разработаны генераторы. Исходные данные для отправителя содержат минимальное и максимальное целые числа к отправке для одного отправителя. В расчётной модели (рисунок 11) исходные данные для отправителя заданы в диапазоне ячеек C5:D5. Генератор исходных данных отправителя задан в диапазоне ячеек J4:K4 и содержит формулу C5+ЦЕЛОЕ(СЛЧИС()*(D5-C5+1)).

Исходные данные для получателя должны в сумме быть равны данным отправителя. Поэтому содержат только минимальное целое число к получению для одного получателя. В расчётной модели (рисунок 7) исходные данные для получателя заданы в ячейке C11. Генератор исходных данных получателя задан в диапазоне ячеек J5:K5. Ячейка J5 содержит формулу C11+ЦЕЛОЕ(СЛЧИС()*C11) для расчета целого случайного числа большего чем минимальное, но меньшего чем удвоенное минимальное. В ячейке K5 формула L4-J5 дополняет набор получателей до суммы, равной сумме набора отправителей.

Исходные данные для стоимости доставки единицы груза содержат минимальное и максимальное целые числа стоимости одной доставки. В расчётной модели исходные данные для стоимости доставки заданы в диапазоне ячеек C19:D19. Генератор исходных данных стоимости задан в диапазоне ячеек J6:K9 и содержит формулу ЦЕЛОЕ((C19+D19)/2+(СЛЧИС()-0.5)*(D19-C19)).



1	2																	
2	размер задачи									генератор				копия генератора				
3												сум						
4		отправитель	от	до					отпр	10	5	15		отпр	6	3		
5			2	12					пол	16	-1	15		пол	2	7		
6			На складе отправителя к отправке шт						т/с1	1	8			т/с1	6	5		
7			отпр1	отпр2						5	6				5	6		
8			6	3					т/с2	3	6			т/с2	7	6		
9										4	3				2	8		
10		получатель	от															
11			2															
12			Получатели заявили получить шт							коэффициент	0.51							
13										расчет грузоподъемности								
14			пол1	2						т/с1	1							
15			пол2	7						т/с2	1							
16																		
17		стоимость																
18			от	до														
19			1	9														
20										копия расчета грузоподъемности								
21					отпр1	отпр2				т/с1	1000							
22		т/с1	пол1		6	5				т/с2	1000							
23		т/с1	пол2		5	6												
24		т/с2	пол1		7	6												
25		т/с2	пол2		2	8												
26																		
27																		
28	**решение**																	
29				отпр1	отпр2	сум по получателю												
30	т/с1	пол1				пол1	0											
31	т/с1	пол2				пол2	0			решение 1	ЦФ1	28						
32	т/с2	пол1								решение 2	ЦФ2	45.36						
33	т/с2	пол2					ЦФ			решение 3	ЦФ3	46						
34		сум по отпр		0	0		0											
35																		

Рисунок 11 – Расчетная модель для задачи 2x2x2

Генератор использует случайные числа и результаты изменяются при пересчете листа и, в частности, при запуске надстройки «Поиск решения». Для сохранения результатов генерации неизменными в диапазоне ячеек O4:P9 создана копия генератора, содержащая только значения без расчетных формул.

Для удобства выбора ограничений в поиске решения данные из диапазона копии генератора также отображаются в ячейках C8:D8 (набор к отправке), D14:D15 (набор к получению) и D22:E25 (набор стоимости доставки отправителя получателю транспортным средством).

Чтобы не менять ограничения в модели при переходе от расчета 1 к расчету 2, ограничения по грузоподъёмности сразу включены в расчётную

модель, но при первом расчете их значения (ячейки K21:K22) задаются заведомо больше, чтобы ограничения не работали.

Результаты решения, показанные на рисунке 12, свидетельствуют о том, что задача решена корректно.

копия результатов поиска решения без учета грузоподъемности

		отпр1	отпр2	сум по получателю	
т/с1	пол1	0	2	пол1	2
т/с1	пол2	0	1	пол2	7
т/с2	пол1	0	0		
т/с2	пол2	6	0		**ЦФ**
	сум по отп	6	3		28

копия результатов поиска решения с учетом грузоподъемности

		отпр1	отпр2	сум по получателю	
т/с1	пол1	1.02	0.08	пол1	2
т/с1	пол2	1.02	1.02	пол2	7
т/с2	пол1	0.9	0		
т/с2	пол2	3.06	1.9		**ЦФ**
	сум по отп	6	3		45.36

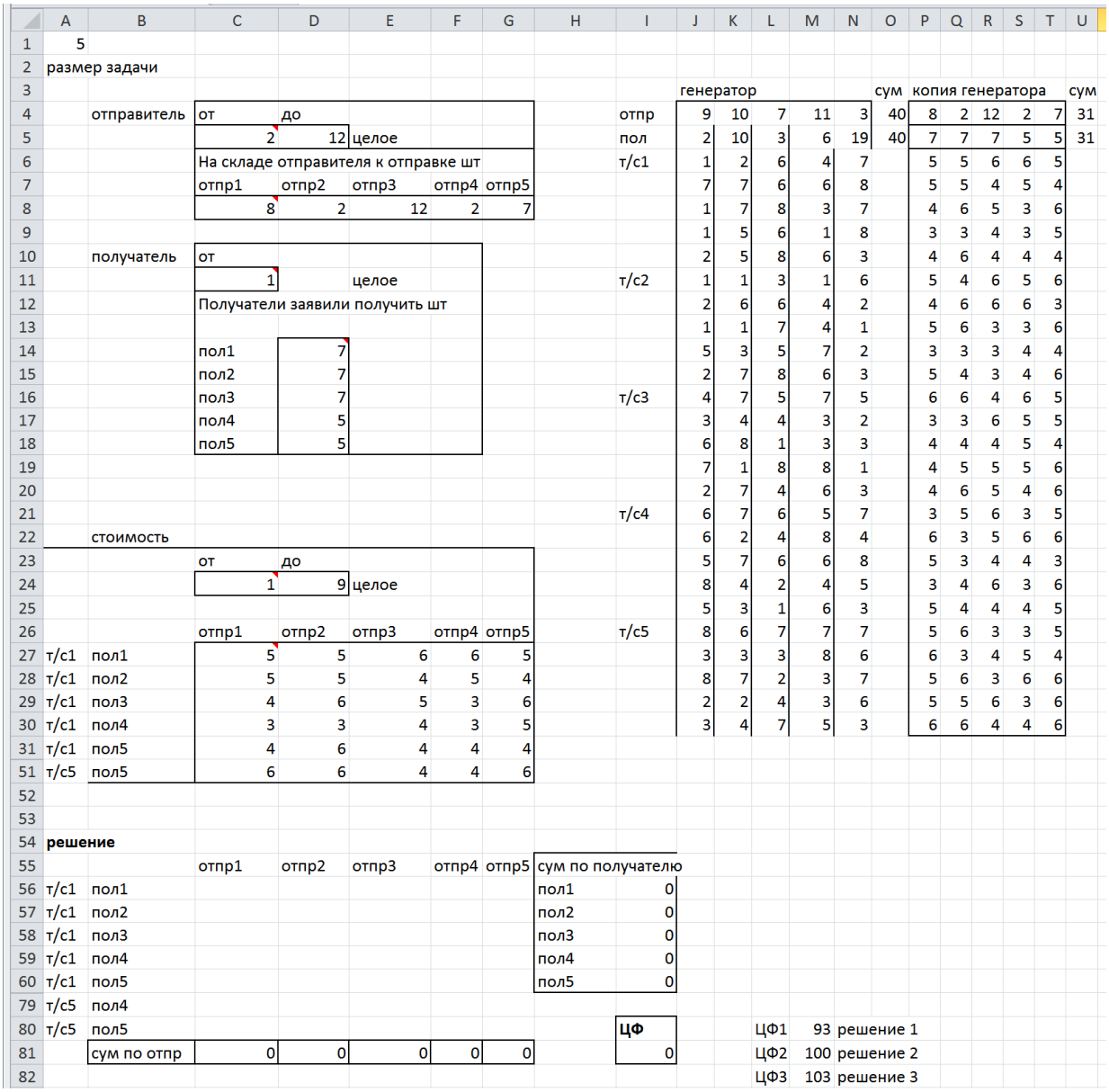
копия результатов поиска решения с учетом грузоподъемности целочисленный

		отпр1	отпр2	сум по получателю	
т/с1	пол1	1	0	пол1	2
т/с1	пол2	1	1	пол2	7
т/с2	пол1	1	0		
т/с2	пол2	3	2		**ЦФ**
	сум по отп	6	3		46

Рисунок 12 – Результаты решения задачи 2x2x2

При переходе к модели размером 5x5x5 (рисунок 13) помимо чисто масштабного увеличения задачи дополнительно прописан генератор получателей (диапазон ячеек J5:N5). Первые четыре ячейки диапазона J5:M5 содержат формулу ЦЕЛОЕ(C11+СЛЧИС()*(O4)/(A1-1)), в которой к генерируется целое число как сумма между минимальным для получателя значением и случайным числом, умноженным на сумму по отправителям, деленную на размер задачи минус один. Последняя ячейка набора получателя генерируется как разность между суммой по отправителям и суммой по четырем ячейкам получателей. Подобный подход позволяет избежать

