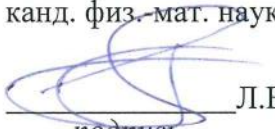


Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Механико-математический факультет

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
канд. физ.-мат. наук, декан


_____ Л.В. Гензе
подпись
« 11 » июня 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

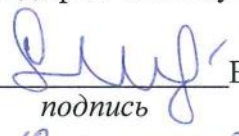
МЕТОДЫ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ

по направлению подготовки 02.03.01 Математика и компьютерные
науки

направленность (профиль) «Основы научно-исследовательской
деятельности в области математики и компьютерных наук»

Мендинов Максим Игоревич

Руководитель ВКР
канд. физ.-мат. наук, доцент


_____ Е. А. Шельмина
подпись
« 10 » июня 2021 г.

Автор работы
студент группы № 041701


_____ М. И. Мендинов
подпись
« 10 » июня 2021 г.

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

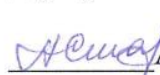
Механико-математический факультет

Кафедра вычислительной математики и компьютерного моделирования

УТВЕРЖДАЮ

Зав. кафедрой КВМ и КМ

д-р. физ.-мат. наук профессор

 А.В. Старченко

" 15 " сентября 2020г.

ЗАДАНИЕ

по бакалаврской работе студенту Мендинову Максиму Игоревичу

Тема работы: " **Методы решения транспортной задачи**"

Сроки защиты: июнь 2021 г

1) Содержание работы (вопросы, подлежащие разработке):

- a) Изучение литературы: по темам «Методы решения транспортной задачи», «Языки программирования высокого уровня и среды разработки».
- b) Постановка транспортной задачи.
- c) Изучение методов решения транспортной задачи: метод северо-западного угла, метод минимального элемента, метод аппроксимации Фогеля, метод потенциалов.
- d) Выбор языка программирования и среды разработки для создания приложения.
- e) Разработка компьютерного приложения и компьютерный эксперимент на модельных данных.
- f) Отладка и тестирование приложения.
- g) Проведение эксперимента на реальных данных. Получение и анализ результатов.
- h) Написание бакалаврской работы и подготовка к защите.

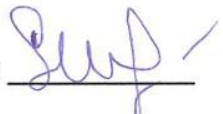
2. Сроки выполнения основных этапов работы:

- a) и b) с 15 сентября- по 15 октября 2020г;
- c) 15 октября – 15 ноября 2020г;
- d) 15 ноября – 20 декабря 2020г;
- e) февраль – март 2021г;
- f) март – апрель 2021г;
- g) апрель – май 2021г;
- h) май 2021г.

Студент обязан являться к руководителю выпускной квалификационной работы с отчетом о проделанной работе и для получения консультаций не реже 1 раза в неделю.

3. Дата выдачи задания 15.09.2020

Руководитель ВКР бакалавра Шельмина Е.А., к.ф.-м.н., доцент ТГУ

Подпись руководителя 

Задание принял к выполнению 15.09.2020 Подпись студента 

Аннотация

Транспортная задача относится к одному из классов задач линейного программирования, для которой разработаны специальные методы ее решения.

Любое предприятие в настоящее время хочет уменьшить транспортные издержки. Поэтому использование методов решения транспортных задач на предприятиях, которые хотят минимизировать затраты на транспортировку и хранение товаров, является актуальным.

Целью данной работы является решение транспортной задачи различными методами и их программная реализация.

Поставлены задачи:

1. изучить математическую модель транспортной задачи;
2. построить опорный план и получить оптимальный план перевозок;
3. осуществить программную реализацию транспортной задачи;
4. протестировать приложение на задаче с известным решением.

Транспортная задача решается, как вручную, так и с использованием специальных программ. В практической части работы разобран пример решения транспортной задачи, а также было реализовано приложение, позволяющее находить опорный план и оптимальный план перевозок. Произведено тестирование приложения на задаче с известным решением, которое показало полное совпадение результатов решения.

Оглавление

Введение	3
1 Транспортная задача	6
1.1 Постановка задачи	6
2 Методы решения транспортной задачи	11
2.1 Метод северо-западного угла	12
2.2 Метод минимального элемента	14
2.3 Метод аппроксимации Фогеля	15
2.4 Улучшение опорного плана методом потенциалов.....	16
3 Пример решения транспортной задачи.....	22
4 Языки программирования и среда разработки.....	32
4.1 Языки программирования	32
4.2 Среда разработки	33
5 Логистические информационные системы (ЛИС)	35
5.1 Системы для управления перевозками	36
6 Программная реализация решения транспортной задачи.....	40
Заключение	48
Список использованных источников и литературы	49
Приложение А	52

Введение

Принятие решений относится к важным частям различного рода управления. Люди всегда сталкивались с проблемой принятия решений в различных ситуациях, причем решений таких, которые приводят к более благоприятному результату, т.е. с использованием рационального способа решения. Такой способ мы будем называть оптимальным. С точки зрения математического описания под принятием решения понимается выбор элемента x из некоторого множества X по некоторому заданному правилу выбора F , что $F(y) \rightarrow x, \forall y \in X$. Математическая теория принятия решений занимается анализом процесса принятия решений во всех областях жизнедеятельности [10].

Линейное программирование является разделом математического программирования, в котором изучают теорию и методы решения экстремальных задач, связанных с оптимизацией линейных функций при линейной зависимости между переменными. Такие задачи являются первыми хорошо исследованными задачами поиска экстремума целевой функции, т.е. линейной функции, зависящей от переменных $x = (x_1, x_2, \dots, x_n)$ при условии, что переменные x удовлетворяют системе ограничений-равенств и неравенств. В 1949 американский математик А. Данциг создал действенный численный метод решения задач линейного программирования, называемый симплекс-методом, который стал основным методом в решении таких задач. После появления вычислительной техники были разработаны различные алгоритмы и методы для решения множества задач, связанных с оптимизацией производства при использовании ограниченных ресурсов. Линейное программирование применяется в различных задачах оптимизации и различных областях экономики, сельского хозяйства, военного дела, транспорта и многих других.

Транспортной задачей называют отдельный класс задач линейного программирования с общей математической моделью, которые могут решаться с помощью симплекс-метода, но для больших объемов данных и

специфической математической модели транспортной задачи затруднительно получить решение симплекс-методом. Для решения транспортных задач были созданы специальные численные методы, которые являются простыми в отличие от методов решения общей задачи линейного программирования. Реализация таких методов происходит либо вручную, если задача не очень сложная и входных данных немного, либо используется вычислительная техника и различные языки программирования, поскольку зачастую требуется осуществлять большое количество итераций для нахождения оптимального решения. Как и для любой задачи линейного программирования, сначала находится опорный план, например методом северо-западного угла, методом минимального элемента или методом аппроксимации Фогеля, и затем его оптимизируют. На данный момент существует множество различных методов поиска оптимального решения транспортной задачи, самый распространенный из них метод потенциалов, разработанный советскими учеными Л.В. Канторовичем и М.К. Гавуриным в [11].

Классическая транспортная задача – это задача нахождения некоторого плана перевозок однородного груза (груз, который можно перевозить одним и тем же составом) из пунктов отправления (от поставщиков) в пункты назначения (к потребителям), при котором суммарная стоимость затрат на перевозки минимальна. Такой план является экономически эффективным. При этом учитывают некоторые ограничения, налагаемые на объемы грузов, имеющихся в пунктах отправления, и ограничения, учитывающие потребность грузов в пунктах назначения. Начальные условия принято записывать в виде таблицы [1].

Также математическая модель транспортной задачи позволяет описывать множество других ситуаций, связанных не только с перевозками.

В современном мире, в условиях развития рыночной экономики, любое предприятие хочет уменьшить транспортные издержки. Поэтому использование методов решения транспортных задач на предприятиях,

которые хотят минимизировать затраты на транспортировку и хранение товаров, является актуальным.

Целью данной работы является решение транспортной задачи различными методами и их программная реализация. Для этого были сформулированы и решены следующие задачи:

1. изучить математическую модель транспортной задачи;
2. построить опорный план методом северо-западного угла, методом минимального элемента и методом аппроксимации Фогеля;
3. получить оптимальный план перевозок методом потенциалов;
4. выбрать язык программирования и среду разработки приложения;
5. осуществить программную реализацию транспортной задачи в виде оконного приложения;
6. протестировать приложение на задаче с известным решением.

1 Транспортная задача

1.1 Постановка задачи

Допустим, что у m поставщиков на складах $A_1, A_2, \dots, A_m$ сосредоточен запас однородной продукции в объеме $a_1, a_2, \dots, a_m$ единиц соответственно. Данную продукцию необходимо доставить n потребителям в пункты назначения $B_1, B_2, \dots, B_n$, которые заказали $b_1, b_2, \dots, b_n$ единиц этой продукции соответственно. За $c_{i,j}$ обозначим стоимость перевозки единицы продукции от пункта отправления A_i к пункту назначения B_j . Под стоимостью перевозки может пониматься себестоимость перевозки, расстояние между пунктами, время доставки [14].

Нужно составить такой план перевозок, при котором суммарная стоимость затрат на все перевозки будет наименьшей при условии, что все запросы потребителей будут удовлетворены. Эта задача называется транспортной. Здесь суммарная стоимость перевозок – это показатель эффективности транспортной задачи по критерию стоимости [14].

Перевозки $x_{i,j}$, $i = \overline{1, m}, j = \overline{1, n}$ – это количество единиц груза, перевозимого от i -го поставщика к j -му потребителю. Совокупность перевозок $x = \{x_{i,j}\}$ называется планом перевозок, а $x_{i,j}$ являются переменными, удовлетворяющими условиям

$$x_{i,j} \geq 0, i = \overline{1, m}, j = \overline{1, n}. \quad (1.1)$$

Так как количество единиц перевозимого груза не может быть отрицательным. Также перевозки $x_{i,j}$ удовлетворяют ограничениям, связанным с количеством перевозимого груза [14].

Количество груза, вывозимого от одного поставщика из i -го пункта отправления A_i во все пункты назначения, не может быть больше запаса продукции a_i в этом пункте, т.е.:

$$\sum_{j=1}^n x_{i,j} \leq a_i, i = \overline{1, m}. \quad (1.2)$$

Количество груза, ввозимого к одному поставщику в j -й пункта назначения B_j из всех пунктов отправления, не должно превышать заявки на продукцию b_j в этом пункте, т.е.:

$$\sum_{i=1}^m x_{i,j} \leq b_j, j = \overline{1, n}. \quad (1.3)$$

Суммарная стоимость всех перевозок определяется целевой функцией $f(x)$:

$$f(x) = \sum_{i=1}^m \sum_{j=1}^n c_{i,j} x_{i,j}. \quad (1.4)$$

и значение функции должно стремиться к минимуму, т.е.

$$f(x) \rightarrow \min. \quad (1.5)$$

Так как целевая функция (1.4) и ограничения (1.2)-(1.3) являются линейными функциями, то транспортная задача (1.1)-(1.5) является задачей линейного программирования.

Определение 1. План перевозок называем допустимым, если он удовлетворяет условиям (1.1)-(1.3).

Определение 2. Допустимый план перевозок назовем оптимальным, если соответствующая ему стоимость перевозок минимальна среди всех допустимых планов перевозок. Оптимальный план перевозок является решением транспортной задачи (1.1)-(1.5) [14].

Для решения транспортной задачи удобно составить транспортную таблицу. Так, для m поставщиков и n заказчиков в первой строке и первом столбце таблицы пишут пункты назначения B_j и пункты отправления A_i соответственно, последние строка и столбец соответствуют заявкам b_j и

запасам a_i . Нижняя правая клетка содержит суммарную заявку $\sum_{j=1}^n b_j$ и суммарный запас $\sum_{i=1}^m a_i$. Во внутренних клетках в левом верхнем углу записывается стоимость соответствующих перевозок $c_{i,j}$, а в нижнем правом углу записывается стоимость перевозки $x_{i,j}$, удовлетворяющей условиям (1.1)-(1.3).

Таблица 1 – Транспортная таблица

Пункты	B_1	...	B_j	...	B_n	Запасы	
A_1	$c_{1,1}$ $x_{1,1}$	...	$c_{1,j}$ $x_{1,j}$	...	$c_{1,n}$ $x_{1,n}$	a_1	
$\vdots$	...	...	...	...	...	$\vdots$	
A_i	$c_{i,1}$ $x_{i,1}$	...	$c_{i,j}$ $x_{i,j}$	...	$c_{i,n}$ $x_{i,n}$	a_i	
$\vdots$	...	...	...	...	...	$\vdots$	
A_m	$c_{n,1}$ $x_{n,1}$	...	$c_{n,j}$ $x_{n,j}$	...	$c_{m,n}$ $x_{m,n}$	a_m	
Заявки	b_1	...	b_j	...	b_n	$\sum_{j=1}^n b_j$	$\sum_{i=1}^m a_i$

Определение 3. Задача называется сбалансированной, а ее модель замкнутой, если

$$\sum_{j=1}^n b_j = \sum_{i=1}^m a_i. \quad (1.6)$$

Определение 4. Задача называется несбалансированной, а ее модель открытой, если

$$\sum_{j=1}^n b_j \neq \sum_{i=1}^m a_i. \quad (1.7)$$

Пусть выполняется равенство (1.6), тогда ограничения (1.2)-(1.3) представляются равенствами.

