

Санкт-Петербургский политехнический университет
Высшая школа прикладной математики и вычислительной физики, ФизМех

Направление подготовки
«01.03.02 Прикладная математика и информатика»
Специальность «Системное программирование»

Лабораторная работа №2
тема **"Решение задач транспортного типа"**
дисциплина "Методы оптимизации"

Выполнили студенты гр. 5030102/00201

Гвоздев С.Ю.,
Солин И.М.
Хламкин Е.М.

Преподаватель:

Родионова Е.А.

Санкт-Петербург

2023

Содержание

1	Постановка задачи	3
2	Исследование применимости метода	3
2.1	Применимость метода потенциалов	3
2.2	Применимость для метода перебора крайних точек и симплекс метода .	4
3	Описание решения транспортной задачи	4
3.1	Приведение задачи к закрытому виду	4
3.2	Нахождение первого базисного решения методом северо-западного угла .	5
3.3	Метод потенциалов	5
3.3.1	Блок-схема	5
3.3.2	Алгоритм проверки решения на оптимальность	7
3.3.3	Алгоритм поиска цикла пересчета	7
3.3.4	Алгоритм обновления базиса	8
4	Практическое решение поставленной задачи	8
4.1	Метод потенциалов для поставленной задачи	8
4.2	Симплекс- Метод для поставленной задачи	8
4.3	Решение поставленной задачи методом перебора крайних точек	11
5	Обоснование достоверности полученного решения	11
5.1	Сравнительный анализ	11
5.2	Теоретическая оценка алгоритмов	12
6	Дополнительные исследования. Пример приведения задачи в открытой форме к задаче в закрытой форме	13
7	Выводы	13
8	Приложения	14
9	Библиографический список	14

1 Постановка задачи

Решить задачу транспортного типа. Дано: n пунктов хранения, в которых сосредоточен однотипный груз; m пунктов назначения, в которые необходимо груз доставить. Также известны:

- a_i - количество груза в i -ом пункте хранения
- b_j - количество груза, в котором нуждается j -й пункт назначения
- c_{ij} - стоимость перевозки единицы груза из i -ого в j -ый пункт
- x_{ij} - план перевозок
- $\sum_{j=1}^n x_{ij} = a_i$ - ограничение на поставщиков
- $\sum_{i=1}^n x_{ij} = b_j$ - ограничения на потребителей
- $x_{ij} \geq 0$

Необходимо составить план перевозок (определить x_{ij}) так, чтобы минимизировать суммарную стоимость перевозок, то есть $\sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{ij} \rightarrow \min$, где x_{ij} - количество груза, перевезенного из i -ого в j -ый пункт.

Условие транспортной задачи удобно задать в виде таблицы:

	b_1	b_2	b_3	b_4	b_5	
a_1	14	7	4	8	3	23
a_2	9	2	2	12	10	25
a_3	17	7	9	11	10	5
a_4	13	7	12	14	5	8
	20	10	12	7	12	

Необходимо:

1. Решить транспортную задачу методом потенциалов с выбором начального приближения методом северо-западного угла.
2. Решить эту же задачу симплекс методом и сравнить результаты.
3. Решить эту же задачу методом перебора крайних точек.
4. Автоматизировать приведение исходной задачи к закрытому виду.
5. Описать алгоритм построения цикла пересчета.

2 Исследование применимости метода

2.1 Применимость метода потенциалов

Для того чтобы транспортная задача была поставлена корректно, необходимо выполнение следующих условий:

- $x_{ij} \geq 0$, где $i = \overline{1, n}, j = \overline{1, m}$
- $\sum_{i=1}^n a_i \geq \sum_{j=1}^m b_j$

Для того чтобы применять заданные методы к решению транспортной задачи, она должна быть приведена к закрытому виду: $\sum_{i=1}^n a_i = \sum_{j=1}^m b_j$

Проверим эти условия:

1. $x_{ij} \geq 0$, где $i = \overline{1, n}, j = \overline{1, m}$, задается в коде программы при определении данных переменных
2. $\sum_{i=1}^n a_i = 23 + 25 + 5 + 8 = 61$ и $\sum_{j=1}^m b_j = 20 + 10 + 12 + 7 + 12 = 61$. Следовательно, задача задана в закрытом виде

Таким образом, заданные методы применимы к данной задаче.