отрицательных значений и получить достаточно широко распределенный набор случайных целых величин по указанному для получателя диапазону с равенством сумм получателей и отправителей.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	5																				
2	размер задачи																				
3										генератор					сум	копия генератора					сум
4		отправитель	от	до					отпр	9	10	7	11	3	40	8	2	12	2	7	31
5			2	12	целое				пол	2	10	3	6	19	40	7	7	7	5	5	31
6			На складе отправителя к отправке шт						т/с1	1	2	6	4	7		5	5	6	6	5	
7			отпр1	отпр2	отпр3	отпр4	отпр5			7	7	6	6	8		5	5	4	5	4	
8			8	2	12	2	7			1	7	8	3	7		4	6	5	3	6	
9										1	5	6	1	8		3	3	4	3	5	
10		получатель	от							2	5	8	6	3		4	6	4	4	4	
11			1		целое				т/с2	1	1	3	1	6		5	4	6	5	6	
12			Получатели заявили получить шт							2	6	6	4	2		4	6	6	6	3	
13										1	1	7	4	1		5	6	3	3	6	
14			пол1	7						5	3	5	7	2		3	3	3	4	4	
15			пол2	7						2	7	8	6	3		5	4	3	4	6	
16			пол3	7					т/с3	4	7	5	7	5		6	6	4	6	5	
17			пол4	5						3	4	4	3	2		3	3	6	5	5	
18			пол5	5						6	8	1	3	3		4	4	4	5	4	
19										7	1	8	8	1		4	5	5	5	6	
20										2	7	4	6	3		4	6	5	4	6	
21									т/с4	6	7	6	5	7		3	5	6	3	5	
22		стоимость								6	2	4	8	4		6	3	5	6	6	
23			от	до						5	7	6	6	8		5	3	4	4	3	
24			1	9	целое					8	4	2	4	5		3	4	6	3	6	
25										5	3	1	6	3		5	4	4	4	5	
26			отпр1	отпр2	отпр3	отпр4	отпр5		т/с5	8	6	7	7	7		5	6	3	3	5	
27	т/с1	пол1	5	5	6	6	5			3	3	3	8	6		6	3	4	5	4	
28	т/с1	пол2	5	5	4	5	4			8	7	2	3	7		5	6	3	6	6	
29	т/с1	пол3	4	6	5	3	6			2	2	4	3	6		5	5	6	3	6	
30	т/с1	пол4	3	3	4	3	5			3	4	7	5	3		6	6	4	4	6	
31	т/с1	пол5	4	6	4	4	4														
51	т/с5	пол5	6	6	4	4	6														
52																					
53																					
54	**решение**																				
55			отпр1	отпр2	отпр3	отпр4	отпр5	сум по получателю													
56	т/с1	пол1						пол1 0													
57	т/с1	пол2						пол2 0													
58	т/с1	пол3						пол3 0													
59	т/с1	пол4						пол4 0													
60	т/с1	пол5						пол5 0													
79	т/с5	пол4																			
80	т/с5	пол5							**ЦФ**			ЦФ1	93	решение 1							
81		сум по отпр	0	0	0	0	0		0			ЦФ2	100	решение 2							
82												ЦФ3	103	решение 3							

Рисунок 13 – Расчетная модель для задачи 5x5x5 (фрагмент)

Результаты решения по модели 5x5x5 (рисунок 14) показывают, что расчет проведен корректно. Все ограничения по суммам выполняются. С переходом к решению с учетом грузоподъёмности целевая функция увеличивается от 93 до 100, а в таблице результатов присутствуют нецелые

числа. С переходом к целочисленному решению с учетом грузоподъёмности целевая функция увеличивается от 100 до 103, а в таблице результатов присутствуют только целые числа.

	C	D	E	F	G	H	I	J	K	L	M
90	**копия результатов поиска решения без учета грузоподъемности**										
91			отпр1	отпр2	отпр3	отпр4	отпр5	сум по получателю			
92	т/с1	пол1	0	0	0	0	0	пол1	7		
93	т/с1	пол2	0	0	0	0	0	пол2	7		
94	т/с1	пол3	0	0	0	0	0	пол3	7		
95	т/с1	пол4	5	0	0	0	0	пол4	5		
96	т/с1	пол5	0	0	0	0	0	пол5	5		
116	т/с5	пол5	0	0	0	0	0		ЦФ		
117		сум по о	8	2	12	2	7		93		
118											
119											
120	**копия результатов поиска решения с учетом грузоподъемности нецелого**										
121			отпр1	отпр2	отпр3	отпр4	отпр5	сум по получателю			
122	т/с1	пол1	0	0	0	0	0.5	пол1	7		
123	т/с1	пол2	0	0	1	0	1.5	пол2	7		
124	т/с1	пол3	0	0	0	0	0	пол3	7		
125	т/с1	пол4	2.5	0	0	0	0	пол4	5		
126	т/с1	пол5	0.5	0	0	0	0	пол5	5		
146	т/с5	пол5	0	0	1	0	0		ЦФ		
147		сум по о	8	2	12	2	7		100		
148											
149											
150	**копия результатов поиска решения с учетом грузоподъемности целочисленного**										
151			отпр1	отпр2	отпр3	отпр4	отпр5	сум по получателю			
152	т/с1	пол1	0	0	0	0	1	пол1	7		
153	т/с1	пол2	0	0	2	0	1	пол2	7		
154	т/с1	пол3	0	0	0	0	0	пол3	7		
155	т/с1	пол4	2	0	0	0	0	пол4	5		
156	т/с1	пол5	1	0	1	0	0	пол5	5		
176	т/с5	пол5	0	0	0	0	0		ЦФ		
177		сум по о	8	2	12	2	7		103		

Рисунок 14 – Результаты решения задачи 5x5x5

В целом применение Excel для предварительной оценки и исследования задачи на малых размерах достаточно эффективно, т.к. электронные таблицы обладают простотой и наглядностью. Но даже на

размере задачи 5 объем таблиц становится слишком большим для визуальной оценки. Также время, необходимое для подготовки модели к расчету, связанное с ручной разметкой листа и записью формул с увеличением размера задачи становится очень большим. Программа на языке программирования, разработанная в следующем разделе, позволит автоматизировать подготовку данных для любого размера задачи и представить результаты с нужной подробностью в зависимости от цели того или иного расчета. Также преимуществом программы на языке программирования, по сравнению с Excel, является возможность в будущем легкой интеграции с реальными системами транспортной логистики путем замены генераторов исходных данных и механизмов вывода на соответствующие интерфейсы обмена данными.

Выводы по разделу:

Во втором разделе выполнено проектирование ПО для решения проблемы целочисленности в многоиндексной транспортной задаче.

Выполнен выбор средств разработки и постановка задачи. Разработан алгоритм, обеспечивающий решение проблемы целочисленности. Проведено моделирование работы приложения и генераторов исходных данных на прототипе в программе MS Excel.

3 Разработка ПО для решения проблемы целочисленности в многоиндексной транспортной задаче

3.1 Разработка приложения

Для разработки консольного приложения на C++ использована IDE Code::Blocks 20.03.

Необходимые для расчета библиотеки - iostream, vector, iomanip, algorithm, fstream.

«Iostream — заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от Input/Output Stream («поток ввода-вывода»).»[5]

«Vector (std::vector<T>) — стандартный шаблон обобщённого программирования языка C++, реализующий динамический массив.

Шаблон vector расположен в заголовочном файле <vector>. Как и все стандартные компоненты, он расположен в пространстве имён std. Данный интерфейс эмулирует работу стандартного массива C (например, быстрый произвольный доступ к элементам), а также некоторые дополнительные возможности, вроде автоматического изменения размера вектора при вставке или удалении элементов.»[7]

Библиотека iomanip содержит инструменты для работы с форматирование вывода.

«Algorithm — заголовочный файл в стандартной библиотеке языка программирования C++, включающий набор функций для выполнения алгоритмических операций над контейнерами и над другими последовательностями.»[3]

«Fstream (сокращение от «FileStream») — заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в

файл. Для манипуляции с данными файлов используются объекты, называемые потоками («stream»).

Функции, включенные в данный файл, позволяют производить чтение из файлов как побайтово, так и блоками, и записывать так же. В комплект включены все необходимые функции для управления последовательностью доступа к данным файлов, а также множество вспомогательных функций.»[4]

«Time.h — заголовочный файл стандартной библиотеки языка программирования C, содержащий типы и функции для работы с датой и временем.» [6]

Библиотека bits/stdc++.h подключена для использования функции суммирования элементов векторов.

Библиотека chrono это заголовок C ++, который предоставляет набор типов и функций для работы со временем. Он является частью стандартной библиотеки шаблонов C ++ (STL) и включен в C ++ 11 и более поздние версии. [14]

Chrono предоставляет три основных типа часов: system_clock, steady_clock и high_resolution_clock. Эти часы используются для измерения времени различными способами. System_clock представляет собой общесистемные настенные часы реального времени. На это влияют настройки системного времени. Steady_clock представляет собой монотонно увеличивающиеся часы, на которые не влияют изменения системного времени. High_resolution_clock - это часы с наименьшим тиковым периодом, доступным в системе.

Chrono также предоставляет коллекцию типов длительности, которые могут использоваться для представления длительности времени. Rep - это тип представления (например, int или long), а Period - это отношение длительности (например, наносекунд или секунд).

Кроме того, chrono предоставляет коллекцию типов временных точек, включая time_point, которые могут использоваться для представления момента времени.[2]

Для изменения единиц времени используется формат преобразования в стиле C++ duration_cast, которому, в качестве шаблонного параметра, необходимо передать один из возможных типов перечисленных ниже:

– std::chrono::nanoseconds;
– std::chrono::microseconds;
– std::chrono::milliseconds;
– std::chrono::seconds;
– std::chrono::minutes;
– std::chrono::hours.

Также в начальной части кода программы (рисунок 15) определяются параметры генераторов исходных данных, такие как размер задачи, предельные значения к отправке каждого отправителя, предельные значения цены перевозки единицы груза от определенного отправителя определенному получателю определенным транспортным средством и доля (в процентах) грузоподъёмности транспортного средства, используемая для расчета дополнительного ограничения.