Количество продукции, отправленной от одного поставщика из i -го пункта отправления A_i во все пункты назначения, равно запасу продукции a_i в этом пункте, т.е.:

$$\sum_{j=1}^n x_{i,j} = a_i, i = \overline{1, m}. \quad (1.8)$$

Количество продукции, перевозимой к одному поставщику в j -й пункт назначения B_j из всех пунктов отправления, равно потребности на продукцию b_j в этом пункте, т.е.:

$$\sum_{i=1}^m x_{i,j} = b_j, j = \overline{1, n}. \quad (1.9)$$

Система ограничений (1.8)-(1.9) состоит из $n+m$ уравнений с mn переменными $x_{i,j}$. Условие (1.6) является причиной того, что одно из уравнений (1.8)-(1.9) линейно зависит от других. Получим, что ранг системы равен $r=m+n-1$ и сбалансированная транспортная задача должна содержать r базисных переменных. Число свободных переменных равняется $p=mn-(m+n-1)=(n-1)(m-1)$, т.е. p перевозок равняются нулю [11].

Допустимое решение (план перевозок) сбалансированной транспортной задачи можно получить, заполняя клетки таблицы 1 перевозками $x_{i,j}$, следующим образом:

- сумма перевозок строке $\sum_{j=1}^n x_{i,j}$ равно запасам в a_i единиц, т.е.

$$\sum_{j=1}^n x_{i,j} = a_i, i = \overline{1, m};$$
- сумма перевозок в каждом столбце $\sum_{i=1}^m x_{i,j}$ равняется заявке в b_j , т.е.

$$\sum_{i=1}^m x_{i,j} = b_j, j = \overline{1, n}.$$

Очевидно, что мы можем заполнить таблицу 1 различными способами. Решением является тот допустимый план перевозок, для которого суммарная стоимость будет наименьшей.

Для решения замкнутой транспортной задачи, как и в любой задаче линейного программирования, нужно сначала отыскать какой-либо начальный допустимый план перевозок, а потом построить итерационный процесс для нахождения оптимального плана перевозок. Такое решение всегда существует [14].

Теорема 2. Пусть имеется замкнутая транспортная задача и $\sum_{j=1}^n b_j = \sum_{i=1}^m a_i$, тогда она имеет решение [1].

Итак, рассмотрим сбалансированную транспортную задачу:

$$f(x) = \sum_{i=1}^m \sum_{j=1}^n c_{i,j} x_{i,j}, \quad (1.10)$$

$$\sum_{j=1}^n x_{i,j} \leq a_i, i = \overline{1, m}, \quad (1.11)$$

$$\sum_{i=1}^m x_{i,j} \leq b_j, j = \overline{1, n}, \quad (1.12)$$

$$x_{i,j} \geq 0, i = \overline{1, m}, j = \overline{1, n}, \quad (1.13)$$

где $\sum_{i=1}^m a_i = \sum_{j=1}^n b_j$.

2 Методы решения транспортной задачи

Для любых задач линейного программирования, процесс по отысканию оптимального плана начинается с опорного плана.

Определение 5. Опорное решение (опорный план) транспортной задачи – это любое возможное решение, соответствующее положительным координатам, для которого векторы условий системы (1.8)-(1.9) линейно независимы [6].

Так как ранг системы ограничений (1.8)-(1.9) равен r , то опорное решение не может иметь отличных от нуля координат более чем r . В невырожденном случае опорного плана число отличных от нуля координат равно r .

Любое решение транспортной задачи можно записать в транспортную таблицу. Клетки таблицы транспортной таблицы, в которых перевозки $x_{i,j} > 0$ называются базисными или занятыми, остальные, где $x_{i,j} = 0$ называются свободными. В транспортной таблице имеются r базисных клеток. Для проверки линейной независимости допустимого решения (1.8)-(1.9) используют циклы.

Определение 6. Циклом называется последовательность клеток транспортной таблицы $(i_1, j_1), (i_1, j_2), (i_2, j_2), \dots, (i_k, j_1)$, в которых ровно две соседние клетки находятся в одной строке (столбце), причем первая и последняя клетки находятся в одной и той же строке (столбце). Циклы также используются для перехода от одного опорного плана к другому [1].

Циклы могут иметь разнообразную форму, например, как на рисунке 1.

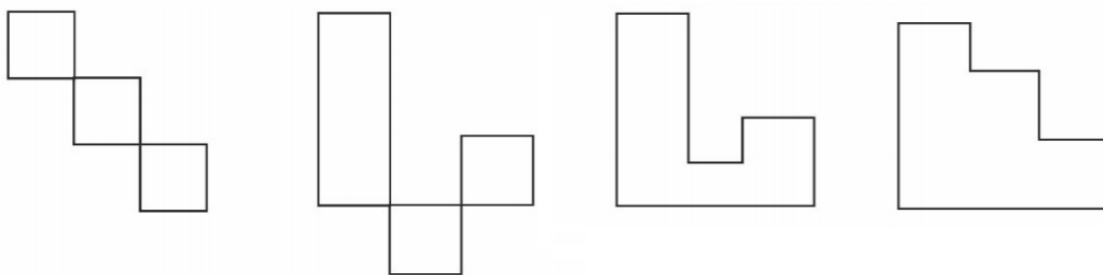


Рисунок 1 – Примеры циклов

Теорема 3. (О связи между линейной зависимостью векторов условий и циклами).

Для того чтобы система векторов условий (1.8)-(1.9) транспортной задачи была линейно зависимой, необходимо и достаточно, чтобы из соответствующих клеток таблицы можно было выделить часть, которая образует цикл [13].

Следствие. Система векторов условий (1.8)-(1.9) транспортной задачи линейно независима тогда и только тогда, когда из соответствующих им клеток таблицы нельзя образовать цикл [13].

Поэтому, допустимое решение транспортной задачи является опорным решением только тогда, когда из базисных клеток нельзя построить ни одного цикла, т.е. в таблице нельзя построить замкнутый цикл, все вершины которого лежат в занятых клетках. Построение циклов начинают с любой базисной клетки и переходят к следующей базисной клетке в строке или столбце и т.д., пытаясь вернуться к началу. Если это возможно, то получим цикл, и план не будет являться опорным.

Существует несколько методов построения опорного плана транспортной задачи. Далее рассмотрим некоторые из них.

Общим для всех методов является формула нахождения значения $x_{i,j}$ в транспортной таблице

$$x_{i,j} = \min \begin{cases} \text{остаток груза в пункте отправления } A_i \\ \text{неудовлетворенные потребности в пункте назначения } B_j \end{cases}$$

Вместо написания значения $x_{i,j} = 0$ в свободной клетке можно ставить точку [14].

2.1 Метод северо-западного угла

Метод северо-западного угла состоит в том, что один поставщик A_i , используя свои запасы, выполняет максимальное количество запросов

некоторого количества потребителей $B_j, j = 1..k, k \geq 1$, так, чтобы не были превышены заявленные возможности поставщика или потребности потребителя, после этого используются запасы A_{i+1} поставщика. На стоимость перевозки единицы продукции в этом методе не обращают внимания, так как предполагается дальнейшая оптимизация [6].

Заполнение транспортной таблицы начинается с левого верхнего угла (клетки (1,1)) и состоит из ряда однотипных шагов. На каждом шаге, исходя из запасов поставщика и запросов потребителя, заполняется только одна клетка и соответственно исключается из рассмотрения один поставщик или потребитель. Осуществляется это согласно приведенному ниже алгоритму [6].

На первом шаге $x_{1,1} := \min(a_1, b_1)$. Если $a_1 < b_1$ то $x_{1,1} := a_1$ и первая строка исключается из рассмотрения, так как запасы первого поставщика закончились, а остальные клетки строки остаются незаполненными, тогда дальше будем рассматривать $b_1 = b_1 - a_1$. Если $a_1 > b_1$, тогда $x_{1,1} := b_1$, и мы больше не рассматриваем первый столбец, потому что первый потребитель удовлетворен и не нуждается в товаре и рассматриваем $a_1 = a_1 - b_1$. Далее мы переходим на новую строку или новый столбец [1].

В общем случае получим:

1. если $a_i < b_j$, то $x_{i,j} := a_i$ и мы исключаем поставщика A_i (больше не рассматриваем строку с номером i), $x_{i,k} = 0, k = 1..n, k \neq j$, и далее $b_j = b_j - a_i$;
2. если $a_i \geq b_j$, то $x_{i,j} := b_j$ и мы исключаем потребителя B_j (больше не рассматриваем столбец с номером j), $x_{k,j} = 0, k = 1..m, k \neq i$, и далее $a_i = a_i - b_j$.

Заметим, что возможен случай, когда при вычислении очередного $x_{i,j}$ получается, что поставщик A_i или потребитель B_j имеет нулевые запасы или запросы, т.е. потребности в пункте B_j удовлетворены, а запасы в пункте A_i исчерпаны и получим, что из рассмотрения выбывает строка и столбец

одновременно. Тогда в клетку ставится базисный нуль (в ней пишется 0), остальные клетки с нулевыми перевозками остаются пустыми, и после этого исключается из рассмотрения соответствующий поставщик или потребитель. Алгоритм работает до полного распределения груза по пунктам потребителей $B_j, j = 1..n$. После заполнения должны получить r базисных клеток. План перевозок, полученный методом северо-западного угла, всегда является опорным, так как на каждом шаге мы исключаем строку или столбец, поэтому заполняемые клетки не попадут в цикл [6].

2.2 Метод минимального элемента

Метод минимального элемента заключается в том, что совершается поставка с минимальной стоимостью (при наличии одинаковых тарифов стоимости поставки выберем любой из них). Далее самая дешевая из оставшихся и так далее по возрастанию стоимости перевозки до полного удовлетворения всех заявок потребителей на продукцию [1].

Пусть клетка с минимальной ценой перевозки $c_{i,j} = \min_{k=\overline{1,m}, g=\overline{1,n}} \{c_{k,g}\}$. Тогда величина перевозки $x_{i,j} = \min(a_i, b_j)$ записывается в транспортную таблицу. Если $x_{i,j} = a_i$, то поставщик A_i полностью израсходовал все запасы и больше не рассматривается, но нужно учесть, что потребность изменилась и равна $b_j := b_j - a_i$. Если же $x_{i,j} = b_j$, то потребитель B_j исключается из рассмотрения, т.к. удовлетворил полностью свои потребности, т.е. не обращаем внимания на j -й столбец. Заметим, что у поставщика A_i остался запас $a_i = a_i - b_j$. В том случае, когда $a_i = b_j = x_{i,j}$, у поставщика A_i не осталось запасов, а потребитель B_j удовлетворен, то на следующем шаге, когда от поставщика потребуются поставить продукцию, в соответствующую клетку транспортной таблицы записывается базисный нуль и поставщик исключается из рассмотрения. Далее i -я строка и j -й столбец не рассматриваются [11].

Получим сокращенную таблицу стоимостей перевозок и снова выберем минимальную стоимость перевозки из оставшихся и определим

соответствующую ей перевозку и т.д. На каждом шаге исключается либо один пункт поставки (строка), либо один потребитель (столбец). Процесс продолжаем до тех пор, пока не получим полное распределение продукции между потребителями. Ясно, что метод минимального элемента сводится к методу северо-западного угла, если перенумеровать пункты отправления A_i и пункты назначения B_j в соответствии с порядком вычеркивания строк и столбцов. Отсюда следует, что план перевозок, построенный методом минимального элемента, приводится является. Количество базисных клеток равно r [11].

2.3 Метод аппроксимации Фогеля

Метод аппроксимации Фогеля является наиболее трудоемким, по сравнению с методом северо-западного угла или методом минимального элемента. Но в отличие от последних двух, метод аппроксимации Фогеля позволяет построить начальный план перевозок, как правило, близкий к оптимальному плану. В этом методе используются так называемые штрафы – абсолютная разность между двумя минимальными стоимостями перевозки единицы груза. Штраф берется за неудачно выбранный маршрут [1].

Суть метода состоит в нахождении штрафов по всем строкам и по всем столбцам. Среди штрафов выбирают наибольший по всем строкам и столбцам. Заполняется клетка с минимальной стоимостью перевозок, соответствующая максимальному штрафу по строке или столбцу, в зависимости от того, где наибольший штраф был найден (в строке или столбце) [1].

Заведем дополнительную строку и столбец для записи штрафов. На каждой итерации метода вычисляются штрафы d_i и D_j , $i = \overline{1, m}$, $j = \overline{1, n}$, как абсолютная разность двух наименьших стоимостей перевозок $c_{i,j}$ и записываются в специально отведенную строку и столбец. Если минимальные стоимости перевозок равны, то штраф равен 0 [8].

Далее находим наибольший штраф $\max_{i=\overline{1,m}, j=\overline{1,n}} \{d_i, D_j\}$. Затем выбираем пустую клетку для заполнения с минимальной стоимостью перевозок $\min_{i=\overline{1,m}, j=\overline{1,n}} c_{i,j}$, которая принадлежит строке/ столбцу с максимальным штрафом. На последующих итерациях эту строку/ столбец не рассматриваем [8].