2.2 Применимость для метода перебора крайних точек и симплекс метода

Для того чтобы применить данный метод к задаче, заданной в табличном виде, нужно преобразовать условие к СЛАУ и привести задачу к каноническому виду.

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 23 \\ x_6 + x_7 + x_8 + x_9 + x_{10} = 25 \\ x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 5 \\ x_{16} + x_{17} + x_{18} + x_{19} + x_{20} = 8 \\ x_1 + x_6 + x_{11} + x_{16} = 20 \\ x_2 + x_7 + x_{12} + x_{17} = 10 \\ x_3 + x_8 + x_{13} + x_{18} = 12 \\ x_4 + x_9 + x_{14} + x_{19} = 7 \\ x_i \geq 0, i = \overline{1, n}, n = 20 \end{cases} \quad (1)$$

Функция цели примет вид $\sum_{i=1}^n c_i x_i \rightarrow \min$, где c_i - соответствующие коэффициенты транспортной таблицы.

3 Описание решения транспортной задачи

3.1 Приведение задачи к закрытому виду

Для того, чтобы приступить к решению транспортной задачи, необходимо в первую очередь привести ее к закрытому виду

Input: транспортная задача открытого вида

Output: транспортная задача закрытого вида

Алгоритм

1. Вычислить общий объем товара поставщиков и товара потребителей
2. Добавить столбец или строку с нулевой стоимостью перевозки:

- Если превышают запасы, то добавить фиктивного потребителя (нулевой столбец). Количество товара, необходимое для доставки фиктивному потребителю - модуль разности объемов поставщиков и объемов потребителей
- Если превышают потребности, то добавить фиктивного поставщика (нулевую строку). Количество товара, которое необходимо отправить от фиктивного поставщика - модуль разности объемов поставщиков и объемов потребителей

3.2 Нахождение первого базисного решения методом северо-западного угла

Input: массивы a_i, b_j - запасов и потребностей грузов соответственно, $i = \overline{1, n}, j = \overline{1, m}$

Output: x_{ij} - начальный план перевозок (не является оптимальным, т.к. не учитывается матрица стоимости перевозки c_{ij})

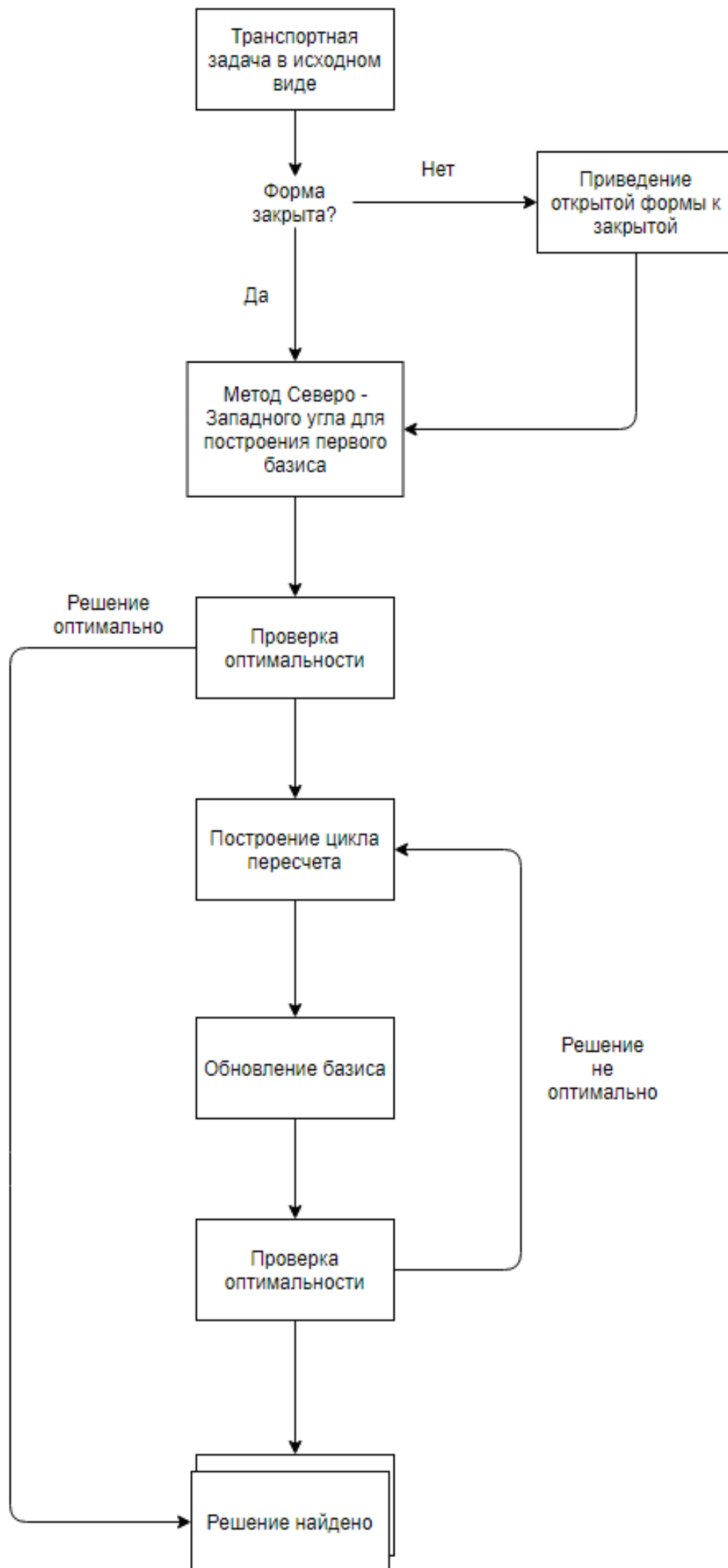
Алгоритм:

1. Двигаемся по таблице $n * m$ - размера с верхнего левого угла, заполняя ее ячейки значениями соответствующих объемов перевозок, в северо-западном направлении
2. Находим максимально возможный объем груза для соответствующей ячейки (i, j) посредством подсчета $\min \{a_i, b_j\}$ и заполняем этим значением данную ячейку
3. Вычитаем найденный объем груза из значений соответствующих полей запаса и потребности
4. Если $a_i \neq 0$, то двигаемся вправо по матрице. Если $b_j \neq 0$, то двигаемся вниз по матрице. Иначе двигаемся по диагонали, добавляя 0 в угловую клетку

3.3 Метод потенциалов

3.3.1 Блок-схема

На блок схеме изобразим основные модули программы, реализующей метод потенциалов. Далее, в отдельных разделах, раскроем их смысл



3.3.2 Алгоритм проверки решения на оптимальность

Алгоритм состоит из следующих этапов:

1. Нахождение значений потенциалов. Сначала находится первая непустая ячейка таблицы базисных переменных. Значению потенциала u присваивается 0, значению соответствующего потенциала v - значение ячейки весовой матрицы. Далее алгоритм принципиально повторяет алгоритм поиска цикла пересчета. (см. далее) Точно так же создаются две функции. Одна - бежит по строкам, другая - по столбцам. В каждой ячейке происходит проверка: если соответствующая базисная ячейка не пуста и какой-то из потенциалов еще не заполнен, то происходит его заполнение и последующий запуск процедуры, изменяющей направление прохода ($line \rightarrow column$ or $column \rightarrow line$). Аналогично алгоритму поиска цикла пересчета, этот этап отрабатывает за $O((m + n)^2)$
2. Пересчет разности потенциалов для всех небазисных элементов.
3. Проверка разностей потенциалов. Если оказывается, что она больше веса, в соответствующей клетке матрицы, то план признаётся не оптимальным. Выполняется переход к построению цикла пересчёта.