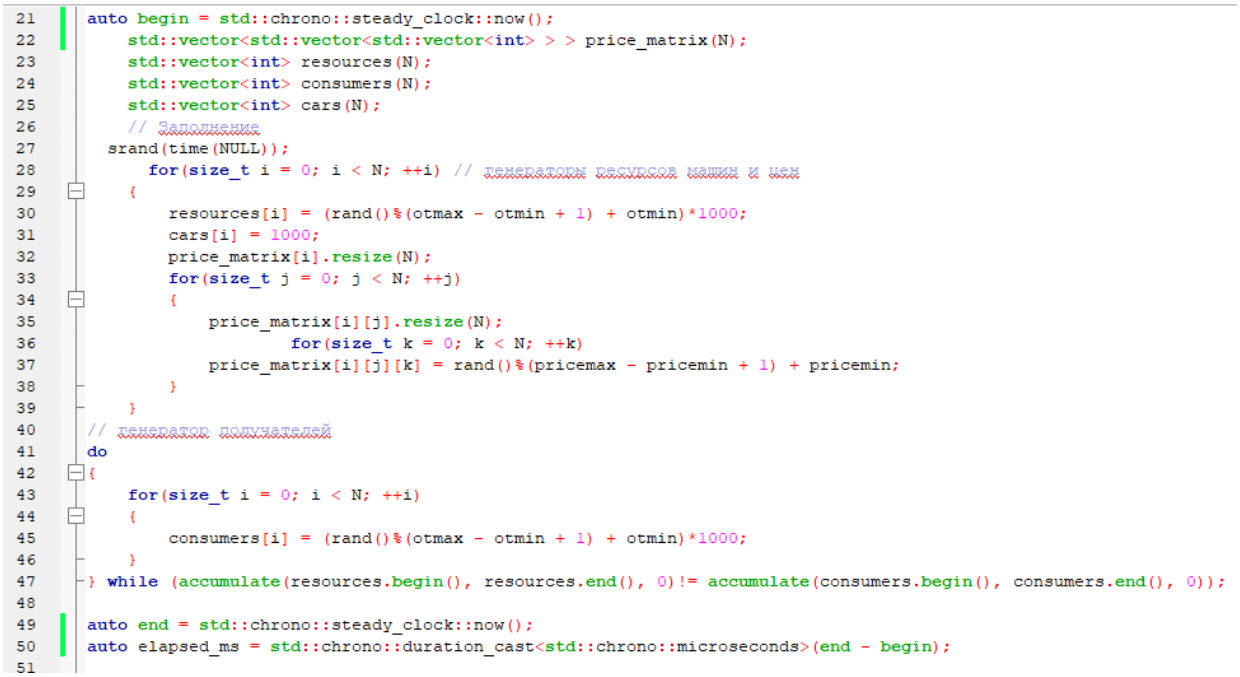
```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
#include <iomanip>

#include <time.h> // для генератора случайных
#include <bits/stdc++.h> // для суммирования элементов вектора
#include <fstream> // вывод в файл
#include <chrono> // замер времени

const int N = 5; // размер задачи
const int pricemin = 1; // минимальная цена превозки
const int pricemax = 6; // максимальная цена превозки
const int otmin = 1; // минимальное количество к отправке или по
const int otmax = 200; // максимальное  к отправке или получению
const int kg = 51;//процент грузоподьемности от максимума
```

Рисунок 15 – Подключение библиотек и задание исходных данных

Код генераторов массивов исходных данных показан на рисунке 16.



```
21  auto begin = std::chrono::steady_clock::now();
22      std::vector<std::vector<std::vector<int> > > price_matrix(N);
23      std::vector<int> resources(N);
24      std::vector<int> consumers(N);
25      std::vector<int> cars(N);
26      // Заполнение
27    srand(time(NULL));
28        for(size_t i = 0; i < N; ++i) // генераторы ресурсов машин и цен
29      {
30          resources[i] = (rand()%(otmax - otmin + 1) + otmin)*1000;
31          cars[i] = 1000;
32          price_matrix[i].resize(N);
33          for(size_t j = 0; j < N; ++j)
34          {
35              price_matrix[i][j].resize(N);
36                      for(size_t k = 0; k < N; ++k)
37              price_matrix[i][j][k] = rand()%(pricemax - pricemin + 1) + pricemin;
38          }
39      }
40  // генератор получателей
41  do
42  {
43      for(size_t i = 0; i < N; ++i)
44      {
45          consumers[i] = (rand()%(otmax - otmin + 1) + otmin)*1000;
46      }
47  } while (accumulate(resources.begin(), resources.end(), 0)!= accumulate(consumers.begin(), consumers.end(), 0));
48
49  auto end = std::chrono::steady_clock::now();
50  auto elapsed_ms = std::chrono::duration_cast<std::chrono::microseconds>(end - begin);
51
```

Рисунок 16 – Генераторы исходных данных

В строках 19-22 выполняется инициализация трехмерного массива цены перевозки и одномерных массивов получателей, отправителей и транспортных средств. Все массивы объявлены целыми.

Затем выполняется наполнение случайными данными массивов цен и отправителей. Массив грузоподъёмности заполняется числом 1000 – это большое значение исключает ограничение по грузоподъёмности на первом симплекс-расчете.

Генератор получателей выполнен отдельно, поскольку необходимо обеспечить равенство сумм отправителей и получателей. Генерация получателей выполняется от тех же предельных значений, что и отправителей, но в цикле, выход их которого выполняется при условии равенства сумм.

Поскольку в решаемую задачу входит анализ целочисленных и нецелочисленных решений, то непосредственное использование типа данных integer не представляется возможным – этот тип данных хранит только целые числа и в этом случае в принципе невозможно получить нецелочисленное

решение. С другой стороны, переход к типу данных float с плавающей запятой также сопряжен с определенными трудностями, поскольку float не является точным типом данных. Решением может быть тип данных с фиксированной запятой, подобный типу decimal языка SQL, который является точным и рассчитывается по целочисленным технологиям. В языке C++ подобный тип данных отсутствует, но может быть введен искусственно, так, для точных нецелочисленных расчетов до третьего знака поле запятой достаточно умножить исходные данные на 1000 и проводить расчет по целочисленным технологиям, а при выводе разделить результат на 1000 и обеспечить раздельный вывод целой части и остатка через десятичный разделитель. Код вывода на экран сгенерированных исходных данных показан на рисунке 17, пример вывода для задачи с размером 5 – на рисунке 18.

```
// Вывод на экран отправителей
std::cout << std::endl;
std::cout << "resources" ;
std::cout << std::endl;
    for(size_t i = 0; i < N; ++i)
         std::cout << resources[i]/1000 << "." << resources[i]%1000 << "  ";
std::cout << "sum resources " << accumulate(resources.begin(), resources.end(), 0)/1000
<< "." << accumulate(resources.begin(), resources.end(), 0)%1000 << std::endl<< std::endl;
// Вывод на экран получателей
std::cout << std::endl;
std::cout << "consumers" ;
std::cout << std::endl;
    for(size_t i = 0; i < N; ++i)
         std::cout << consumers[i]/1000 << "." << consumers[i]%1000 << "  ";
std::cout << "sum consumers " << accumulate(consumers.begin(), consumers.end(), 0)/1000
<< "." << accumulate(consumers.begin(), consumers.end(), 0)%1000 << std::endl<< std::endl;

//Вывод на экран цен перевозки
std::cout << "price" ;
    for(size_t i = 0; i < N; ++i)
    {
     std::cout << std::endl;
        for(size_t j = 0; j < N; ++j)
        {
         std::cout << std::endl;
            for(size_t k = 0; k < N; ++k)
            std::cout << price_matrix[i][j][k] << "  ";
        }
    }
```

Рисунок 17 – Код вывода на экран сгенерированных исходных данных

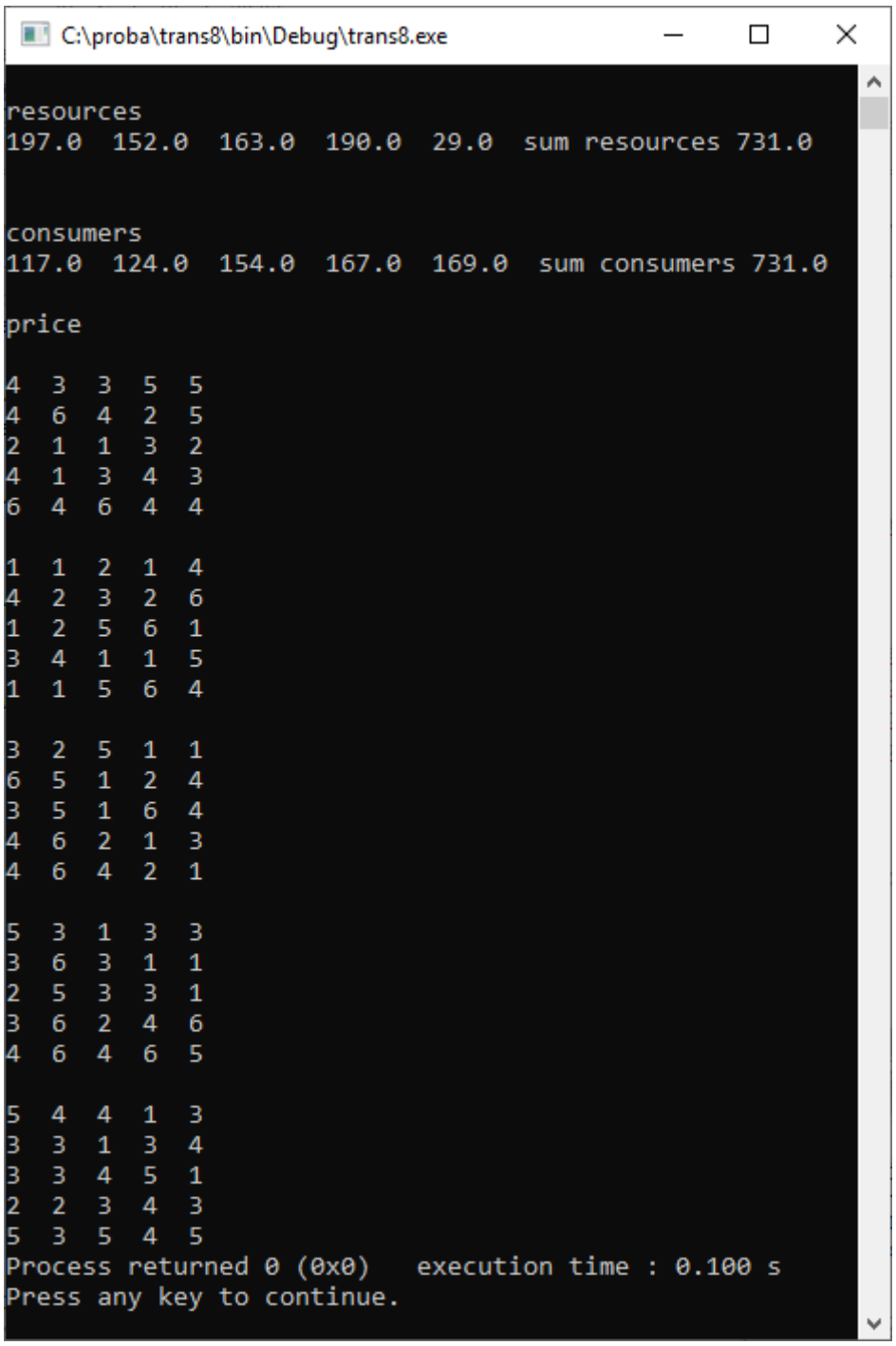


Рисунок 18 – Пример вывода на экран сгенерированных исходных данных

Поскольку результаты решения являются достаточно объемными, что делает неудобным представление их в виде консольного вывода в программе организован вывод в файл. Также вывод в файл методом дополнения

облегчает процесс анализа результатов при нескольких повторных запусках программы.