Если встречаются равные максимальные штрафы, то в строках или столбцах, которые соответствуют максимальным штрафам, берется не вычеркнутая клетка с минимальной стоимостью перевозок. Также если клеток с минимальной стоимостью перевозок несколько в выбранной строке/ столбце, то выбираем такую клетку (i, j) для заполнения, которая соответствует максимальной сумме штрафов по i -ой строке и j -му столбцу [8].

Алгоритм продолжается до полного распределения груза между потребителями $B_j, j = 1..n$.

2.4 Улучшение опорного плана методом потенциалов

Итак, был получен некоторый допустимый план, но полученный опорный план не обязан являться оптимальным. Метод потенциалов предоставляет возможность получить оптимальный план перевозок в транспортной задаче на основе опорного плана, построенного, например, методами, которые были рассмотрены выше. Метод потенциалов является модификацией симплекс-метода при решении транспортной задачи. Для того чтобы улучшить план перевозок, аналогично симплекс-методу, необходимо одну свободную клетку сделать базисной, а соответствующую ей базисную клетку сделать свободной. Причем число базисных клеток останется тем же, и мы получим новый опорный план перевозок [14].

Для перехода от одного опорного плана к другому используются циклы. Выбирается свободная клетка таблицы, из которой строится цикл, включающий в себя базисные клетки. По этому циклу перераспределяются

объемы перевозок путем заполнения выбранной свободной клетки и освобождения одной из базисных клеток [14].

Цикл имеет четное число вершин и ребер, т.е. как в строке, так и в столбце транспортной таблицы содержится только четное число клеток, содержащих вершины цикла. В связи с этим фактом можно перераспределять объемы перевозок в вершинах цикла так, что суммы перевозок $\sum_{j=1}^n x_{i,j}$ и $\sum_{i=1}^m x_{i,j}$ останутся прежними. Для этого последовательно нумеруются вершины цикла, начиная с 1, и в клетках с нечетными номерами, увеличить поставку на величину $\theta \geq 0$, а в клетках с четными номерами, уменьшить поставку на эту же величину θ , то сумма по строкам и столбцам не изменится, но суммарная стоимость затрат на перевозки может измениться [14].

Определение 7. Цикл называется означенным, если его вершины пронумерованы по порядку и нечетным клеткам, в которых увеличивается значение перевозки, приписан знак «+», а четным клеткам, в которых уменьшается значение перевозки, знак «-» [6].

Определение 8. Сдвигом по циклу на величину θ называется процесс изменения перевозок, при котором увеличивается объем поставок во всех нечетных клетках цикла на величину θ , и уменьшается объем поставок во всех четных клетках цикла на величину θ [6].

Итак, имеется означенный цикл. Максимальное значение θ выбирается так, чтобы выполнялось условие неотрицательности перевозок, т.е. $x_{i,j} \geq 0, i = \overline{1, m}, j = \overline{1, n}$.

Предположим, что перевозчик получает плату u_i за вывоз единицы груза из пункта A_i и плату v_j за доставку единицы груза в пункт B_j . Сумму платежей $s_{i,j} = u_i + v_j$ будем называть псевдостоимостью перевозки единицы продукции из пункта A_i в пункт B_j . Платежи u_i и v_j также могут быть отрицательными, т.е. перевозчик сам платит за транспортировку груза. Система платежей u_i и v_j называется транспортными потенциалами пунктов поставщиков и потребителей [14].

Лемма 1. Потенциалы $\{u_i, v_j\}$ определяют суммарную псевдостоимость перевозок $\Phi(x) = \sum_{i=1}^m \sum_{j=1}^n s_{i,j} x_{i,j}$ так, что она не зависит от выбора допустимого плана [14].

Если подбирать потенциалы u_i, v_j так, что во всех базисных клетках опорного плана $c_{i,j} = u_i + v_j$, а в оставшихся свободных клетках также определять псевдостоимость $s_{i,j}$, то получим, что стоимость перевозки единицы груза равна псевдостоимости во всех базисных клетках, а в свободных клетках псевдостоимость может отличаться от реальной стоимости перевозки.

Теорема 1 (условие оптимальности опорного плана).

Пусть есть невырожденный опорный план $X = \{x_{i,j}\}$, $i = \overline{1, m}, j = \overline{1, n}$, потенциалы u_i, v_j и псевдостоимость $s_{i,j} = u_i + v_j$. Тогда, если для всех свободных клеток X выполнено $c_{i,j} \geq s_{i,j}$, то такой опорный план является оптимальным [14].

Итак, условие оптимальности опорного плана формулируется в виде следующих условий:

- во всех базисных клетках псевдостоимость должна быть равна стоимости перевозки $c_{i,j} = s_{i,j}$;
- во всех свободных клетках псевдостоимость должна быть меньше или равна стоимости перевозки $c_{i,j} \geq s_{i,j}$.

Определение 10. Относительной оценкой $\Delta_{i,j}$ назовем разность между реальной стоимостью перевозки $c_{i,j}$ и псевдостоимостью $s_{i,j}$. $\Delta_{i,j} = c_{i,j} - s_{i,j}$.

Условие оптимальности примет вид:

- во всех базисных клетках $\Delta_{i,j} = 0$;
- во всех свободных клетках $\Delta_{i,j} \geq 0$.

Иначе план можно улучшить с помощью перераспределения поставок.

Определение 9. Цена цикла q – это изменение стоимости перевозок при перемещении единицы груза по циклу [14].

Лемма 2. Для невырожденного опорного плана, имеющего потенциалы u_i, v_j , который удовлетворяет условию $c_{i,j} = u_i + v_j$ в базисных клетках, цена $q_{k,l}$ цикла, построенного из любой свободной клетки $A_k B_l$, остальные вершины которого находятся в базисных клетках, равняется относительной оценке $\Delta_{k,l}$ клетки с координатами (k, l) [14].

Лемма 2 позволяет сформулировать алгоритм построения оптимального плана. Если взять клетку с наименьшей относительной оценкой $\Delta_{k,l}$, при этом свободные клетки могут содержать отрицательную относительную оценку, составить из нее цикл, в котором кроме этой клетки все остальные являются базисными (такой цикл всегда существует и единственен), перераспределить по нему максимально возможное число груза θ , тогда суммарная стоимость затрат на перевозки снизится на величину $\theta|\Delta_{k,l}|$. Таким образом, связь между суммарными стоимостями затрат на перевозки в таблице с номером g и таблице с номером $g+1$ представляется в виде равенства $f^{g+1} = f^g + \theta\Delta_{k,l}$. Далее улучшенный план необходимо опять проверить на оптимальность. Повторять процесс необходимо до выполнения условия оптимальности [14].

Алгоритм метода потенциалов:

1. Построить первоначальный план перевозок одним из методов построения опорного плана, содержащий $m+n-1$ базисных клеток;
2. Для всех базисных клеток ($x_{i,j} > 0$) найти потенциалы u_i, v_j ($i = \overline{1, m}, j = \overline{1, n}$). Для этого необходимо составить и найти решение системы уравнений $u_i + v_j = c_{i,j}$. Так как план содержит $m+n-1$ базисных клеток, то система состоит из $m+n-1$ уравнений с $m+n$ неизвестными. Такая система имеет бесконечное множество решений. Поэтому для нахождения значений всех потенциалов, одному из них задают произвольное значение, для определенности положим $u_1 = 0$. После этого из системы остальные потенциалы вычисляются однозначно. В транспортной таблице для потенциалов заводится

дополнительная строка и столбец, куда проставляются найденные значения;

3. Для всех свободных клеток найти относительные оценки $\Delta_{i,j} = c_{i,j} - (u_i + v_j)$;
4. Проанализировать относительные оценки $\Delta_{i,j}$:
 - а) если $\Delta_{i,j} \geq 0 \forall i = \overline{1, m}, j = \overline{1, n}$, то задача считается решенной, а план является оптимальным, находим стоимость этого оптимального плана;
 - б) если среди оценок $\Delta_{i,j}$ есть хоть одна отрицательная, тогда план можно улучшить. Находим самую маленькую оценку (если имеется несколько одинаковых значений, то из них выбираем любое). Пусть это будет оценка $\Delta_{k,l}$;
5. Для этой клетки $A_k B_l$ с оценкой $\Delta_{k,l}$ построить означенный цикл. Все вершины цикла, не считая выбранной, находятся в базисных клетках. Этот цикл единственный. Знаки расставляются поочередно, начиная с выбранной клетки $A_k B_l$ со знаком "+";
6. Осуществить перераспределение продукции по циклу. Для этого нужно выбрать наименьшую поставку среди клеток со знаком «-» $\theta = \min_{i=\overline{1, m}, j=\overline{1, n}} \{x_{i,j}\}$. Выполнить сдвиг по циклу на величину θ . Клетка, которой соответствует наименьшее количество продукции θ , становится свободной. Также, если $\theta = 0$, то при сдвиге по циклу свободная клетка становится базисным нулем. Если значение θ находится в нескольких отрицательных вершинах, тогда при сдвиге по циклу нужно поставить базисный нуль в одну из этих вершин, например, в клетку с наименьшей стоимостью перевозок. Тогда число базисных клеток останется равным $m+n-1$;
7. Клетки таблицы, которые не входят в цикл, не изменяют своих значений. Далее переходим к пункту 2. Процесс продолжается до тех

пор, пока все свободные клетки не будут удовлетворять условию оптимальности.

3 Пример решения транспортной задачи

Необходимо решить транспортную задачу, заданную таблицей 2.

Таблица 2 – Пример транспортной задачи

Пункты	B_1	B_2	B_3	Запасы
A_1	8	4	5	30
A_2	3	7	2	80
A_3	1	4	6	60
Заявки	70	60	40	170

где $\sum_{i=1}^m a_i = 170$, $\sum_{j=1}^n b_j = 170$. Имеем задачу замкнутого типа. Найдем опорный план методом северо-западного угла, методом минимального элемента и методом аппроксимации Фогеля. Далее найдем оптимальный план перевозок методом потенциалов.

Решение методом северо-западного угла (таблица 3.1-таблица 3.3)

Начинаем с клетки $x_{1,1} := \min(a_1, b_1) = 30$. Тогда в пункте A_1 запасы исчерпаны, и мы больше не рассматриваем первую строку, т.е. $x_{1,2} = x_{1,3} = 0$ (в этих клетках ставим точки) и запросы $b_1 := 70 - 30 = 40$.

Таблица 3.1 – Решение транспортной задачи методом северо-западного угла

Пункты	B_1	B_2	B_3	Запасы
A_1	8 30	4 •	5 •	30
A_2	3	7	2	80
A_3	1	4	6	60
Заявки	40	60	40	170

Так как запросы первого потребителя не удовлетворены, рассматриваем $x_{2,1} := \min(a_2, b_1) = 40$. Тогда в пункте В1 потребности удовлетворены, и в клетке A_3B_1 ставим точку, также $a_2 = 80 - 40 = 40$.

Таблица 3.2 – Решение транспортной задачи методом северо-западного угла

Пункты	B_1	B_2	B_3	Запасы
A_1	8 30	4 •	5 •	0
A_2	3 40	7	2	80
A_3	1 •	4	6	60
Заявки	40	60	40	170

Продолжая этим методом, получим $x_{2,2} := 40$. Далее $x_{3,2} = 20$. Остается перевозка $x_{3,3}$, которая будет равна 40.

Таблица 3.3 – Решение транспортной задачи методом северо-западного угла

Пункты	B_1	B_2	B_3	Запасы
A_1	8 30	4 •	5 •	0
A_2	3 40	7 40	2 •	0
A_3	1 •	4 20	6 40	0
Заявки	0	0	0	170

Число занятых клеток транспортной таблицы $m + n - 1 = 5$. Следовательно, опорный план является невырожденным. Получили опорный план, суммарная стоимость всех перевозок которого равна $f(x) = 8 \cdot 30 + 3 \cdot 40 + 7 \cdot 40 + 4 \cdot 20 + 6 \cdot 40 = 960$.

Решение методом минимального элемента (таблица 4.1-таблица 4.3)

Находим клетку с минимальной ценой перевозки $\min_{k=\overline{1,3}, g=\overline{1,3}} \{c_{k,g}\} = c_{3,1} = 1$, тогда $x_{3,1} = \min(a_3, b_1) = 60$ и больше не рассматриваем поставщика A_3 . $b_1 = 70 - 60 = 10$.

Таблица 4.1 – Решение транспортной задачи методом минимального элемента

Пункты	B_1	B_2	B_3	Запасы
A_1	8	4	5	30
A_2	3	7	2	80
A_3	1 60	4 •	6 •	60
Заявки	10	60	40	170

Из оставшихся клеток находим клетку с минимальной стоимостью $c_{2,3} = 2$, для этой клетки $x_{2,3} = 40$ и потребности потребителя B_3 удовлетворены. $a_2 = 80 - 40 = 40$.