3.3.3 Алгоритм поиска цикла пересчета

После того, как проверка на оптимальность дала отрицательный результат, необходимо найти переменную, которая будет выведена из базиса (переменная, вводимая в базис определяется во время проверки на оптимальность). Для этого построим цикл пересчета по следующим правилам:

1. Циклом является последовательность ячеек таблицы (переменных), при которой любые две соседние ячейки стоят либо в одной строке, либо в одном столбце
2. Первая ячейка совпадает с последней в данной последовательности
3. Первая ячейка берется как вводимая переменная
4. Поворот цикла может совершать только в базисных переменных
5. Небазисные переменные (кроме первого элемента последовательности) за ненадобностью в цикл не включаются

Для нахождения такого цикла будем применять следующий незамысловатый алгоритм, основанный на рекурсионных вызовах:

1. Создаем пустой массив *LoopArray*, который на выходе будет представлять из себя искомый цикл
2. Добавляем первым элементом в *LoopArray* вводимую переменную
3. Запускаем процедуру *FindNewTurnInColumn*, которая принимает на вход объект транспортной задачи и номер колонки. Возвращает истинностное значение: *True*, если цикл найден, иначе - *False*. Сейчас номер колонки - это колонка, в которой стоит вводимая переменная. Данная процедура ищет в данной колонке базисные переменные. Пробегая по элементам колонки, мы проверяем следующие условия:

- (a) Если переменная нашлась, и ее еще нет в массиве, то добавляем ее в *LoopArray* и запускаем процедуру *FindNewTurnInLine* (принцип ее работы повторяет принцип работы *FindNewTurnInColumn*), передавая ей на вход номер строки последнего элемента в *LoopArray*. Если *FindNewTurnInLine* вернула *True*, то решение найдено - вернуть *True* (это не тавтология!).
- (b) Если же при проходе по колонке мы встретили самый первый элемент нашего массива *LoopArray*, то возвращаем *True*
- (c) Если оказалось, что после первых двух проверок решение все еще не нашлось, мы переходим к рассмотрению следующего элемента колонки
- (d) Если же все элементы колонки уже просмотрены, то из *LoopArray* удаляется последний элемент и возвращается *False*

4. Если *FindNewTurnInColumn* вернула *False*, то запускаем *FindNewTurnInLine*

Несколько слов о скорости работы вышеуказанного алгоритма:

- 1. В худшем случае в цикле будет содержаться $m + n$ элементов (базисные + вводимая)
- 2. На каждом элементе цикла (при поиске следующего элемента цикла) алгоритм рассматривает не более чем $m + n$ ячеек.
- 3. Получается, что в самом худшем случае поиск цикла пересчета займет $(m + n)^2$ шагов. В общем-то, есть основания полагать, что это хороший результат.

3.3.4 Алгоритм обновления базиса

Для каждого элемента в найденном цикле пересчета проделываем следующую процедуру:

- 1. Если элемент стоит на нечетной позиции в цикле пересчета (1, 3...) запоминаем его как "отрицательного"
- 2. Если элемент стоит на четной позиции в цикле пересчета (2, 4...) запоминаем его как "положительного"

Среди всех 'положительных' элементов выбираем тот, который имеет наименьшее значение базисной переменной: $\theta = \min$ ("положительные переменные цикла")

Для каждого элемента в найденном цикле пересчета проделываем следующую процедуру:

- 1. Если элемент 'отрицательный', то вычитаем из него θ
- 2. Если элемент 'положительный', то прибавляем к нему θ

4 Практическое решение поставленной задачи

На следующих двух страницах соответственно:

4.1 Метод потенциалов для поставленной задачи

4.2 Симплекс- Метод для поставленной задачи

До проверки на закрытость:
Объем груза, хранящийся у поставщиков:
[23. 25. 5. 8.]
Объем груза, который необходимо доставить каждому из производителей:
[20. 10. 12. 7. 12.]
Весовая матрица:

	14.0		7.0		4.0
	9.0		2.0		2.0
	17.0		7.0		9.0
	13.0		7.0		12.0

После проверки на закрытость:
Объем груза, хранящийся у поставщиков:
[23. 25. 5. 8.]
Объем груза, который необходимо доставить каждому из производителей:
[20. 10. 12. 7. 12.]
Весовая матрица:

	14.0		7.0		4.0
	9.0		2.0		2.0
	17.0		7.0		9.0
	13.0		7.0		12.0

Это не оптимальный план

Текущие затраты: 502.0

Матрица базисных элементов:

	20.0		3.0		None		None		None	
	None		7.0		12.0		6.0		None	
	None		None		None		1.0		4.0	
	None		None		None		None		8.0	

Пересчет цикла:

	20.0		3.0		*		*		NEW	
	None		7.0		*		6.0		*	
	None		None		None		1.0		4.0	
	None		None		None		None		8.0	

Это не оптимальный план

Текущие затраты: 463.0

Матрица базисных элементов:

	20.0		None		None		None		3.0	
	None		10.0		12.0		3.0		None	
	None		None		None		4.0		1.0	
	None		None		None		None		8.0	

Пересчет цикла:

	20.0		*		*		*		3.0	
	NEW		*		*		3.0		*	
	None		None		None		4.0		1.0	
	None		None		None		None		8.0	

Это не оптимальный план

Текущие затраты: 450.0

Матрица базисных элементов:

	19.0		None		None		None		4.0	
	1.0		10.0		12.0		2.0		None	
	None		None		None		5.0		None	
	None		None		None		None		8.0	

Пересчет цикла:

	19.0		*		*		NEW		4.0	
	1.0		*		*		2.0		None	
	None		None		None		5.0		None	
	None		None		None		None		8.0	

Это не оптимальный план

Текущие затраты: 432.0

Матрица базисных элементов:

	17.0		None		None		2.0		4.0	
	3.0		10.0		12.0		None		None	
	None		None		None		5.0		None	
	None		None		None		None		8.0	

Пересчет цикла:

	17.0		*		NEW		2.0		4.0	
	3.0		*		12.0		None		None	
	None		None		None		5.0		None	
	None		None		None		None		8.0	

Это не оптимальный план

Текущие затраты: 396.0

Матрица базисных элементов:

	5.0		None		12.0		2.0		4.0	
	15.0		10.0		None		None		None	
	None		None		None		5.0		None	
	None		None		None		None		8.0	

Пересчет цикла:

	5.0		*		*		2.0		4.0	
	15.0		10.0		None		*		None	
	None		NEW		*		5.0		None	
	None		None		None		None		8.0	

Опорный план найден

Текущие затраты: 381.0

Итоговая матрица базисных элементов:

	None		None		12.0		7.0		4.0	
	20.0		5.0		None		None		None	
	None		5.0		None		0.0		None	
	None		None		None		None		8.0	

---ИНИЦИАЛИЗАЦИЯ СИМПЛЕКС-МЕТОДА. ПРОВЕРКА СУЩЕСТВОВАНИЯ РЕШЕНИЯ---

---СИМПЛЕКС - МЕТОД---

Матрица ограничений:

1.0 0.0 1.0 1.0 0.0 0.0 -1.0 0.0 0.0 -1.0 0.0 -1.0 0.0 0.0 -1.0 0.0 -1.0 0.0 0.0 -1.0 = 1.0	1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.0 -1.0 -1.0 -1.0 0.0 -1.0 -1.0 -1.0 = 7.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 = 5.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 = 8.0	0.0 0.0 0.0 0.0 1.0 0.0 -0.0 0.0 0.0 1.0 0.0 -0.0 0.0 0.0 1.0 0.0 -0.0 0.0 0.0 = 12.0	0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 = 10.0	0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 = 12.0	-1.0 0.0 -1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0 0.0 1.0 = 6.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 = 0.0		9.0 0.0 6.0 0.0 0.0 0.0 -9.0 0.0 0.0 3.0 0.0 -12.0 -1.0 -9.0 -5.0 0.0 -8.0 6.0 -2.0 -6.0 -> 462.0
---------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------	--	---------------------------------------------------------------------------------------------------------------------------------------------

Базисные индексы:

[3, 5, 10, 15, 4, 1, 7, 8, 6]

Матрица ограничений:

1.0 0.0 1.0 1.0 0.0 0.0 -1.0 0.0 0.0 -1.0 1.0 0.0 1.0 1.0 0.0 0.0 -1.0 0.0 0.0 -1.0 = 6.0	1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.0 -1.0 -1.0 -1.0 = 12.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 = 5.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 = 8.0	0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 = 12.0	0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 -1.0 0.0 -1.0 -1.0 -1.0 0.0 1.0 0.0 0.0 0.0 = 5.0	0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 = 12.0	-1.0 0.0 -1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 -1.0 0.0 -1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 = 1.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 = 0.0		9.0 0.0 6.0 0.0 0.0 0.0 -9.0 0.0 0.0 3.0 12.0 0.0 11.0 3.0 7.0 0.0 -8.0 6.0 -2.0 -6.0 -> 402.0
-------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	--	------------------------------------------------------------------------------------------------------------------------------------------