Подробный вывод в файл содержит полную информацию об исходных данных, результатах и времени выполнения и используется при тестировании программы.

Код вывода в файл показан на рисунке 19.

```
    //вывод в файл полный
     std::ofstream out;          // поток для записи
    out.open("outputfile.txt", std::ios::app);      // открываем файл для записи
    if (out.is_open())
    {
        out << "настройки" << std::endl;
        out << "N pricemin pricemax otmin otmax kg" << std::endl;
        out << N << "  " << pricemin << "  " <<pricemax << "  " <<otmin<< "  " << otmax<< "  " << kg
        out << "исходные данные" << std::endl;

out << "к отправке" << std::endl;
    for(size_t i = 0; i < N; ++i)
         out << resources[i]/1000 << "." << resources[i]%1000 << "  ";
out << "сумма " << accumulate(resources.begin(), resources.end(), 0)/1000 << "." << accumulate(resou
out << std::endl;
out << "к получению"  << std::endl;
    for(size_t i = 0; i < N; ++i)
         out << consumers[i]/1000 << "." << consumers[i]%1000 << "  ";
out << "сумма " << accumulate(consumers.begin(), consumers.end(), 0)/1000 << "." << accumulate(consu
out << "цены перевозки" ;
    for(size_t i = 0; i < N; ++i)
    {
     out << std::endl;
        for(size_t j = 0; j < N; ++j)
        {
         out << std::endl;
            for(size_t k = 0; k < N; ++k)
            out << price_matrix[i][j][k] << "  ";
        }
    }
out << std::endl << "время генерации исходных данных " << elapsed_ms.count() << " mks\n" << std::end

          }
    out.close();
    std::cout << "File has been written" << std::endl;
```

```
    void vivodfile3()
    {
std::ofstream out;          // поток для записи
    out.open("outputfile.txt", std::ios::app);      // открываем файл для записи
    if (out.is_open())
    {
        out << "решение 3" << std::endl;
     for(size_t i = 0; i < N; ++i)
    {
     out << std::endl;
        for(size_t j = 0; j < N; ++j)
        {
         out << std::endl;
            for(size_t k = 0; k < N; ++k)
            out << matrix[i][j][k]/1000 << "." << matrix[i][j][k]%1000<< "  ";
        }
    }
out << std::endl << "время решения 3 " << elapsed_ms.count() << " mks\n" << std::endl;
out << std::endl << "Ц.ф решения 3 " << func << std::endl;
          }
    out.close();
    std::cout << "File has been written" << std::endl;
    }
```

Рисунок 19 – Код вывода в файл

Пример подробного вывода в файл показан на рисунке 20.

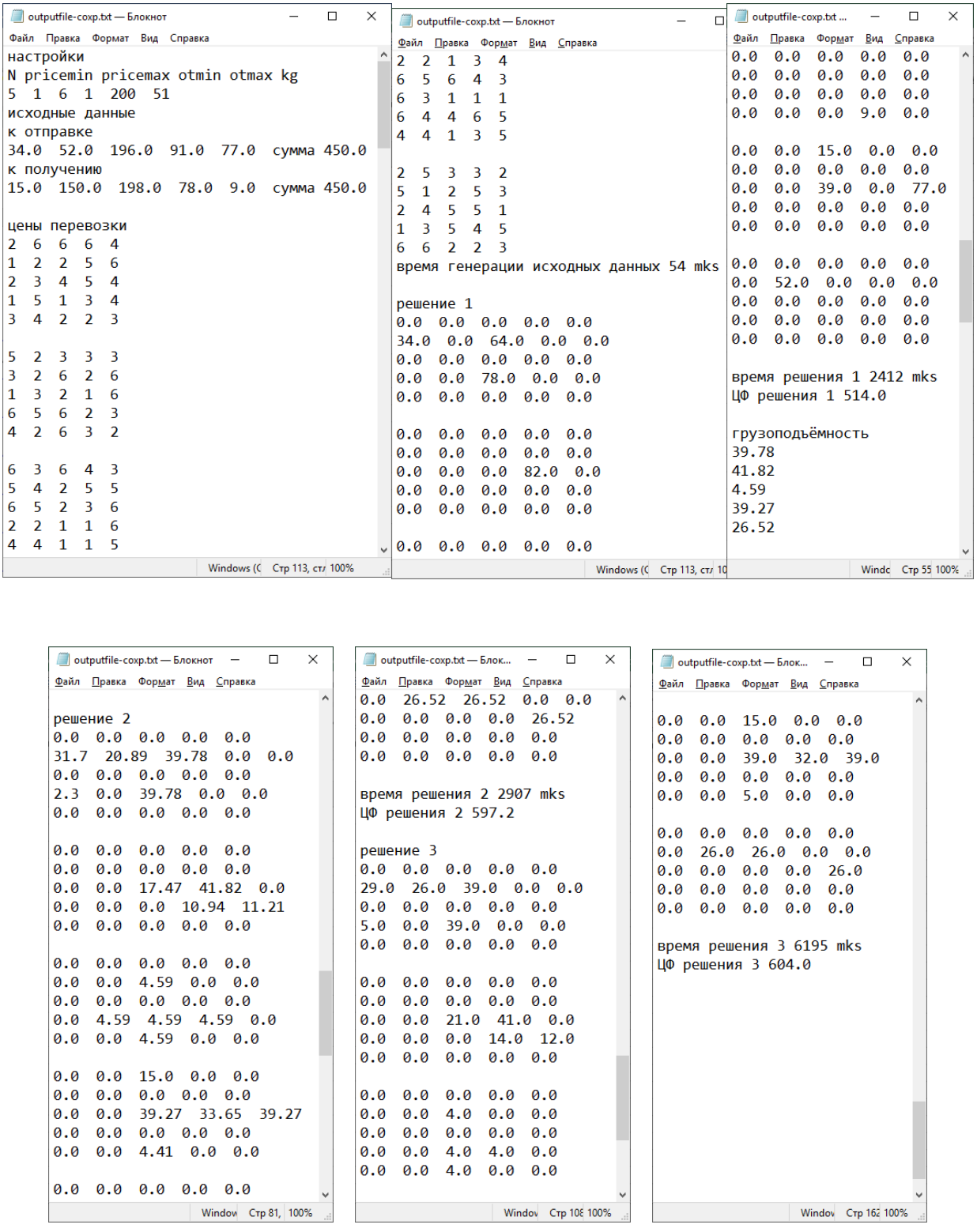


Рисунок 20 – Пример подробного вывода в файл

Для проведения компьютерного эксперимента, в ходе которого оценивается время выполнения использован сокращенный вывод в файл путем комментирования соответствующих строк в коде. Пример сокращенного вывода в файл на 3 повторных запусках программы приведен на рисунке 21.