Таблица 4.2 – Решение транспортной задачи методом минимального элемента

Пункты	B_1	B_2	B_3	Запасы
A_1	8	4	5 •	30
A_2	3	7	2 40	40
A_3	1 60	4 •	6 •	60
Заявки	10	60	0	170

Продолжаем этот процесс, выбирая минимальные стоимости перевозок и заполняя эти клетки минимумом из запасов поставщиков и запросов потребителей, получим:

$$c_{2,1} = 3, x_{2,1} = 10, a_2 = 30;$$

$$c_{1,2} = 4, x_{2,1} = 30, b_2 = 30;$$

$$c_{2,2} = 7, x_{2,2} = 30, b_2 = a_2 = 0.$$

Таблица 4.3 – Решение транспортной задачи методом минимального элемента

Пункты	B_1	B_2	B_3	Запасы
A_1	8 •	4 30	5 •	0
A_2	3 10	7 30	2 40	0
A_3	1 60	4 •	6 •	0
Заявки	0	0	0	170

Число занятых клеток транспортной таблицы $m + n - 1 = 5$.

Следовательно, опорный план является невырожденным. Получили опорный план с суммарной стоимостью $f(x) = 1*60 + 2*40 + 3*10 + 4*30 + 7*30 = 500$.

Решение методом аппроксимации Фогеля (таблица 5.1-таблица 5.4)

Таблица 5.1 – Решение транспортной задачи методом аппроксимации Фогеля

Пункты	B_1	B_2	B_3	Запасы	d_i
A_1	8	4	5	30	
A_2	3	7	2	80	
A_3	1	4	6	60	
Заявки	70	60	40	170	
d_j					

Найдем штрафы по всем строкам и столбцам.

Строка 1: Первый минимальный элемент равен 4. Второй минимальный элемент равен 5. Штраф равен 1.

Строка 2: Первый минимальный элемент равен 2. Второй минимальный элемент равен 3. Штраф равен 1.

Строка 3: Первый минимальный элемент равен 1. Второй минимальный элемент равен 4. Штраф равен 3.

Столбец 1: Первый минимальный элемент равен 1. Второй минимальный элемент равен 3. Штраф равен 2.

Столбец 2: Первый минимальный элемент равен 4. Второй минимальный элемент равен 4. Штраф равен 0.

Столбец 3: Первый минимальный элемент равен 2. Второй минимальный элемент равен 5. Штраф равен 3.

В строке 3 и в столбце 3 максимальные штрафы одинаковые, выбираем клетку с минимальной стоимостью, это клетка (3, 1) и заполняем эту клетку как $\min(a_3, b_1)=60$, $b_1 = b_1 - a_3 = 10$. Вычеркиваем 3 строку.

Таблица 5.2 – Решение транспортной задачи методом аппроксимации Фогеля

Пункты	B_1	B_2	B_3	Запасы	d_i
A_1	8	4	5	30	1
A_2	3	7	2	80	1
A_3	1 60	4 •	6 •	0	3
Заявки	10	60	40	170	
d_j	2	0	3		

Снова ищем штрафы.

Строка 1: Первый минимальный элемент равен 4. Второй минимальный элемент равен 5. Штраф равен 1.

Строка 2: Первый минимальный элемент равен 2. Второй минимальный элемент равен 3. Штраф равен 1.

Столбец 1: Первый минимальный элемент равен 3. Второй минимальный элемент равен 8. Штраф равен 5.

Столбец 2: Первый минимальный элемент равен 4. Второй минимальный элемент равен 7. Штраф равен 3.

Столбец 3: Первый минимальный элемент равен 2. Второй минимальный элемент равен 5. Штраф равен 3.

Выбираем клетку с минимальной стоимостью из 1 столбца, это клетка (2, 1) и заполняем эту клетку как $\min(a_2, b_1)=10$, $a_2 = a_2 - b_1 = 70$. Вычеркиваем 1 столбец.

Таблица 5.3 – Решение транспортной задачи методом аппроксимации Фогеля

Пункты	B_1	B_2	B_3	Запасы	d_i
A_1	8 •	4	5	30	1
A_2	3 10	7	2	70	1
A_3	1 60	4 •	6 •	0	-
Заявки	0	60	40	170	
d_j	5	3	3		

Продолжаем действовать по алгоритму. Получим опорный план с суммарной стоимостью $f(x)=1*60+2*40+3*10+4*30+7*30=500$ и с транспортной таблицей 5.4.

Таблица 5.4 – Решение транспортной задачи методом аппроксимации Фогеля

Пункты	B_1	B_2	B_3	Запасы
A_1	8 •	4 30	5 •	0
A_2	3 10	7 30	2 40	0
A_3	1 60	4 •	6 •	0
Заявки	0	0	0	170

Теперь проверим оптимальность полученного опорного плана методом потенциалов для транспортной таблицы, полученной методом минимального элемента.

Для базисных клеток найдем u_i, v_j ($i = \overline{1,3}, j = \overline{1,3}$). Положим $u_1 = 0$.

Тогда из условия $u_1 + v_2 = 4$, получим $v_2 = 4$ и т.д. Получим систему потенциалов и запишем их в таблицу (таблица 6.1).

Таблица 6.1 – Решение транспортной задачи методом потенциалов.

Пункты	B_1	B_2	B_3	Запасы	U
A_1	8 •	4 30	5 •	30	0
A_2	3 10	7 30	2 40	80	3
A_3	1 60	4 •	6 •	60	1
Заявки	70	60	40	170	
V	0	4	-1		

Для каждой свободной клетки определим относительные оценки:

$\Delta_{1,1} = 8$, $\Delta_{1,3} = 6$, $\Delta_{3,2} = -1$, $\Delta_{3,3} = 6$. $\Delta_{3,2} < 0$, значит наш план не является оптимальным и его можно улучшить. Построим означенный цикл из клетки с относительной оценкой $\Delta_{3,2}$. Запишем в эту клетку знак «+», также в цикл попадут клетки A_3B_1 , A_2B_1 и A_2B_2 (таблица 6.2).

Таблица 6.2 – Решение транспортной задачи методом потенциалов.

Пункты	B_1	B_2	B_3	Запасы	U
A_1	8 •	4 30	5 •	30	0
A_2	3 + 10	7 - 30	2 40	80	3
A_3	1 - 60	4 + •	6 •	60	1
Заявки	70	60	40	170	
V	0	4	-1		

Из перевозок $x_{i,j}$ стоящих в клетках со знаком минус, выбираем наименьшую $\theta = \min \{60, 30\} = 30$. Прибавляем θ к перевозкам, стоящим в

положительных клетках, и вычитаем θ из перевозок, стоящих в отрицательных клетках. В результате получим новый опорный план (таблица 6.3).

Таблица 6.3 – Решение транспортной задачи методом потенциалов.

Пункты	B_1	B_2	B_3	Запасы
A_1	8 •	4 30	5 •	30
A_2	3 40	7 •	2 40	80
A_3	1 30	4 30	6 •	60
Заявки	70	60	40	170

Проверим оптимальность опорного плана, для этого заново найдем потенциалы и проверим условие оптимальности плана. Положим $u_1 = 0$.

Тогда получим:

$$u_1 + v_2 = 4, v_2 = 4;$$

$$u_3 + v_2 = 4, u_3 = 0;$$

$$u_3 + v_1 = 1, v_1 = 1;$$

$$u_2 + v_1 = 3, u_2 = 2;$$

$$u_2 + v_3 = 2, v_3 = 0.$$

Запишем их в новую транспортную таблицу (таблица 6.4).

Таблица 6.4 – Решение транспортной задачи методом потенциалов.

Пункты	B_1	B_2	B_3	Запасы	U
A_1	8 •	4 30	5 •	30	0
A_2	3 40	7 •	2 40	80	2
A_3	1 30	4 30	6 •	60	0
Заявки	70	60	40	170	
V	1	4	0		

Для каждой свободной клетки определим относительные оценки:

$$\Delta_{1,1} = 7, \quad \Delta_{1,3} = 5, \quad \Delta_{2,2} = 1, \quad \Delta_{3,3} = 6. \quad \Delta_{i,j} \geq 0 \forall i = \overline{1, m}, j = \overline{1, n},$$

следовательно, по теореме 3.2. наш план является оптимальным и целевая функция (минимальные затраты на перевозки) равна $f(x) = 1 \cdot 30 + 2 \cdot 40 + 3 \cdot 40 + 4 \cdot 30 + 4 \cdot 30 = 470$.

4 Языки программирования и среда разработки

4.1 Языки программирования

Язык программирования – это формальный язык, предназначенный для написания программ на компьютере. Языки программирования можно разделить на языки программирования низкого уровня и языки программирования высокого уровня.

Язык программирования низкого уровня создан для определенного типа ЭВМ и представляется в виде синтаксических систем, которые обращаются непосредственно к «железу», и отражают его внутренний машинный код [4].

Язык программирования высокого уровня это язык, который описывает задачи в ясном для программиста виде, использующий абстрактные понятия и смысловые конструкции, которые кратко описывают структуры данных и операции над ними. Высокоуровневые языки нацелены на эффективность и высокую скорость разработки. Программы, написанные на высокоуровневом языке, требуют перевода в машинный код, это происходит с помощью инструментальных программ – трансляторов (компиляторов и интерпретаторов) [4].

Примеры высокоуровневых языков: C, Objective-C, C++, C#, Python, Java, JavaScript, PHP, Ruby, Perl, Go, Swift, Pascal, Delphi и т. д.

В качестве языка программирования для создания приложения, позволяющего реализовать решение транспортной задачи, был взят язык C++.

C++ – компилируемый, статически типизированный язык программирования. Одним из аспектов разработки было сохранение совместимости с языком C. Не считая некоторых деталей, C++ является надстройкой языка C. Язык C взят в качестве базового языка для C++, так как он является многоцелевым, отвечает требованиям большинства задач системного программирования, идет везде и на всем. C++ дополняет эти

возможности определением новых типов данных, которые носят название классы (типы, определяемые пользователем). С++ поддерживает разные парадигмы программирования, такие как процедурное программирование, объектно – ориентированное программирование и т. д. [2].

Область применения этого языка достаточно широкая, она включает в себя создание операционных систем, программного обеспечения, прикладных программ, серверов, развлекательных приложений и игр [2].

Плюсы языка С++:

- Совместимость с Си, позволяющая использовать весь существующий С-код;
- Высокая производительность, в смысле скорости работы программ, а не их написания;
- Кроссплатформенность. Минимальные требования на ЭВМ для запуска скомпилированных программ. Независимость от архитектуры компьютера;
- Многопоточность. Тем самым более высокая производительность программ;
- Поддержка разных парадигм программирования;
- Универсальность [16].

Минусы языка С++:

- Не подойдет для работы с Web;
- громоздкий синтаксис;
- Не подходит для создания корпоративных приложений (для их создания предпочитают Java или С#) [16].

4.2 Среда разработки

Среда разработки или IDE (Integrated Development Environment) – это программа в которой разработчик реализует различного рода программы, может посмотреть на ошибки в коде и результат работы программы. Среда разработки, чаще всего, содержит текстовый редактор, компилятор или

интерпретатор, отладчик и средство автоматической сборки. Часто IDE являются универсальными, в таких средах можно работать с различными языками программирования, но бывают среды, направленные только на один язык программирования.

Также как и языков программирования, сред разработки достаточно много, перечислим некоторые из них: CLion, Microsoft Visual Studio, PyCharm, IntelliJ IDEA, Eclipse, Xcode, Code::Blocks, Visual Studio Code. У каждой среды свои преимущества и недостатки, поддерживаются на различных платформах, например, Windows, Linux, macOS. В качестве среды разработки выбрана среда от компании Microsoft, которая называется Microsoft Visual Studio 2017.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии Intelli Sense и возможностью простейшего перепроектирования кода, а также различные инструменты для разработки, которые позволяют создавать web-сайты, web-приложения, базы данных, консольные и графические приложения с поддержкой Windows Forms и WPF. Visual Studio поддерживает такие языки, как JavaScript, Visual Basic, Visual C#, Visual C++, Python, DHTML, ASP.NET, Visual F#, XAML и другие, на платформах Windows и macOS, для Linux есть только редактор кода [17].

Особенности Visual Studio:

- Технология редактирования исходного кода Intelli Sense.
- Возможность индивидуализации рабочей панели.
- Поддержка разделенного экрана.

Из недостатков Visual Studio можно выделить ее тяжеловесность. Для написания легкой программы, решающую простую задачу, удобнее воспользоваться другой средой, с легким редактором [17].

5 Логистические информационные системы (ЛИС)

Логистика – это планирование и управление экономическими и технологическими процессами, контроль и регулировании различного рода потоков. Целью логистики является оптимизация экономического процесса, начиная с поставщика и заканчивая потребителем [3].

Транспортная логистика является отраслью логистической науки, она представляет собой организацию грузопотоков, которая занимается планированием и управления перевозками грузов из начальной точки в конечную по оптимальному маршруту. Транспортная логистика предоставляет возможность эффективного управления транспортными потоками, а также она минимизирует финансовые и временные затраты. Основная цель транспортной логистики – это транспортировка некоторого товара из начальной точки в конечную точно в срок, с наилучшим соотношением цены и качества. К задачам транспортной логистики относятся:

- планирование транспортного процесса;
- Выбор правильного транспортного средства;
- Выбор перевозчика и логистических партнёров;
- Построение наилучшего маршрута доставки;
- Контроль товара во время его перевозки [15].

Логистическая информационная система – это структура, состоящая из различного программного обеспечения и персонала, позволяющая управлять, организовывать, автоматизировать и оптимизировать задачи логистики, связанные, в основном, со складскими и транспортными ресурсами. Вообще говоря, компании делят логистическую структуру на отдельные взаимосвязанные области. Перечислим наиболее важные из этих областей:

- Логистика в области поставок;
- Производственная логистика;
- Сбытовая логистика;
- Транспортная логистика;

— Управление логистической цепочкой [7].

Для каждой из этих областей существуют собственные подходы в управлении, реализованные с помощью различных программных обеспечений. К методам управления относятся:

- S&OP (Sales & Operation Planning) — Система планирования продаж и операционной деятельности;
- FP&S (Factory planning & Scheduling) — Система планирования технологических процессов и создания календарных графиков;
- SRM (Supplier Relationship Management) — Система управления взаимоотношениями с Поставщиками;
- CRM (Customer Relationship Management) — Система управления взаимоотношениями с Заказчиками;
- TMS (Transportation Management System) — Система управления транспортом;
- WMS (Warehouse Management System) — Система управления складом [7].

5.1 Системы для управления перевозками

Система управления перевозками (TMS) – это программное обеспечение для автоматизации задач транспортной логистики, позволяющее получить оптимальное решение всего процесса транспортировки грузов и товаров с учетом всех нюансов, таких как стоимость транспортировки, выбор транспорта для перевозки и количество транспорта, сроки перевозки и т. д. Также TMS позволяет планировать загрузку и работу транспорта и осуществлять контроль за товаром [5].

TMS производит расчет стоимости транспортировки различными видами транспорта (наземный, воздушный или морской), содержит данные о таможенных затратах, информацию о погрузочно-разгрузочных работах. Одна из задач системы — по запросу суметь предоставить информацию о нахождении груза, когда должен быть доставлен.

К достоинствам системы можно отнести анализ различных вариантов транспортировки и принятие экономически обоснованного оптимального решения. Также при моделировании можно найти возможные риски, способные прослеживаться при моделировании транспортной сети. Такие моменты можно рассчитать до начала грузоперевозок [9].

Большинство TMS схожи между собой, но некоторые компании предлагают особые функции, например, расчет надбавки и улучшение плана транспортировки по маршруту с ограничениями по экологическому налогу. Рассмотрим возможности некоторых систем управления [12].

Система Qguar TMS представляет собой системный модуль, с помощью которого производится планирование, отслеживание и подсчитывается стоимость транспортировки груза в различных схемах сбыта. Данная система используется во многих странах Европы для улучшения транспортной логистики. Построение наилучших маршрутов доставки и другие интересные функции модуля Qguar TMS делают эту систему мощным аппаратом в управлении транспортной логистики, позволяющим автоматизировать транспортно - складские комплексы [12].

Ясно, что транспортные задачи и их решения связаны со складским управлением, иначе непрерывное проектирование плана перевозок невозможно достичь при отсутствии информации о складских запасах на данный момент. Автоматизация логистики — это такие тесные отношения между системами управления транспортом и системами управления складом, которая предоставляет возможность компаниям эффективно управлять и контролировать работу транспорта и складскими запасами онлайн, при изменяющихся требованиях сторон [12].

Система Qguar TMS предоставляет следующие возможности:

- Управление заказами при транспортировке. Запись клиентов, прием заявок на транспортировку груза, поиск и изменение заказов, запись переговоров.
- Построение маршрутов транспортировки груза.

- Оценка стоимости на предлагаемые транспортные услуги. Тарифы, налоги, подготовка счетов.
- Статистика и анализ информации по транспортной логистике.

К основным функциям системы можно отнести: прием и отслеживание заявок на транспортировку груза, выбор транспортного средства, мануальное и автоматическое составление плана перевозки, подсчет стоимости на транспортировку груза по выбранному плану, расчет предлагаемых услуг, анализ, получившихся в ходе транспортировки, затрат [12].

Система «РУССЛАНД-ЭКСПЕДИТОР» наравне с системой Qguar TMS имеет множество возможностей и функций по управлению и контролю за перевозками, которая автоматизирует работу компании ОАО «Шереметьево-Карго». Данная система разработана для создания единой информационной среды и всех организаций, участвующих и взаимодействующих в перевозках внутри страны, а также за границей [12].

Система «РУССЛАНД-ЭКСПЕДИТОР» решает следующие задачи:

- организация различного вида перевозок;
- организация коммуникации лиц, несущих ответственность за транспорт;
- поиск заказчиков, подрядчиков и всевозможных участников перевозки;
- расчет тарифов услуг;
- организация типовых схем по контролю за перевозками;
- оповещение клиентов о том, в каком состоянии прибывает груз, а также наблюдение за процессом доставки.

Пользуясь системой «РУССЛАНД-ЭКСПЕДИТОР», компания ОАО «Шереметьево – Карго» может получать, хранить, пользоваться и анализировать информацию о спросе и потребностях на транспортные услуги. Также, благодаря системе, участники перевозок отлично взаимодействуют, а компании и их агенты могут обмениваться информацией, используя такую информационную среду. За счет этой системы, стало возможным организовывать транспортировку и логистические модели,

уменьшив различного рода издержки, например, расходы на хранение и транспортировку [12].

К преимуществам такой системы можно отнести: способность организовать единую базу услуг; возможность взаимодействия участников процесса перевозок; открытый доступ к нормативно-справочным данным; возможность удаленно управлять производственной деятельностью[12].

6 Программная реализация решения транспортной задачи

Программа реализована в среде Microsoft Visual Studio 2017 на языке программирования C++, так как данная среда и язык являются многоцелевыми и отвечают всем необходимым требованиям для создания приложения. Для создания пользовательского интерфейса приложения (API) использовались различные возможности Windows Forms. Приложение состоит из одной основной формы, которая представлена на рисунке 2.1 и рисунке 2.2.

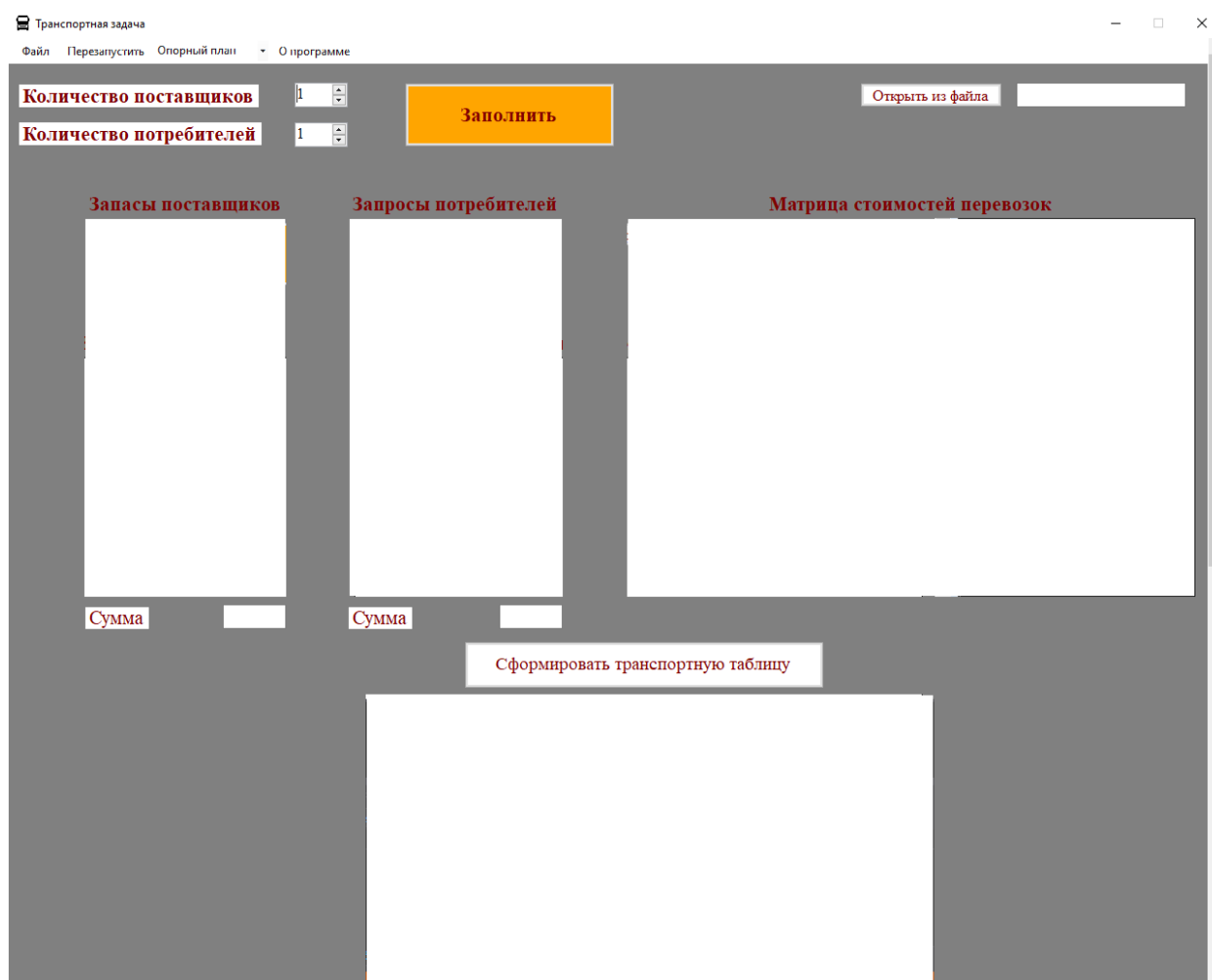


Рисунок 2.1 – Графический интерфейс приложения

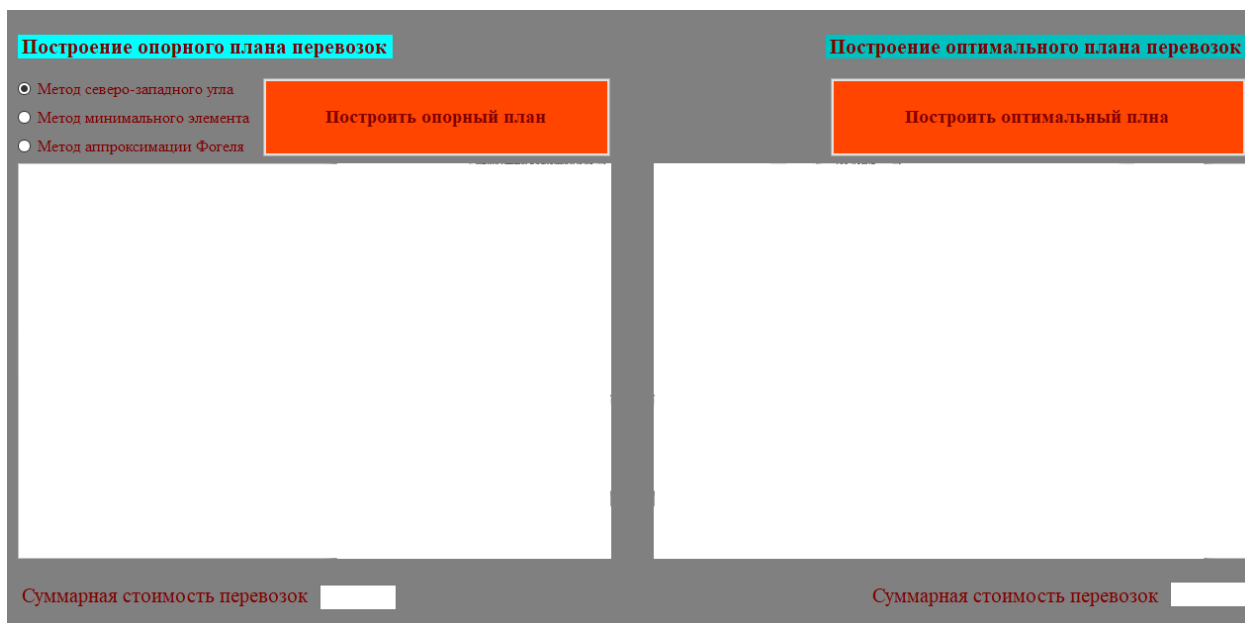


Рисунок 2.2 – Графический интерфейс приложения

Для создания интерфейса использовались различные элементы управления. Кнопки Button позволяют пользователю при щелчке ввести начальные данные, как вручную, так и открыть из файла, отобразить транспортную таблицу, построить опорный и оптимальный планы перевозок, которые отображаются в элементах DataGridView. MenuShip содержит в себе элементы «Файл», «Перезапустить», «Опорный план», «О программе». В меню в элементе «Файл» можно открыть из файла начальные данные, сохранить в файл, получившийся оптимальный план и его стоимость, и выйти из приложения, а в элементе «Опорный план» выбрать метод, по которому будет строиться опорный план. Также используются элементы TextBox для отображения каких либо данных, RadioButton для выбора метода поиска опорного плана, которые связаны с элементом «Опорный план» из меню, элементы Label с текстом и DomainUpDown для удобного выбора числа поставщиков и потребителей.