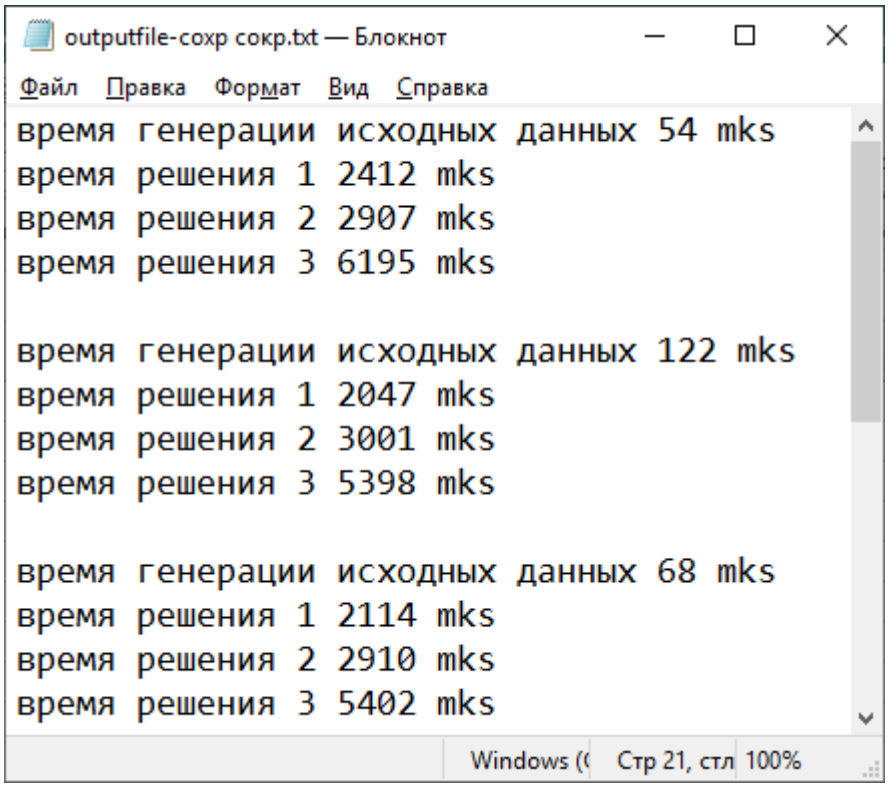


Рисунок 21 – Пример сокращенного вывода в файл

Если в Excel решение можно начинать с нулевых исходных данных, то в данном случае для запуска симплекс-метода необходимо вычислить начальное приближение, представляющее собой неоптимальное решение задачи с выполнением граничных условий – равенства количества грузов перевезенных транспортными средствами от определенного отправителя определенному получателю соответствующим величинам к отправке и к получению. Для этого использован метод северо-западного угла, реализованный кодом, приведенным на рисунке 22.

```
90      void northwestCorner()
91      {
92
93          for (int i = 0; i < this->N; i++)
94          {
95              for (int j = 0; j < this->N; j++)
96              {
97                          for (int k = 0; k < this->N; k++)
98                          {
99                  if (resources.at(i) < consumers.at(j) && resources.at(i) != 0 && consumers.at(j) != 0)
100                 {
101                     matrix[i][j][k] += resources[i];
102                     consumers[i] -= resources[i];
103                     resources[j] -= resources[i];
104                     cars[k] -= resources[i];
105
106                 }
107                 else if (resources.at(i) == consumers.at(j) && resources.at(i) != 0 && consumers.at(j) != 0)
108                 {
109                     matrix[i][j][k] = consumers[j];
110                     resources[i] -= consumers[j];
111                     consumers.erase(consumers.begin() + i);
112                     cars[k] -= resources[i];
113                 }
114                 else if (resources.at(i) > consumers.at(j) && resources.at(i) != 0 && consumers.at(j) != 0)
115                 {
116                     matrix[i][j][k] += resources.at(i) - (resources.at(i) - consumers.at(j));
117                     resources[i] -= consumers[j];
118                     consumers[j] -= consumers[j];
119                     cars[k] -= resources[i];
120                 }
121                         }
122             }
123         }
124     }
```

Рисунок 22 – Код функции начального приближения методом северо-западного угла

Результат начального приближения, представляющий собой трехмерный массив matrix, передается для оптимизации в функцию расчета симплекс-методом. В результате получается решение 1 – при целых значениях исходных данных по отправителям и получателям оно также является целочисленным. Полный код программы приведен в приложении А.

Чтобы обеспечить нецелочисленное решение необходимо внести дополнительное ограничение по грузоподъёмности такое, чтобы предельные грузоподъёмности каждого транспортного средства были нецелыми. Нецелочисленное решение необходимо обеспечить для того чтобы транспортная задача содержала проблему целочисленности.

Расчет грузоподъёмности транспортных средств выполняется исходя из максимальных величин перевозимого товара, полученных при решении 1 и

коэффициента грузоподъёмности. Предельные грузоподъёмности перезаписываются в вектор cars. После этого выполняется расчет симплекс-методом с ограничением по грузоподъёмности и получается нецелочисленное распределение грузов по транспортным средствам – решение 2. Решение 2 является промежуточным результатом, необходимым для того чтобы продемонстрировать решение проблемы целочисленности.

Третьим и заключительным расчетом является расчет методом Гомори, обеспечивающий целочисленное решение при нецелочисленных грузоподъёмностях транспортных средств.

3.2 Тестирование приложения

Фрагмент файла вывода на тестировании времени генерации исходных данных показан на рисунке 23.

Результаты тестирования показаны в таблице 1

Таблица 1 - Результаты тестирования времени генерации исходных данных, мкс

Параметр	N=5	N=10	N=50	N=100	N=500
повтор 1	45	124	7226	40387	4318966
повтор 2	57	395	7777	62752	4290488
повтор 3	67	1082	5271	36746	4199283
повтор 4	56	2049	6022	38982	4130837
повтор 5	168	144	8379	60828	4070321
повтор 6	71	654	22524	65734	4245349
повтор 7	144	557	6963	43182	4099059
повтор 8	3378	536	9042	93919	4105178
повтор 9	48	84	6040	1828099	4181276
повтор 10	93	1056	8187	45619	4691730
среднее	412,7	668,1	8743,1	231624,8	4233249
превышение максимального от среднего по девяти другим опытам	40,59	3,98	3,12	33,70	1,12

В ходе тестирования необходимо проверить работоспособность генератора получателей, который работает в бесконечном цикле с выходом по условию равенства суммы к получению сумме к отправке. Теоретически возможно, что на больших размерах задачи решение по подбору случайного вектора в сумме дающего определенное число может быть чрезмерно долгим. Для проверки данного утверждения на практике проведено по 10 повторов на различных размерах задачи с выводом информации в файл.

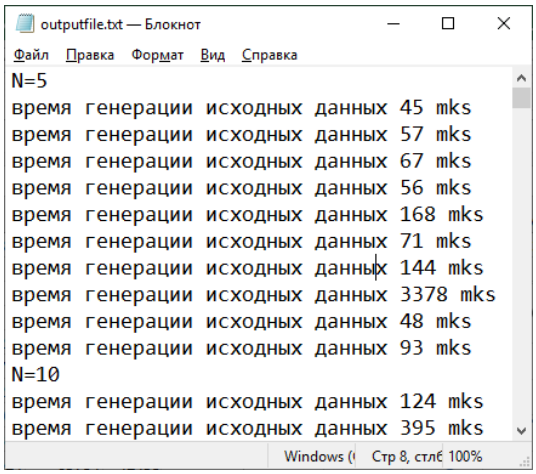


Рисунок 23 - Фрагмент файла вывода на тестировании времени генерации исходных данных

Для анализа результатов экспериментов выполнен расчет средних и оценка величины выбросов как относительное (количество раз) превышение максимального значения из десяти опытов над средним значением по остальным девяти опытам.

Выброс – это характеристика времени работы одного из используемых в программе генераторов, которая потенциально может повлиять на работоспособность программы.

Зависимость среднего времени расчета (рисунок 24) удовлетворительно линеаризуется в двойных логарифмических координатах.

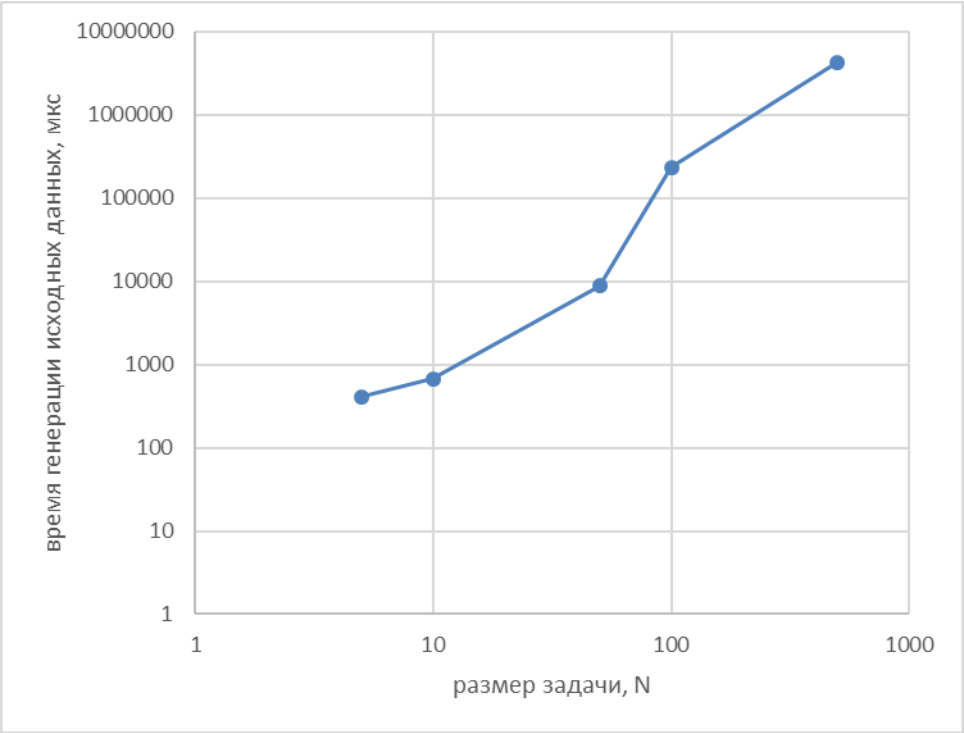


Рисунок 24 - Зависимость среднего времени расчета от размера задачи

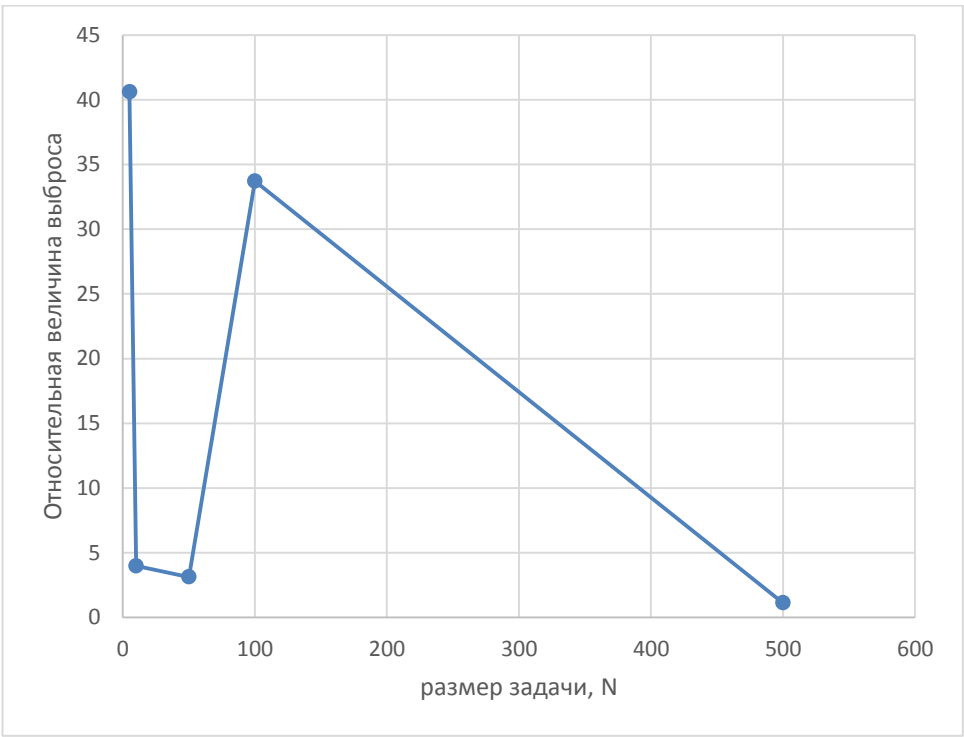


Рисунок 25 - Зависимость относительной величины выброса от размера задачи

График зависимости относительной величины выброса от среднего значения, показанный на рисунке 25, может послужить основанием для предположения о том, что с ростом размера задачи предельная величина выброса снижается

В целом по результатам тестирования генератора исходных данных можно сделать вывод о его пригодности для целей исследования.