Также программа защищена от того, что пользователь вводит неправильные данные, например, слова вместо чисел. Проверка на число может быть реализована, например, так, как показано на рисунке 3.

```

schB = new float[n];
for (int i = 0; i < n; i++) schB[i] = 0;
for (int ii = 0; ii < n; ii++) {
    bool error = false;
    int k = 0;
    std::string nums;
    ConvertToString(MasPotr->Rows[ii]->Cells[0]->Value, nums);

    const char *s = nums.c_str();

    for (int i = 0; i < strlen(s); i++) {
        if (nums[i] == '.') k++;
    }

    for (int i = 0; i < strlen(s); i++) {
        if (!isdigit(s[i]) || (k > 1) || (s[0] == '.') || (s[strlen(s) - 1] == '.')) {
            if ((s[i] != '.') || (k > 1) || (s[0] == '.') || (s[strlen(s) - 1] == '.')) {
                error = true;
                schB[ii] = 1;
            }
        }
    }
}
if (error) {
    MessageBox::Show("Ошибка ввода данных. Введите положительное число в B" + Convert::ToString(ii + 1), "Внимание!");
}
}

```

Рисунок 3 – Пример функции проверки на число

Допустим в ячейки массива или матрицы введены некоторые значения строкового типа. Значение строкового типа преобразуется в символьный тип и далее посимвольно проверяется, является ли данный символ цифрой или нет, если условие истина и символ не является цифрой, то переменная логического типа принимает значение истины. По окончании цикла проверки ячейки, если логическая переменная имеет значение истины, то пользователь предупреждается, для этого выводится на экран окно с предупреждением, например, как на рисунке 4. Данная проверка осуществляется по всем ячейкам массива или матрицы.

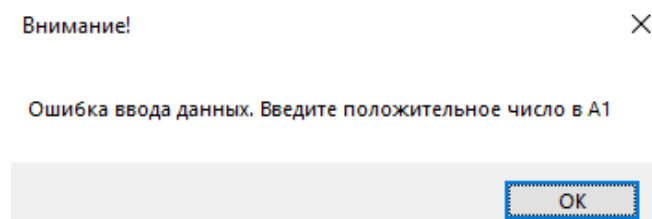


Рисунок 4 – Предупреждение пользователя об ошибке ввода

Если задача является открытой, то пользователя также ждет предупреждение, чтобы ввел данные заново, так как рассматриваются только задачи закрытого типа. Если пользователь нажимает кнопки «Построить опорный план» или «Построить оптимальный план», при этом не ввел начальные данные,

выводится на экран окно с предупреждением о том, что нужно ввести запасы, запросы и стоимости перевозок.

При открытии приложения есть возможность выбрать способ введения начальных данных, если вводим вручную, то выбираем количество поставщиков и потребителей и нажимаем кнопку «Заполнить», после этого сформируются таблицы нужной размерности, в которые вносятся запасы, запросы и стоимости перевозок. Если данные считываются с файла, то нажимаем кнопку «Открыть из файла», после чего откроется окно, в котором можно найти нужный файл, он откроется, если в этом файле данные введены корректно. Далее мы можем сформировать транспортную таблицу для наглядности, либо сразу построить опорный план перевозок методом северо-западного угла, методом минимального элемента или методом аппроксимации Фогеля и найти суммарную стоимость перевозок. Для выбора метода нужно нажать на тот метод, которым пользователь хочет найти опорный план, либо выбрать из списка в меню один из этих методов. Реализация, например, метода северо-западного угла и метода минимального элемента может выглядеть так, как показано на рисунке 5 и рисунке 6:

```
while (i <= M and j <= N) {  
    if (A[i] < B[j]) {  
        X[i][j] = A[i];  
        B[j] = B[j] - A[i];  
        i++;  
        for (int k = 1; k <= N; k++)  
            if (k != j) {  
                X[i][k] = -1;  
            }  
    }  
    else {  
        X[i][j] = B[j];  
        A[i] = A[i] - B[j];  
        j++;  
        for (int k = 1; k <= M; k++)  
            if (k != i)  
                X[k][j] = -1;  
    }  
}
```

Рисунок 5 – Метод северо-западного угла

```

while (ii <= M and jj <= N) {
    int minc = maxc;
    for (int k = 1; k <= M; k++) {
        for (int g = 1; g <= N; g++)
            if (minc > c1[k][g]) {
                minc = c1[k][g];
                i1 = k;
                j1 = g;
            }
    }
    if (A[i1] < B[j1]) {
        X[i1][j1] = A[i1];
        B[j1] = B[j1] - A[i1];
        ii++;
        for (int k = 1; k <= N; k++)
            c1[i1][k] = maxc + 1;
    }
    else {
        X[i1][j1] = B[j1];
        A[i1] = A[i1] - B[j1];
        jj++;
        for (int k = 1; k <= M; k++)
            c1[k][j1] = maxc + 1;
    }
}
}

```

Рисунок 6 – Метод минимального элемента

Оптимальный план получим методом потенциалов, для этого пользователю нужно нажать кнопку «Построить оптимальный план», после чего вызывается функция, в которой реализован метод потенциалов. Функция построена на основе алгоритма метода потенциалов. В ней ищутся потенциалы для базисных клеток, как показано на рисунке 7.

```

while (tmp <= m + n) {
    for (int i = 1; i <= m; i++)
        if (ub[i] == true)
            for (int j = 1; j <= n; j++)
                if (x[i][j] != -1)
                    if (vb[j] == false) {
                        v[j] = c[i][j] - u[i];
                        vb[j] = true;
                    }
            for (int j = 1; j <= n; j++)
                if (vb[j] == true)
                    for (int i = 1; i <= m; i++)
                        if (x[i][j] != -1)
                            if (ub[i] == false){
                                u[i] = c[i][j] - v[j];
                                ub[i] = true;
                            }
        tmp++;
};

```

Рисунок 7 – Поиск потенциалов

Далее находим относительные оценки для свободных клеток и проверяем условие оптимальности (рисунок 8).

```

bool flag = true;
for (int i = 1; i <= m; i++)
    for (int j = 1; j <= n; j++)
        if (x[i][j] == -1) {
            if (delta[i][j] < 0)
                flag = false;
        }

```

Рисунок 8 – Проверка условия оптимальности

Если условие не выполняется, то ищем цикл для минимальной относительной оценки и перераспределяем поставки (рисунок 9).

```

find1 = find_gor(k, l, k, l, m, n, x, v, 0, -1); //поиск цикла, возвращает минимальный элемент цикла, стоящий в клетке со знаком "-"
for (int i = 1; i <= m; i++)
    for (int j = 1; j <= n; j++)
        if (v[i][j] != -1) {
            x[i][j] = x[i][j] + v[i][j]; //меняем опорный планперевозок
            if ((i == k) && (j == l)) x[i][j] = x[i][j] + 1; //добавляем 1 (т.к. до этого было -1 )
            if ((v[i][j] <= 0) && (x[i][j] == 0)) x[i][j] = -1; //если ячейка обнулилась, то выбрасываем её
        }
x[k][l] = v[k][l];

```

Рисунок 9 – Перераспределение по циклу

Далее возвращаемся к поиску потенциалов.

По известному нам решению, которое мы получили вручную, протестируем программу, для этого используем начальные данные из примера решения транспортной задачи, который приведен выше и получим

решение с помощью разработанного приложения. Данные считаны с файла и выведены в таблицы с начальными данными, это можно увидеть на рисунке 10.

Запасы поставщиков

Поставщик	Запасы
▶ A1	30
A2	80
A3	60

Сумма

1/0

Запросы потребителей

Потребитель	Запросы
▶ B1	70
B2	60
B3	40

Сумма

1/0

Матрица стоимостей перевозок

Стоимость	B1	B2	B3
▶ A1	8	4	5
A2	3	7	2
A3	1	4	6

Построение опорного плана перевозок

☐ Метод северо-западного угла
☒ Метод минимального элемента
☐ Метод аппроксимации Фогеля

Построить опорный план

Пункты	B1	B2	B3	Запасы
▶ A1	30	30		30
A2	10	30	40	80
A3	60			60
Запросы	70	60	40	170/170

Суммарная стоимость перевозок
500

Построение оптимального плана перевозок

Построить оптимальный план

Пункты	B1	B2	B3	Запасы
▶ A1	30			30
A2	40		40	80
A3	30	30		60
Запросы	70	60	40	170/170

Суммарная стоимость перевозок
470

Рисунок 12 – Результат использования программы

Сохраним результат в файл. Для этого в меню нажимаем «Файл», далее «Сохранить как...», после чего откроется окно, выбираем место хранения файла и его имя. Нажимаем кнопку сохранить, после чего файл успешно сохранится. В файле будет отображаться оптимальный план перевозок и его суммарная стоимость (рисунок 13).

 Результат – Блокнот

Файл Правка Формат Вид Справка

Оптимальный план перевозок:

-	30	-
40	-	40
30	30	-

суммарная стоимость перевозок: 470

Рисунок 13 – Файл с результатом использования программы

По результатам тестирования получили оптимальный план перевозок, с суммарной стоимостью $f(x)=470$. При проведении тестирования приложения ошибки не были найдены. Если сравнивать результат использования программы с известным нам решением, полученным вручную, то мы увидим, что решения нашей задачи полностью совпадают.

Заключение

При выполнении выпускной квалификационной работы была изучена математическая модель транспортной задачи и рассмотрены основные методы ее решения. Построен опорный план перевозок и получен оптимальный план методом потенциалов.

Транспортная задача является задачей минимизации стоимости перевозок некоторого груза. Задачу можно решить, как вручную, так и с использованием специальных программ, но при решении вручную, решение может оказаться трудоемким.

В практической части работы разобран пример решения транспортной задачи, а также было реализовано приложение, написанное в интегрированной среде разработки Visual Studio 2017 на языке программирования высокого уровня C++, позволяющее находить опорный план одним из трех рассмотренных методов и оптимальный план перевозок методом потенциалов. Произведено тестирование приложения на задаче с известным решением, которое показало полное совпадение результатов решения.

Данное приложение можно использовать как модуль TMS или же использовать его в учебном процессе для расчета транспортных задач.

Список использованных источников и литературы

1. Болотникова О. В. Линейное программирование: транспортные и сетевые модели: учеб. пособие / О. В. Болотникова, Д. В. Тарасов, Р. В. Тарасов. – Пенза: ПГУ, 2016. – 88 с.
2. Бьярн Страустрап. Введение в язык C++ [Электронный ресурс]. – Режим доступа: <http://citforum.ru/programming/cpp/apredis.shtml> (дата обращения: 08.02.2021).
3. Введение в транспортную логику: учебное пособие / А. В. Кириченко, А. Л. Кузнецов, О. А. Ражев, В. А. Фетисов. СПб.: ГУАП, 2011. – 228 с.
4. Высокие и низкие языки программирования [Электронный ресурс]: Справочник студенческий / Справочник24, 2012-2021. – URL: https://spravochnik.ru/programirovanie/vysokie_i_nizkie_yazyki_programirovaniya/ (дата обращения: 08.02.2021).
5. Левина Т. В. Системы управления транспортировкой [Электронный ресурс]: Логистика и управление цепями поставок / Copyright, 2016. – Режим доступа: <http://www.lscm.ru/index.php/ru/po-godam/item/790> (дата обращения: 09.02.2021).
6. Леонова Н. Л. Задачи линейного программирования и методы их решения: учебно-методическое пособие / Н. Л. Леонова. – СПб. : ВШТЭ СПбГУПТД, 2017. – 75 с.
7. Логистическая информационная система (классификация и методы) [Электронный ресурс]: Государство. Бизнес. ИТ, 2005-2021. – Режим доступа: [http://www.tadviser.ru/index.php/Статьи:Логистическая_информационная_система_\(классификации_и_методы\)](http://www.tadviser.ru/index.php/Статьи:Логистическая_информационная_система_(классификации_и_методы)) (дата обращения: 09.02.2021).
8. Метод Фогеля [Электронный ресурс]: Онлайн калькуляторы по математике и статистике / Новый семестр. – М., 2006-2018. – URL: <https://math.semestr.ru/transp/fogel.php> (дата обращения: 21.12.2020).

9. Оптимизационные логистические решения управления транспортировкой в цепях поставок. Методы и модели оптимальной маршрутизации [Электронный ресурс]: Студопедия – лекционный материал для студентов / Студопедия, 2013-2021. – Режим доступа: http://studopedia.su/7_2071_optimizatsionnie-logisticheskie-resheniya-upravleniya-transportirovkoj-v-tsepyah-postavok-metodi-i-modeli-optimalnoy-marshrutizatsii.html (дата обращения: 09.02.2021).
10. Орлов А.И. Теория принятия решений. Учебное пособие / А.И. Орлов. — М.: Издательство «Экзамен», 2005. — 656 с.
11. Плетнева Л. А. Задачи линейного программирования по курсу «Прикладная математика»: учеб. пособие / Л. А. Плетнева, И. Г. Каграманова, М. А. Леева. – М.: МАДИ, 2015. – 120 с.
12. Просянкин С. М., Красникова Д. А. Актуальность применения TMS-систем для управления современными транспортными компаниями // Научно-методический электронный журнал «Концепт». – 2015. – Т. 35. – С. 131–135. – URL: <http://e-koncept.ru/2015/95580.htm>.
13. Решение транспортной задачи линейного программирования [Электронный ресурс]: Онлайн калькуляторы по математике и статистике / Новый семестр. – М., 2006-2018. – URL: <https://math.semestr.ru/transp/tzlp.php> (дата обращения: 20.04.2020)
14. Рыхлов В. С. Прикладные методы оптимизации уравнениям: учебно-методическое пособие для студентов / В. С. Рыхлов, В.В. Корневч, В.П. Курдюмов. – Издание второе, исправленное. – Саратов: СГУ им. Н. Г. Чернышевского, 2012. – 172 с.
15. Транспортная логистика [Электронный ресурс]: Решения для логистики. – Режим доступа: https://www.axelot.ru/knowhow/press/detail_48008/ (дата обращения: 09.02.2021).
16. Тюгашев А. А. Основы программирования. Часть I. – СПб: Университет ИТМО, 2016. – 160 с.

17. Эсеналиева Г.А., Сейтказиева Н.С. VisualBasic для выполнения лабораторных работ Visual Studio.Net 2010: Учебно-методическое пособие. – Бишкек: Колледж Кыргызского государственного университета им. И. Арабаева, 2018. – 46 с.

Приложение А

программная реализация методов решения транспортной задачи

```
//метод северо-западного угла
void nwsm(int M, int N, float A[], float B[], float *C[], float *X[]) {
    int i = 1, j = 1;
    while (i <= M && j <= N) {
        if (A[i] < B[j]) {
            X[i][j] = A[i];
            B[j] = B[j] - A[i];
            i++;
            for (int k = 1; k <= N; k++)
                if (k != j) {
                    X[i][k] = -1;
                }
        }
        else {
            X[i][j] = B[j];
            A[i] = A[i] - B[j];
            j++;
            for (int k = 1; k <= M; k++)
                if (k != i)
                    X[k][j] = -1;
        }
    }
    //клетки со значениями "-1" - это пустые клетки
}
```

Рисунок А.1 – Метод северо-западного угла.

```
//метод минимального элемента
void mme(int M, int N, float A[], float B[], float *C[], float *X[]) {
    int ii = 1, i1 = 1, j1 = 1;
    float **c1;
    c1 = new float *[M]; //вспомогательная матрица
    for (int i = 1; i <= M; i++) c1[i] = new float[N];
    for (int i = 1; i <= M; i++) {
        for (int j = 1; j <= N; j++)
            c1[i][j] = C[i][j];
    }
    float maxc = C[1][1];
    for (int k = 1; k <= M; k++) {
        for (int g = 1; g <= N; g++)
            if (maxc < C[k][g])
                maxc = C[k][g]; //максимальная стоимость перевозки
    }
    while (ii <= M+N-2) {
        int minc = maxc;
        for (int k = 1; k <= M; k++) {
            for (int g = 1; g <= N; g++)
                if (minc > c1[k][g]) {
                    minc = c1[k][g];
                    i1 = k;
                    j1 = g;
                }
        }
    }
}
```

Рисунок А.2.1 – Метод минимального элемента.

```

if (A[i1] < B[j1]) {
    X[i1][j1] = A[i1];
    B[j1] = B[j1] - A[i1];
    A[i1] = 0;
    ii++;
    for (int k = 1; k <= N; k++)
        c1[i1][k] = maxc + 1;
}
else {
    X[i1][j1] = B[j1];
    A[i1] = A[i1] - B[j1];
    B[j1] = 0;
    ii++;
    for (int k = 1; k <= M; k++)
        c1[k][j1] = maxc + 1;
}
}
// если осталась незаполненная клетка
for (int i = 1; i <= M; i++) {
    for (int j = 1; j <= N; j++)
        if ((A[i] != 0) && (B[j] != 0)) {
            X[i][j] = A[i];
            A[i] = A[i] - B[i];
        }
}
}

```

Рисунок А.2.2 – Метод минимального элемента.

```

//метод аппроксимации Фогеля
void maf(int M, int N, float A[], float B[], float *C[], float *X[]) {
    int k1, l1, k2, l2, count = 1;
    float c1, c2, *d = new float[M + N], **newC = new float *[M];
    for (int i = 1; i <= M; i++) newC[i] = new float[N];
    for (int i = 1; i <= M; i++) {
        for (int j = 1; j <= N; j++)
            newC[i][j] = C[i][j];
    }; //вспомогательная матрица стоимостей
    while (count < M + N - 1) {
        float maxc = newC[1][1];
        for (int i = 1; i <= M; i++)
            for (int j = 1; j <= N; j++)
                if (newC[i][j] != -1) {
                    if (maxc <= newC[i][j]) {
                        maxc = newC[i][j]; //максимальная стоимость перевозки
                        k2 = i; l2 = j;
                    }
                }
    }
}

```

Рисунок А.3.1 – Метод аппроксимации Фогеля.

```

//штрафы по строкам
for (int i = 1; i <= M; i++) {
    c1 = c2 = maxc;
    for (int j = 1; j <= N; j++)
        if (newC[i][j] != -1) {
            if (newC[i][j] < c1) {
                c1 = newC[i][j];
                l1 = j;
            }
        }
    for (int j = 1; j <= N; j++)
        if (newC[i][j] != -1) {
            if (j != l1) {
                if (newC[i][j] < c2) {
                    c2 = newC[i][j];
                }
            }
        }
    d[i] = abs(c1 - c2);
}

//штрафы по столбцам
for (int j = 1; j <= N; j++) {
    c1 = c2 = maxc + 1;
    for (int i = 1; i <= M; i++)
        if (newC[i][j] != -1) {
            if (newC[i][j] <= c1) {
                c1 = newC[i][j];
                l1 = i;
            }
        }
    for (int i = 1; i <= M; i++)
        if (newC[i][j] != -1) {
            if (i != l1) {
                if (newC[i][j] <= c2) {
                    c2 = newC[i][j];
                }
            }
        }
    d[M + j] = abs(c1 - c2);
}

```

Рисунок А.3.2 – Метод аппроксимации Фогеля.

```

//первый максимальный штраф
float maxd = -1;
for (int i = 1; i <= M + N; i++)
    if (maxd < d[i]) {
        maxd = d[i];
        k1 = i;
    }

//количество максимальных штрафов
int k = 1;
for (int i = 1; i <= M + N; i++)
    if ((maxd == d[i]) && (i != k1)) {
        k++;
    }

//если максимальных штрафов несколько
if (k > 1) {
    int *arrk = new int[k];
    arrk[1] = k1;
    int l = 2;
    for (int i = 1; i <= M + N; i++)
        if (((maxd == d[i]) && (i != k1))) {
            arrk[l] = i;
            l++;
        }
}

```

Рисунок А.3.3 – Метод аппроксимации Фогеля.

```

//массив хранящий минимальные элементы по строкам и столбцам и с их координатами
float **mc = new float *[3];
for (int i = 1; i <= 3; i++) mc[i] = new float[2 * M*N];
int q = 1;
//по строкам
for (int i = 1; i <= M; i++)
    for (int g = 1; g <= k; g++)
        if (i == arrk[g]) {
            c1 = maxc;
            for (int j = 1; j <= N; j++)
                if (newC[i][j] != -1) {
                    if (newC[i][j] < c1) {
                        c1 = newC[i][j];
                        k2 = i; l2 = j;
                    }
                }
            mc[1][q] = c1;
            mc[2][q] = k2;
            mc[3][q] = l2;
            q++;
            for (int j = 1; j <= N; j++)
                if (newC[i][j] != -1) {
                    if (((mc[1][q - 1] == newC[i][j]) && (j != mc[3][q - 1]))) {
                        mc[1][q] = newC[i][j];
                        mc[2][q] = i;
                        mc[3][q] = j;
                        q++;
                    }
                }
        }
}

```

Рисунок А.3.4 – Метод аппроксимации Фогеля.

```

//по столбцам
for (int j = 1; j <= N; j++)
    for (int g = q; g <= k; g++)
        if (j == arrk[g] - M) {
            c1 = maxc;
            for (int i = 1; i <= M; i++)
                if (newC[i][j] != -1) {
                    if (newC[i][j] < c1) {
                        c1 = newC[i][j];
                        k2 = i; l2 = j;
                    }
                }
            mc[1][q] = c1;
            mc[2][q] = k2;
            mc[3][q] = l2;
            q++;
            for (int i = 1; i <= M; i++)
                if (newC[i][j] != -1) {
                    if (((mc[1][q - 1] == newC[i][j]) && (i != mc[2][q - 1]))) {
                        mc[1][q] = newC[i][j];
                        mc[2][q] = i;
                        mc[3][q] = j;
                        q++;
                    }
                }
        }
}
q--;

```

Рисунок А.3.5 – Метод аппроксимации Фогеля.

```

//минимальные элементы(могут повторяться)
float **mmc = new float *[3];
for (int i = 1; i <= 3; i++) mmc[i] = new float[q];
float minmmc = mc[1][1];
k2 = mc[2][1]; l2 = mc[3][1];
int t = 1, w = 1;
for (int i = 2; i <= q; i++)
    if (minmmc > mc[1][i]) {
        minmmc = mc[1][i];
        w = i;
        k2 = mc[2][i]; l2 = mc[3][i];
    }
mmc[1][t] = minmmc;
mmc[2][t] = k2;
mmc[3][t] = l2;
t++;
for (int i = 1; i <= q; i++)
    if (((mmc[1][t - 1] == mc[1][i]) && (i != w))) {
        mmc[1][t] = mc[1][i];
        mmc[2][t] = mc[2][i];
        mmc[3][t] = mc[3][i];
        t++;
    }
t--;

```

Рисунок А.3.6 – Метод аппроксимации Фогеля.

```

//если минимальных несколько
if (t > 1) {
    //минимальные элементы без повторений
    int **mmmc = new int *[3];
    for (int i = 1; i <= 3; i++) mmmc[i] = new int[t];
    bool f;
    int g = 1;
    mmmc[1][1] = mmc[1][1];
    mmmc[2][1] = mmc[2][1];
    mmmc[3][1] = mmc[3][1];
    for (int i = 2; i <= t; i++) {
        f = true;
        for (int j = 1; j <= g; j++)
            if ((mmc[2][i] == mmc[2][j]) && (mmc[3][i] == mmc[3][j])) {
                f = false;
            }
        if (f == true) {
            g = g + 1;
            mmmc[1][g] = mmc[1][i];
            mmmc[2][g] = mmc[2][i];
            mmmc[3][g] = mmc[3][i];
        }
    }
}

```

Рисунок А.3.7 – Метод аппроксимации Фогеля.

```

//если минимальных элементов без повторов несколько
if (g > 1) {
    //смотрим по максимальной сумме штрафов
    int sumd = d[mmmc[2][1]] + d[mmmc[3][1]], e1 = 1;
    for (int j = 2; j <= g; j++)
        if (sumd < d[mmmc[2][j]] + d[mmmc[3][j]]) {
            sumd = d[mmmc[2][j]] + d[mmmc[3][j]];
            e1 = j;
        }
    fillx(M, N, A, B, newC, X, mmmc[2][e1], mmmc[3][e1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
//если минимальный элемент без повторов 1
else {
    fillx(M, N, A, B, newC, X, mmmc[2][1], mmmc[3][1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
//если минимальный элемент 1
else {
    fillx(M, N, A, B, newC, X, mmmc[2][1], mmmc[3][1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
}

```

Рисунок А.3.8 – Метод аппроксимации Фогеля.

```

else {
    //максимальный 1, идем по строке
    c1 = maxc;
    if (k1 <= M) {
        //первый минимальный элемент в строке
        for (int j = 1; j <= N; j++)
            if (newC[k1][j] != -1) {
                if (newC[k1][j] < c1) {
                    c1 = newC[k1][j];
                    l1 = j;
                }
            }
        //массив минимальных элементов в строке с координатами
        int **mmmc = new int *[3];
        for (int i = 1; i <= 3; i++) mmmc[i] = new int[N];
        int g = 1;
        mmmc[1][1] = c1;
        mmmc[2][1] = k1;
        mmmc[3][1] = l1;
        g++;
        for (int j = 1; j <= N; j++)
            if ((mmmc[1][1] == newC[k1][j]) && (j != mmmc[3][1])) {
                mmmc[1][g] = newC[k1][j];
                mmmc[2][g] = k1;
                mmmc[3][g] = j;
                g++;
            }
        g--;
    }
}

```

Рисунок А.3.9 – Метод аппроксимации Фогеля.

```

//если минимальных элементов в строке несколько
if (g > 1) {
    int sumd = d[mmmc[2][1]] + d[mmmc[3][1]], e1 = 1;
    for (int j = 2; j <= g; j++)
        if (sumd < d[mmmc[2][j]] + d[mmmc[3][j]]) {
            sumd = d[mmmc[2][j]] + d[mmmc[3][j]];
            e1 = j;
        }
    fillx(M, N, A, B, newC, X, mmmc[2][e1], mmmc[3][e1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
//если минимальный элемент в строке 1
else {
    fillx(M, N, A, B, newC, X, mmmc[2][1], mmmc[3][1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
}