Для тестирования на правильность расчета в модель прототипа в Excel разработанную в разделе 2 введены данные расчета задачи размера 5 приведенные на рисунке 20. Поскольку и симплекс-метод и метод Гомори относятся к точным методам оптимизации, то в результате тестирование следует ожидать полного соответствия результатов, рассчитанных в Excel и в программе на C++ при условии идентичности исходных данных.

Результаты расчета в Excel показанные в таблицах 2, 3, 4 совпадают с результатами расчета в программе на C++, показанными на рисунке 20, что свидетельствует о том, что разработанная программа работает корректно.

Таблица 2 – Результаты расчета 1 в Excel (целевая функция 514)

Номер транспортного средства	Номер получателя	Отправитель 1	Отправитель 2	Отправитель 3	Отправитель 4	Отправитель 5
1	1	0	0	0	0	0
1	2	34	0	64	0	0
1	3	0	0	0	0	0
1	4	0	0	78	0	0
1	5	0	0	0	0	0
2	1	0	0	0	0	0
2	2	0	0	0	0	0
2	3	0	0	0	82	0
2	4	0	0	0	0	0
2	5	0	0	0	0	0
3	1	0	0	0	0	0
3	2	0	0	0	0	0
3	3	0	0	0	0	0
3	4	0	0	0	0	0
3	5	0	0	0	9	0

Продолжение таблицы 2

Номер транспортного средства	Номер получателя	Отправитель 1	Отправитель 2	Отправитель 3	Отправитель 4	Отправитель 5
4	1	0	0	15	0	0
4	2	0	0	0	0	0
4	3	0	0	39	0	77
4	4	0	0	0	0	0
4	5	0	0	0	0	0
5	1	0	0	0	0	0
5	2	0	52	0	0	0
5	3	0	0	0	0	0
5	4	0	0	0	0	0
5	5	0	0	0	0	0

Таблица 3 – Результаты расчета 2 в Excel (целевая функция 597,2)

Номер транспортного средства	Номер получателя	Отправитель 1	Отправитель 2	Отправитель 3	Отправитель 4	Отправитель 5
1	1	0	0	0	0	0
1	2	31,7	20,89	39,78	0	0
1	3	0	0	0	0	0
1	4	2,3	0	39,78	0	0
1	5	0	0	0	0	0
2	1	0	0	0	0	0
2	2	0	0	0	0	0
2	3	0	0	17,47	41,82	0
2	4	0	0	0	10,94	11,21
2	5	0	0	0	0	0
3	1	0	0	0	0	0
3	2	0	0	4,59	0	0
3	3	0	0	0	0	0
3	4	0	4,59	4,59	4,59	0
3	5	0	0	4,59	0	0
4	1	0	0	15	0	0
4	2	0	0	0	0	0
4	3	0	0	39,27	33,65	39,27
4	4	0	0	0	0	0
4	5	0	0	4,41	0	0
5	1	0	0	0	0	0
5	2	0	26,52	26,52	0	0
5	3	0	0	0	0	26,52
5	4	0	0	0	0	0
5	5	0	0	0	0	0

Таблица 4 – Результаты расчета 3 в Excel (целевая функция 604)

Номер транспортного средства	Номер получателя	Отправитель 1	Отправитель 2	Отправитель 3	Отправитель 4	Отправитель 5
1	1	0	0	0	0	0
1	2	29	26	39	0	0
1	3	0	0	0	0	0
1	4	5	0	39	0	0
1	5	0	0	0	0	0
2	1	0	0	0	0	0
2	2	0	0	0	0	0
2	3	0	0	21	41	0
2	4	0	0	0	14	12
2	5	0	0	0	0	0
3	1	0	0	0	0	0
3	2	0	0	4	0	0
3	3	0	0	0	0	0
3	4	0	0	4	4	0
3	5	0	0	4	0	0
4	1	0	0	15	0	0
4	2	0	0	0	0	0
4	3	0	0	39	32	39
4	4	0	0	0	0	0
4	5	0	0	5	0	0
5	1	0	0	0	0	0
5	2	0	26	26	0	0
5	3	0	0	0	0	26
5	4	0	0	0	0	0
5	5	0	0	0	0	0

В целом по результатам тестирования следует вывод о том, что разработанное программное обеспечение может быть использовано для исследования проблемы целочисленности в многоиндексных транспортных задачах.

3.3 Экспериментальное исследование решения проблемы целочисленности в многоиндексной транспортной задаче.

Цель проведения вычислительного эксперимента – исследовать зависимость времени расчёта от размера задачи.

Уровни варьирования размера задачи N = 5, 10, 50.

Оцениваемые параметры:

– время расчёта задачи 1, T1, мс;

– время расчёта задачи 2, T2, мс;

– время расчёта задачи 3, T3, мс.

Метод оценки – с использованием библиотеки chrono по результатам вывода в текстовый файл.

Число повторов каждого опыта 3.

Результаты экспериментов показаны в таблице 5.

Таблица 5 – Результаты вычислительного эксперимента

Размер задачи, N	Номер повтора	Время расчета, мс		
		T1	T2	T3
5	1	2	3	6
	2	2	3	5
	3	2	3	5
	среднее	2,00	3,00	5,33
10	1	312	380	841
	2	298	401	915
	3	351	347	903
	среднее	320	376	886
50	1	26479	29578	102509
	2	24447	31002	98701
	3	28123	28160	112328
	среднее	26350	29580	104513

График зависимости времени расчета от размера задачи показан на рисунке 26. По графику видно, что время расчета симплекс-методом с

ограничениями (Т2) и без ограничений (Т1) отличаются несущественно по сравнению с временем расчета методом Гомори (Т3).

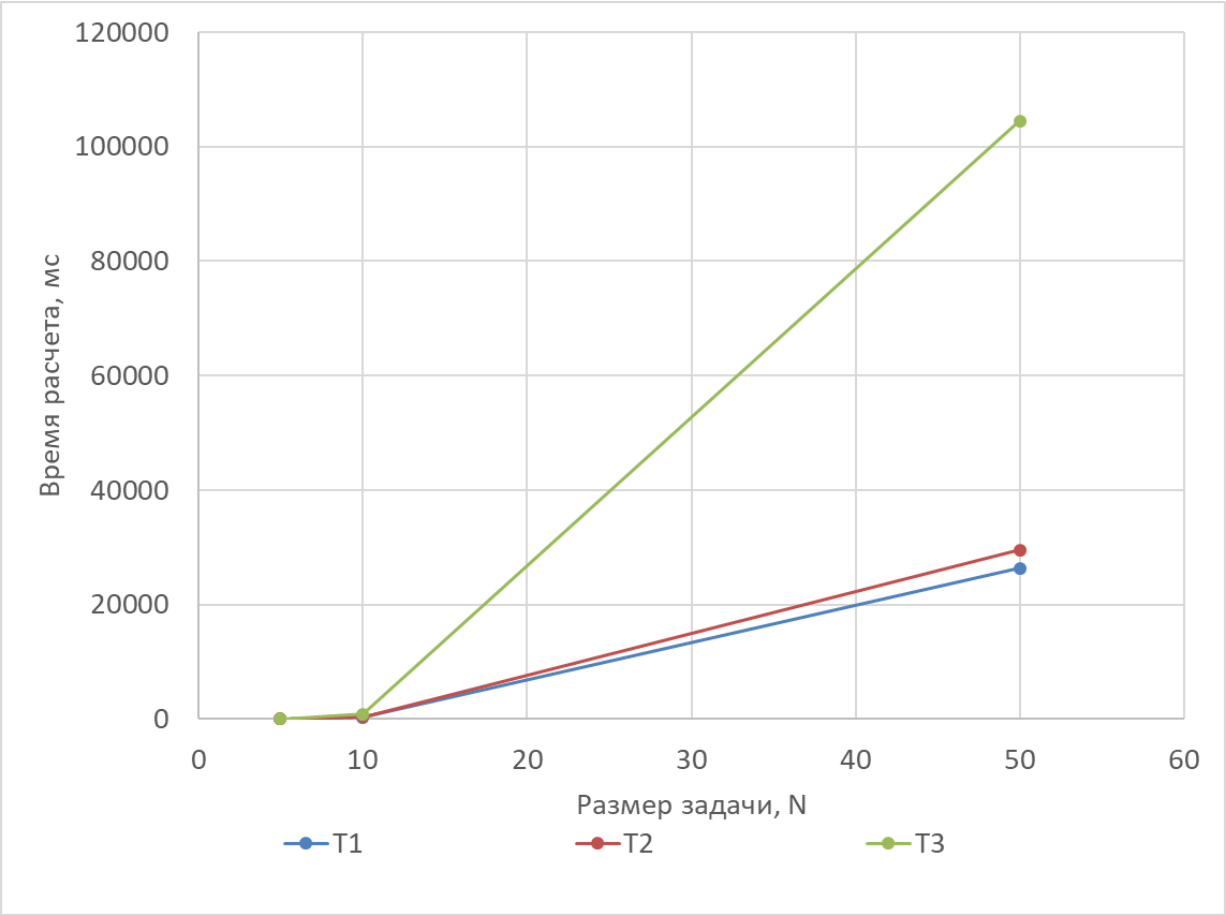


Рисунок 26 - График зависимости времени расчета от размера задачи

Выводы по разделу:

В третьем разделе выполнены разработка и тестирование приложения для решения проблемы целочисленности в многоиндексной транспортной задаче. С применением разработанного приложения выполнен компьютерный эксперимент, получены зависимости времени выполнения от размера задачи.