```

Рисунок А.3.10 – Метод аппроксимации Фогеля.

```

else {
    //максимальный 1, идем по столбцу
    k1 = k1 - M;
    //первый минимальный элемент в строке
    for (int i = 1; i <= M; i++)
        if (newC[i][k1] != -1) {
            if (newC[i][k1] < c1) {
                c1 = newC[i][k1];
                l1 = i;
            }
        }

    //массив минимальных элементов в столбце с координатами
    int **mmmc = new int *[3];
    for (int i = 1; i <= 3; i++) mmmc[i] = new int[M];
    int g = 1;
    mmmc[1][1] = c1;
    mmmc[2][1] = l1;
    mmmc[3][1] = k1;
    g++;
    for (int i = 1; i <= M; i++)
        if ((mmmc[1][1] == newC[i][k1]) && (i != mmmc[2][1])) {
            mmmc[1][g] = newC[i][k1];
            mmmc[2][g] = i;
            mmmc[3][g] = k1;
            g++;
        }
    g--;
}

```

Рисунок А.3.11 – Метод аппроксимации Фогеля.

```

//если минимальных элементов в столбце несколько
if (g > 1) {
    int sumd = d[mmmc[2][1]] + d[mmmc[3][1]], e1 = 1;
    for (int j = 2; j <= g; j++)
        if (sumd < d[mmmc[2][j]] + d[mmmc[3][j]]) {
            sumd = d[mmmc[2][j]] + d[mmmc[3][j]];
            e1 = j;
        }
    fillx(M, N, A, B, newC, X, mmmc[2][e1], mmmc[3][e1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
//если минимальный элемент в столбце 1
else {
    fillx(M, N, A, B, newC, X, mmmc[2][1], mmmc[3][1]); //вычисление X, изменение вспомогательной матрицы
    count++;
}
}
}

// если осталась незаполненная клетка
for (int i = 1; i <= M; i++) {
    for (int j = 1; j <= N; j++)
        if ((A[i] != 0) && (B[j] != 0)) {
            X[i][j] = A[i];
            A[i] = A[i] - B[j];
        }
}
}

```

Рисунок А.3.12 – Метод аппроксимации Фогеля.

```

//вычисление X, изменение вспомогательной матрицы для метода аппроксимаций Фогеля
void fillx(int M, int N, float A[], float B[], float *nC[], float *X[], int I, int J) {
    //если запасы и потребности не равны 0 одновременно, то X=мин(A,B)
    if ((A[I] != 0) && (B[J] != 0)) {
        if (A[I] < B[J]) {
            X[I][J] = A[I];
            B[J] = B[J] - A[I];
            A[I] = 0;
        }
        else {
            X[I][J] = B[J];
            A[I] = A[I] - B[J];
            B[J] = 0;
        }
    }
    //если запасы равны 0, то вычеркиваем строку
    if (A[I] == 0) {
        for (int j = 1; j <= N; j++)
            nC[I][j] = -1;;
    }
    //если запросы равны 0, то вычеркиваем столбец
    if (B[J] == 0) {
        for (int i = 1; i <= M; i++)
            nC[i][J] = -1;
    }
}

```

Рисунок А.3.13 – Метод аппроксимации Фогеля.

```

//метод потенциалов
void mp(int M, int N, float A[], float B[], float *C[], float *X[]) {
    int *u, *v; //массивы потенциалов
    u = new int[M];
    v = new int[N];
    bool *ub; //вспомогательные массивы
    bool *vb;
    ub = new bool[M];
    vb = new bool[N];
    metka:
    for (int i = 1; i <= M; i++) ub[i] = false;
    for (int i = 1; i <= N; i++) vb[i] = false;
    u[1] = 0;
    ub[1] = true;
    int tmp = 1;
    //вычисление потенциалов
    while (tmp <= M + N) {
        for (int i = 1; i <= M; i++)
            if (ub[i] == true)
                for (int j = 1; j <= N; j++)
                    if (X[i][j] != -1)
                        if (vb[j] == false) {
                            v[j] = C[i][j] - u[i];
                            vb[j] = true;
                        }
                for (int j = 1; j <= N; j++)
                    if (vb[j] == true)
                        for (int i = 1; i <= M; i++)
                            if (X[i][j] != -1)
                                if (ub[i] == false) {
                                    u[i] = C[i][j] - v[j];
                                    ub[i] = true;
                                }
            }
        tmp++;
    }
};

```

Рисунок А.4.1 – Метод потенциалов.

```

int **delta;
delta = new int *[M];
for (int i = 1; i <= M; i++) delta[i] = new int[N]; //относительные оценки
for (int i = 1; i <= M; i++)
    for (int j = 1; j <= N; j++)
        if (X[i][j] == -1) {
            //вычисление относительных оценок для каждой свободной клетки
            delta[i][j] = C[i][j] - (u[i] + v[j]);
        }

//проверка условия оптимальности
bool flag = true;
for (int i = 1; i <= M; i++)
    for (int j = 1; j <= N; j++)
        if (X[i][j] == -1) {
            if (delta[i][j] < 0)
                flag = false;
        }

if (flag == true) {
}
else {
    int mindelta = 0;
    int k = 0, l = 0;
    for (int i = 1; i <= M; i++)
        for (int j = 1; j <= N; j++)
            if (X[i][j] == -1)
                if (delta[i][j] < mindelta) {
                    mindelta = delta[i][j]; //самая большая из отрицательных перевозок
                    k = i;
                    l = j;
                }

    float **Y; //массив для хранения цикла перераспределения поставок
    Y = new float *[M];
    for (int i = 1; i <= M; i++) Y[i] = new float[N];
    for (int i = 1; i <= M; i++)
        for (int j = 1; j <= N; j++)
            Y[i][j] = 0;
}

```

Рисунок А.4.2 – Метод потенциалов.

```

//Ищем цикл для ячейки с оценкой D[i][j]:
float find1;
find1 = find_gor(k, 1, k, 1, M, N, X, Y, 0, -1); //поиск цикла, возвращает минимальный элемент цикла, стоящий в клетке со знаком "-"
for (int i = 1; i <= M; i++)
    for (int j = 1; j <= N; j++)
        if (Y[i][j] != -1) {
            X[i][j] = X[i][j] + Y[i][j]; //меняем опорный планперевозок
            if ((i == k) && (j == 1)) X[i][j] = X[i][j] + 1; //добавляем 1 (т.к. до этого было -1)
            if ((Y[i][j] <= 0) && (X[i][j] == 0)) X[i][j] = -1; //если ячейка обнулилась, то выбрасываем её
        }
X[k][1] = Y[k][1];
goto metka;
}
}

```

Рисунок А.4.3 – Метод потенциалов.

```

//Функция поиска ячеек, подлежащих циклу перераспределения (вдоль строки)
float find_gor(int i_next, int j_next, int im, int jm, int n, int m, float *X[], float *Y[], int odd, float Xmin) {
    float rez = -1;
    for (int j = 1; j <= m; j++) //идём вдоль строки, на которой стоим
        //ищем заполненную ячейку(кроме той, где стоим) или начальная ячейка(но уже в конце цикла: odd!=0)
        if (((X[i_next][j] >= 0) && (j != j_next)) || ((j == jm) && (i_next == im) && (odd != 0))) {
            odd++; //номер ячейки в цикле перерасчёта(начало с нуля)
            int Xmin_old = -1;
            if ((odd % 2) == 1) { //если ячейка нечётная в цикле ( начальная- нулевая )
                Xmin_old = Xmin; //запоминаем значение минимальной поставки в цикле (на случай отката назад)
                if (Xmin < 0) Xmin = X[i_next][j]; //если это первая встреченная ненулевая клетка
            }
            else if ((X[i_next][j] < Xmin) && (X[i_next][j] >= 0))
                Xmin = X[i_next][j];
        }
        if ((j == jm) && (i_next == im) && ((odd % 2) == 0)) { //если замкнулся круг и цикл имеет чётное число ячеек
            Y[im][jm] = Xmin; //значение минимальной поставки, на величину которой будем изменять
            return Xmin;
        }
        //если круг еще не замкнулся - переходим к поиску по вертикали:
        else rez = find_ver(i_next, j, im, jm, n, m, X, Y, odd, Xmin); //рекурсивный вызов
        if (rez >= 0) { //обратный ход рекурсии(в случае если круг замкнулся)
            //для каждой клетки цикла заполняем матрицу перерасчёта поставок:
            if (odd % 2 == 0) Y[i_next][j] = Y[im][jm]; //в чётных узлах прибавляем
            else Y[i_next][j] = -Y[im][jm]; //в нечётных узлах вычитаем
            break;
        }
        else { //откат назад в случае неудачи(круг не замкнулся):
            odd--;
            if (Xmin_old >= 0) //если мы изменяли Xmin на этой итерации
                Xmin = Xmin_old;
        }
    }
    return rez; //если круг замкнулся - возвращает найденное минимальное значение поставки в нечётных ячейках цикла, если круг не замкнулся, то возвращает -1.
}

```

Рисунок А.4.4 – Метод потенциалов.

```

//Функция поиска ячеек, подлежащих циклу перераспределения (вдоль столбца)
float find_ver(int i_next, int j_next, int im, int jm, int n, int m, float *X[], float *Y[], int odd, float Xmin) {
    float rez = -1;
    int i;
    for (i = 1; i <= n; i++) //идём вдоль столбца, на котором стоим
        //ищем заполненную ячейку(кроме той, где стоим) или начальная ячейка(но уже в конце цикла: odd!=0)
        if (((X[i][j_next] >= 0) && (i != i_next)) || ((j_next == jm) && (i == im) && (odd != 0))) {
            odd++; //номер ячейки в цикле перерасчёта(начало с нуля)
            int Xmin_old = -1;
            if ((odd % 2) == 1) { //если ячейка нечётная в цикле ( начальная- нулевая )
                Xmin_old = Xmin; //запоминаем значение минимальной поставки в цикле (на случай отката назад)
                if (Xmin < 0) Xmin = X[i][j_next]; //если это первая встреченная ненулевая клетка
            }
            else if ((X[i][j_next] < Xmin) && (X[i][j_next] >= 0))
                Xmin = X[i][j_next];
        }
        if ((i == im) && (j_next == jm) && ((odd % 2) == 0)) { //если замкнулся круг и цикл имеет чётное число ячеек
            Y[im][jm] = Xmin; //Значение минимальной поставки, на величину которой будем изменять
            return Xmin;
        }
        //если круг еще не замкнулся - переходим к поиску по горизонтали:
        else rez = find_gor(i, j_next, im, jm, n, m, X, Y, odd, Xmin); //рекурсивный вызов
        if (rez >= 0) { //обратный ход (в случае если круг замкнулся)
            //для каждой ячейки цикла заполняем матрицу перерасчёта поставок:
            if (odd % 2 == 0) Y[i][j_next] = Y[im][jm]; //эти прибавляются
            else Y[i][j_next] = -Y[im][jm]; //эти вычитаются
            break;
        }
        else { //откат назад в случае неудачи (круг не замкнулся):
            odd--;
            if (Xmin_old >= 0) //если мы изменяли Xmin на этой итерации
                Xmin = Xmin_old;
        }
    }
    return rez;
}

```

Рисунок А.4.5 – Метод потенциалов.

Отчет о проверке на заимствования №1



Автор: Мендинов Максим Игоревич

Проверяющий: Мендинов Максим (mmendinov@gmail.com / ID: 6426428)

Отчет предоставлен сервисом «Антиплагиат» - users.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 9
Начало загрузки: 09.06.2021 08:45:46
Длительность загрузки: 00:00:01
Имя исходного файла: ВКР Мендинов
печатать.pdf
Название документа: ВКР Мендинов печать
Размер текста: 61 кБ
Тип документа: Выпускная
квалификационная работа
Символов в тексте: 62544
Слов в тексте: 7792
Число предложений: 455

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 09.06.2021 08:45:48
Длительность проверки: 00:00:17
Корректировка от 09.06.2021 09:09:38
Комментарии: не указано
Модули поиска: Интернет

ЗАИМСТВОВАНИЯ

3.03%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

0%

ОРИГИНАЛЬНОСТЬ

96,97%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Самоцитирования — доля фрагментов текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа, по отношению к общему объему документа.

Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.

Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.

Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.

Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.

Заимствования, самоцитирования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.

Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Источник	Актуален на	Модуль поиска
[01]	1.1%	170_153_96_0_0.600_59622132 Рыхлов В.С., Корнев В.В., Курдюмов В.П. Прикладные методы оптимизации. 2-е издание. 2012 http://sgu.ru	05 Дек 2020	Интернет
[02]	1.01%	Транспортная задача http://ru.convdocs.org	07 Июл 2016	Интернет
[03]	0.92%	Использование систем мониторинга в транспортной логистике https://cyberleninka.ru	02 Мая 2020	Интернет

Научный руководитель
Зул' / Е. А. Мельникова