Построен график зависимости времени расчета от размера задачи.

Заключение

В ходе анализа предметной области рассмотрены основные виды транспортных задач и определены их отличительные черты в классе задач линейного программирования. Изучены виды и области применения многоиндексных транспортных задач. В результате выполнено описание транспортной задачи, изучены методы решения многоиндексных транспортных задач, определены исследуемые методы – симплекс метод и метод Гомори.

По результатам предварительного исследования проблемы целочисленности многоиндексных транспортных задач на прототипе программного обеспечения в MS Excel установлено, что при отсутствии дополнительных ограничений проблема целочисленности многоиндексной транспортной задачи не возникает. Определены правила формирования дополнительных ограничений, при которых проблема целочисленности в решении многоиндексной транспортной задачи имеет место.

В ходе проектирования ПО для решения проблемы целочисленности в многоиндексной транспортной задаче выполнена постановка задачи на разработку, выбор средств разработки, разработан алгоритм решения задачи, исследован прототипа приложения в MS Excel. В результате моделирования работы приложения и генераторов исходных данных на прототипе в программе MS Excel установлена пригодность прототипа для использования при тестировании разрабатываемого приложения.

Разработано программное обеспечение, позволяющее решить проблему целочисленности. Проведено тестирование разработанного ПО. В частности, при тестировании генератора данных поставщиков установлено, что нестабильность времени расчета имеет место, но практически не оказывает существенного влияния на работоспособность приложения. С использованием разработанного ПО выполнен вычислительный эксперимент по исследованию влияния размера задачи на время расчета.

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: 30.04.2023).
2. Библиотека времени Chrono [Электронный ресурс]. URL: https://inf-w.ru/?p=8305 (дата обращения: 30.04.2023).
3. Википедия. Свободная энциклопедия. algorithm (C++) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Algorithm_(C%2B%2B) (дата обращения: 30.04.2023).
4. Википедия. Свободная энциклопедия. fstream [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Fstream (дата обращения: 30.04.2023).
5. Википедия. Свободная энциклопедия. Iostream [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Iostream (дата обращения: 30.04.2023).
6. Википедия. Свободная энциклопедия. time.h [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Time.h (дата обращения: 30.04.2023).
7. Википедия. Свободная энциклопедия. vector (C++) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Vector_(C%2B%2B) (дата обращения: 30.04.2023).
8. Подробный разбор симплекс-метода [Электронный ресурс]. URL: https://habr.com/ru/post/474286/ (дата обращения: 30.04.2023).
9. 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.
10. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. Учебное пособие. – М.: Физмалит. 2007

11. Смирнов А.В. Задача целочисленного сбалансирования трехмерной матрицы и сетевая модель // Моделирование и анализ информационных систем. 2009. Т. 16. № 3. С. 70–76.

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

13. 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.

14. Chrono на C ++ [Электронный ресурс]. URL: https/www.geeksforgeeks.org/chrono-in-c/ (дата обращения: 30.04.2023).

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

16. 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

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

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

19. 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

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

Приложение А

Полный код программы

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
#include <iomanip>
#include <time.h> // для генератора случайных
#include <bits/stdc++.h> // для суммирования элементов вектора
#include <fstream> // вывод в файл
#include <chrono> // замер времени
const int N = 5; // размер задачи
const int pricemin = 1; // минимальная цена превозки
const int pricemax = 6; // максимальная цена превозки
const int otmin = 1; // минимальное количество к отправке или получению
const int otmax = 200; // максимальное  к отправке или получению
const int kg = 51;//процент грузоподьемности от максимума
int main()
{
auto begin = std::chrono::steady_clock::now();
   std::vector<std::vector<std::vector<int> > > price_matrix(N);
   std::vector<int> resources(N);
   std::vector<int> consumers(N);
   std::vector<int> cars(N);
   // Заполнение
  srand(time(NULL));
     for(size_t i = 0; i < N; ++i) // генераторы ресурсов машин и цен
   {
      resources[i] = (rand()%(otmax - otmin + 1) + otmin)*1000;
      cars[i] = 1000;
      price_matrix[i].resize(N);
      for(size_t j = 0; j < N; ++j)
      {
         price_matrix[i][j].resize(N);
               for(size_t k = 0; k < N; ++k)
         price_matrix[i][j][k] = rand()%(pricemax - pricemin + 1) + pricemin;
      }
   }
// генератор получателей
do
{
   for(size_t i = 0; i < N; ++i)
   {
      consumers[i] = (rand()%(otmax - otmin + 1) + otmin)*1000;
   }
} while (accumulate(resources.begin(), resources.end(), 0)!= accumulate(consumers.begin(),
consumers.end(), 0));

auto end = std::chrono::steady_clock::now();
auto elapsed_ms = std::chrono::duration_cast<std::chrono::microseconds>(end - begin);

// Вывод на экран отправителей
std::cout << std::endl;
std::cout << "resources" ;
```

```
std::cout << std::endl;
   for(size_t i = 0; i < N; ++i)
       std::cout << resources[i]/1000 << "." << resources[i]%1000 << "  ";
std::cout << "sum resources " << accumulate(resources.begin(), resources.end(), 0)/1000 << "."
<< accumulate(resources.begin(), resources.end(), 0)%1000 << std::endl<< std::endl;
// Вывод на экран получателей
std::cout << std::endl;
std::cout << "consumers" ;
std::cout << std::endl;
   for(size_t i = 0; i < N; ++i)
       std::cout << consumers[i]/1000 << "." << consumers[i]%1000 << "  ";
std::cout << "sum consumers " << accumulate(consumers.begin(), consumers.end(), 0)/1000 <<
"." << accumulate(consumers.begin(), consumers.end(), 0)%1000 << std::endl<< std::endl;
//Вывод на экран цен перевозки
std::cout << "price" ;
   for(size_t i = 0; i < N; ++i)
   {
    std::cout << std::endl;
      for(size_t j = 0; j < N; ++j)
      {
       std::cout << std::endl;
          for(size_t k = 0; k < N; ++k)
          std::cout << price_matrix[i][j][k] << "  ";
      }
   }
   //вывод в файл полный
    std::ofstream out;          // поток для записи
   out.open("outputfile.txt", std::ios::app);      // открываем файл для записи
   if (out.is_open())
   {
      out << "настройки" << std::endl;
      out << "N pricemin pricemax otmin otmax kg" << std::endl;
      out << N << "  " << pricemin << "  " <<pricemax << "  " <<otmin<< "  " << otmax<< "  "
<< kg << std::endl;
      out << "исходные данные" << std::endl;
out << "к отправке" << std::endl;
   for(size_t i = 0; i < N; ++i)
       out << resources[i]/1000 << "." << resources[i]%1000 << "  ";
out << "сумма " << accumulate(resources.begin(), resources.end(), 0)/1000 << "." <<
accumulate(resources.begin(), resources.end(), 0)%1000 << std::endl<< std::endl;
out << std::endl;
out << "к получению"  << std::endl;
   for(size_t i = 0; i < N; ++i)
       out << consumers[i]/1000 << "." << consumers[i]%1000 << "  ";
out << "сумма " << accumulate(consumers.begin(), consumers.end(), 0)/1000 << "." <<
accumulate(consumers.begin(), consumers.end(), 0)%1000 << std::endl<< std::endl;
out << "цены перевозки" ;
   for(size_t i = 0; i < N; ++i)
   {
    out << std::endl;
      for(size_t j = 0; j < N; ++j)
      {
       out << std::endl;
          for(size_t k = 0; k < N; ++k)
          out << price_matrix[i][j][k] << "  ";
```

```
   }
out << std::endl << "время генерации исходных данных " << elapsed_ms.count() << " mks\n"
<< std::endl;
        }
   out.close();
   std::cout << "File has been written" << std::endl;
   return 0;
}        void northwestCorner()
         {

                  for (int i = 0; i < this->N; i++)
                  {
                           for (int j = 0; j < this->N; j++)
                           {
                                                for (int k = 0; k < this->N; k++)
                  {
                                    if (resources.at(i) < consumers.at(j) && resources.at(i) != 0 &&
consumers.at(j) != 0)
                                    {
                                             matrix[i][j][k] += resources[i];
                                             consumers[i] -= resources[i];
                                             resources[j] -= resources[i];
            cars[k] -= resources[i];

                                    }
                                    else if (resources.at(i) == consumers.at(j) && resources.at(i) != 0
&& consumers.at(j) != 0)
                                    {
                                             matrix[i][j][k] = consumers[j];
                                             resources[i] -= consumers[j];
                                             consumers.erase(consumers.begin() + i);
                                             cars[k] -= resources[i];
                                    }
                                    else if (resources.at(i) > consumers.at(j) && resources.at(i) != 0
&& consumers.at(j) != 0)
                                    {
                                             matrix[i][j][k]  +=  resources.at(i)  -  (resources.at(i)  -
consumers.at(j));
                                             resources[i] -= consumers[j];
                                             consumers[j] -= consumers[j];
                                             cars[k] -= resources[i];
                                    }
                                    }
                           }
                  }
         }
//Симплекс-метод
         vector <vector <vector <int>>> free_matrix = 0;
         vector <vector <vector <char>>> matrix_plus_minus = '0';
         void mininmumCost()
         {
                  while(IsItEmpty(resources) || IsItEmpty(consumers))
                  {
                           for(int point = 1; point < N; point++)
                           {
```

```
for (int i = 0; i < N; i++)
                                {
                                        for (int j = 0; j < N; j++)
                                        {
                                                for (int k = 0; k < N; k++)
                {
                                                if(price_matrix[i][j][k] == point)
                                                {
                                                        if (resources.at(i) < consumers.at(j) &&
resources.at(i) != 0 && consumers.at(j) !=0)
                                                        {
                                                                matrix[i][j][k] += resources[i];
                                                                consumers[j] -= resources[i];
                                                                resources[i] -= resources[i];
                                                                cars[k] -= cars[i];
                                                        }
                                                        else if (resources.at(i) == consumers.at(j)
&& resources.at(i) != 0 && consumers.at(j) != 0)
                                                        {
                                                                matrix[i][j] = consumers[j];
                                                                resources[i] -= consumers[j];
                                                                cars[k] -= cars[i];

        consumers.erase(consumers.begin() + i);
                                                        }
                                                        else if(resources.at(i) > consumers.at(j)
&& resources.at(i) != 0 && consumers.at(j) != 0)
                                                        {
                                                                matrix[i][j] += resources.at(i) -
(resources.at(i) - consumers.at(j));

                                                                resources[i] -= consumers[j];
                                                                consumers[j] -= consumers[j];
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }

        bool IsItEmpty(vector<int> &vec)
        {
                for (auto c : vec)
                {
                        if (c > 0)
                        {
                                return 1;
                        }
                }
                return 0;
        }
        void fillMatrix(vector <int> matrix)
        {
                int z = 0;
```

```
for (int i = 0; i < this->N; i++) {
                        vector <int> tmp_vec;
                        for (int j = 0; j < this->N; j++) {
                                tmp_vec.push_back(matrix[z++]);
                        }
                        this->matrix.push_back(tmp_vec);
                }
        }

   // Метод Гомори
        bool potentials() {
                vector <bool> boolU(N) = true;
                vector <bool> boolV (N) = false;
                vector <int> u_vec = 0;
                vector <int> v_vec = 0;
                while (IsItTrue(boolU) || IsItTrue(boolV))
                {
                        for (int i = 0; i < N; i++)
                        {
                                for (int j = 0; j < N; j++)
                                {
                                for (int k = 0; k < N; k++)
                                {
                                        if(matrix[i][j][k] != 0)
                                        {
                                                if(boolU[i] != false)
                                                {
                                                        v_vec[j] = price_matrix[i][j] - u_vec[i];
                                                        boolV[j] = true;
                                                }
                                                if(boolV[j] != false)
                                                {
                                                        u_vec[i] = price_matrix[i][j] - v_vec[j];
                                                        boolU[i] = true;
                                                }
                                        }
                                }
                        }
                }
                if(delta(u_vec, v_vec))
                {
                        return 1;
                }
                return 0;
        }
        bool delta(vector<int> u_vec, vector<int> v_vec)
        {
                int negative = 0;
                for(int i = 0; i < N; i++)
                {
                        for( int j = 0; j < N; j++)
                        {
                        for( int k = 0; k < N; k++)
                        {
```

```
	if(matrix[i][j][k] == 0)
				{
					this->free_matrix[i][j][k] = price_matrix[i][j][k] - (u_vec[i] + v_vec[j]);
					//нахождение наибольшего отрицательного значения
					if(free_matrix[i][j][k]<0 && free_matrix[i][j][k] < negative)
					{
						negative = free_matrix[i][j][k];
					}
				}
			}
		}
		// проверка существование отрицательных значений
		if(negative<0)
		{
			//vector <int> ::iterator it = min_element(negative_vec.begin(), negative_vec.end());
			for (int i = 0; i < N; i++)
			{
				for (int j = 0; j < N; j++)
				{
				for (int k = 0; k < N; k++)
				{
					if(negative == free_matrix[i][j][k])
					{
						//cout <<setw(10) <<negative<<endl;
						this->potential_i = i;
						this->potential_j = j;
						this->potential_k = k;
						return 1;
					}
				}
			}
		}
		return 0;
	}

	bool IsItTrue(vector<bool> &bool_vec)
	{
		for(auto c : bool_vec)
		{
			if(c == false)
			{
				return 1;
			}
		}
		return 0;
	}
	void target_func() // подсчёт целевой функции
	{
		this->func = 0;
		for(int i = 0; i <N; i++)
		{
```

```
for (int j = 0; j < N; j++)
                {
                for (int k = 0; k < N; k++)
                {
                        if(matrix[i][j][k] != 0)
                        {
                                this->func += matrix[i][j][k] * price_matrix[i][j][k];
                        }
                }
        }
        cout << " F = " << this->func<<endl;
}
void min_func() // нахождение минимального элемента
{
        for (int i = 0; i < N; i++)
        {
                for (int j = 0; j < N; j++)
                {
                for (int k = 0; j < N; j++)
                {
                        if(matrix_plus_minus[i][j][k] == '-')
                        {
                                this->min_vec.push_back(matrix[i][j][k]);
                        }
                }
        }
        vector <int> ::iterator it = min_element(min_vec.begin(), min_vec.end());
        minimal_element = *it;
        min_vec.clear();
}
void searchLoop(int i, int j) // поиск цикла
{
        matrix_plus_minus[i][j][k] = '+';
        if ((j != 0) && !way)
        {
                go_left(i, j - 1);
        }
        if ((j != N - 1) && !way)
        {
                go_right(i, j + 1);
        }
        if ((i != 0) && !way) {
                go_up(i - 1, j);
        }
        if ((i != N - 1) && !way)
        {
                go_down(i + 1, j);
        }
        way = false;
        symbol = '-';
}
void go_down(int i, int j)
{
        if ((i == potential_i) && (j == potential_j))
```

```
{
		way = true;
	}
	if ((i != this->N - 1) && !way)
	{
		go_down(i + 1, j);
	}
	if ((j != 0) && (matrix[i][j][k] != 0) && !way)
	{
		go_left(i, j - 1);
		if (way and matrix[i][j][k] != '0')
		{
			matrix_plus_minus[i][j][k] = symbol;
			if (symbol == '+') { symbol = '-'; }
			else { symbol = '+'; }
		}
	}
	if ((j != this->N - 1) && (matrix[i][j][k] != 0) && !way)
	{
		go_right(i, j + 1);
		if (way and matrix[i][j][k] != '0')
		{
			matrix_plus_minus[i][j][k] = symbol;
			if (symbol == '+') { symbol = '-'; }
			else { symbol = '+'; }
		}
	}
}

void go_up(int i, int j)
{
	if ((i == potential_i) && (j == potential_j))
	{
		way = true;
	}
	if ((i != 0) && !way)
	{
		go_up(i - 1, j);
	}
	if ((j != 0) && (matrix[i][j][k] != 0) && !way)
	{
		go_left(i, j - 1);
		if (way and matrix[i][j][k] != '0')
		{
			matrix_plus_minus[i][j][k] = symbol;
			if (symbol == '+') { symbol = '-'; }
			else { symbol = '+'; }
		}
	}
	if ((j != this->N - 1) && (matrix[i][j][k] != 0) && !way)
	{
		go_right(i, j + 1);
		if (way and matrix[i][j][k] != '0')
		{
```

```
		matrix_plus_minus[i][j][k] = symbol;
				if (symbol == '+') { symbol = '-'; }
				else { symbol = '+'; }
			}
		}
}

void go_right(int i, int j)
{
		if ((i == potential_i) && (j == potential_j))
		{
			way = true;
		}
		if ((j != this->N - 1) && !way)
		{
			go_right(i, j + 1);
		}
		if ((i != 0) && (matrix[i][j][k] != 0) && !way)
		{
			go_up(i - 1, j);
			if (way and matrix[i][j][k] != '0')
			{
				matrix_plus_minus[i][j][k] = symbol;
				if (symbol == '+') { symbol = '-'; }
				else { symbol = '+'; }
			}
		}
		if ((i != this->N - 1) && (matrix[i][j][k] != 0) && !way)
		{
			go_down(i + 1, j);
			if (way and matrix[i][j][k] != '0')
			{
				matrix_plus_minus[i][j][k] = symbol;
				if (symbol == '+') { symbol = '-'; }
				else { symbol = '+'; }
			}
		}
}
void go_left(int i, int j)
{
		if ((i == potential_i) && (j == potential_j))
		{
			way = true;
		}
		if ((j != 0) && !way)
		{
			go_left(i, j - 1);
		}
		if ((i != 0) && (matrix[i][j][k] != 0) && !way)
		{
			go_up(i - 1, j);
			if (way and matrix[i][j][k] != '0')
			{
				matrix_plus_minus[i][j][k] = symbol;
```

```
                        if (symbol == '+') { symbol = '-'; }
                                else { symbol = '+'; }
                        }
                }
                if ((i != this->N - 1) && (matrix[i][j][k] != 0) && !way)
                {
                        go_down(i + 1, j);
                        if (way and matrix[i][j][k] != '0')
                        {
                                matrix_plus_minus[i][j][k] = symbol;
                                if (symbol == '+') { symbol = '-'; }
                                else { symbol = '+'; }
                        }
                }
        }

        void calculating()
        {
                for (int i = 0; i < N; i++)
                {
                        for (int j = 0; j < N; j++)
                        {
                                if (matrix_plus_minus[i][j][k] == '-')
                                {
                                        matrix[i][j][k] -= minimal_element;
                                }
                                else if (matrix_plus_minus[i][j][k] == '+')
                                {
                                        matrix[i][j][k] += minimal_element;
                                }
                        }
                }
                minimal_element = 0;
        }

        void start()
        {
                while (potentials())
                {
                        searchLoop(potential_i, potential_j);
                        min_func();
                        calculating();
                                        }
        }
    void vivodfile3()
        {
std::ofstream out;          // поток для записи
    out.open("outputfile.txt", std::ios::app);      // открываем файл для записи
    if (out.is_open())
    {
       out << "решение 3" << std::endl;
     for(size_t i = 0; i < N; ++i)
    {
     out << std::endl;
```

```
        for(size_t j = 0; j < N; ++j)
        {
         out << std::endl;
            for(size_t k = 0; k < N; ++k)
            out << matrix[i][j][k]/1000 << "." << matrix[i][j][k]%1000<< "  ";
        }
    }
out << std::endl << "время решения 3 " << elapsed_ms.count() << " mks\n" << std::endl;
out << std::endl << "ЦФ решения 3 " << func << std::endl;
           }
    out.close();
    std::cout << "File has been written" << std::endl;
            }
```