Базисные индексы:

[3, 5, 11, 15, 4, 1, 7, 8, 6]

Матрица ограничений:

0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 = 7.0	1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.0 -1.0 -1.0 -1.0 = 12.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 = 5.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 = 8.0	0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 = 12.0	1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.0 -1.0 0.0 0.0 0.0 -1.0 -1.0 0.0 0.0 0.0 -1.0 -1.0 = 4.0	0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 = 12.0	-1.0 0.0 -1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 -1.0 0.0 -1.0 0.0 0.0 0.0 1.0 0.0 1.0 1.0 = 1.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 = 0.0		0.0 0.0 -3.0 0.0 0.0 0.0 0.0 0.0 9.0 12.0 3.0 0.0 2.0 3.0 7.0 0.0 1.0 6.0 7.0 3.0 -> 393.0
---------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	--	--------------------------------------------------------------------------------------------------------------------------------------

Базисные индексы:

[3, 5, 11, 15, 4, 1, 7, 6, 6]

Матрица ограничений:

0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 = 7.0	1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.0 -1.0 -1.0 -1.0 = 12.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 = 5.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 = 8.0	0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 = 12.0	1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.0 -1.0 0.0 0.0 0.0 -1.0 -1.0 0.0 0.0 0.0 -1.0 -1.0 = 4.0	-1.0 -1.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0 1.0 1.0 = 8.0	0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 -1.0 0.0 -1.0 -1.0 -1.0 0.0 1.0 0.0 0.0 0.0 = 5.0	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 = 0.0		3.0 3.0 0.0 0.0 0.0 0.0 0.0 0.0 6.0 9.0 3.0 0.0 2.0 0.0 4.0 0.0 1.0 6.0 4.0 0.0 -> 381.0
---------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------	--	------------------------------------------------------------------------------------------------------------------------------------

Базисные индексы:

[3, 5, 11, 15, 4, 2, 7, 6, 6]

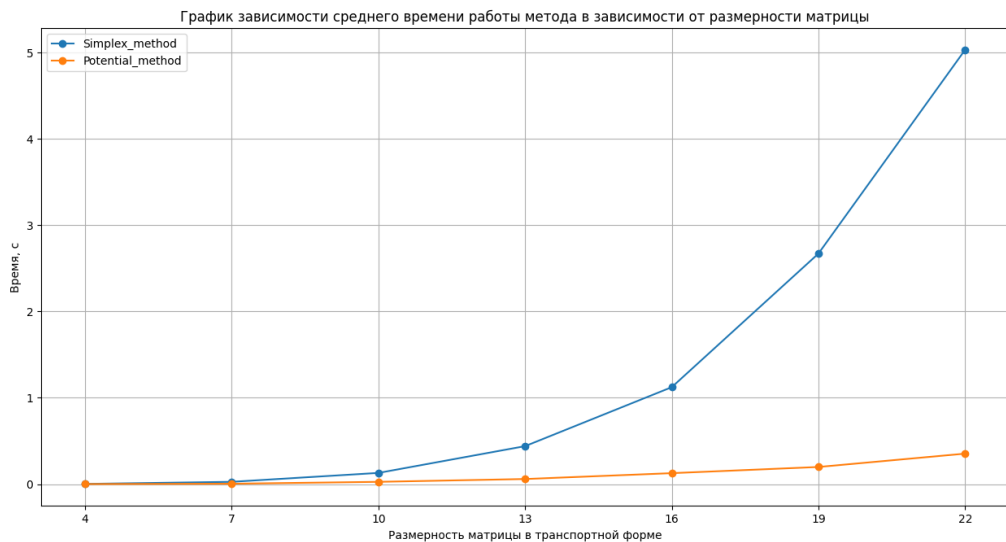
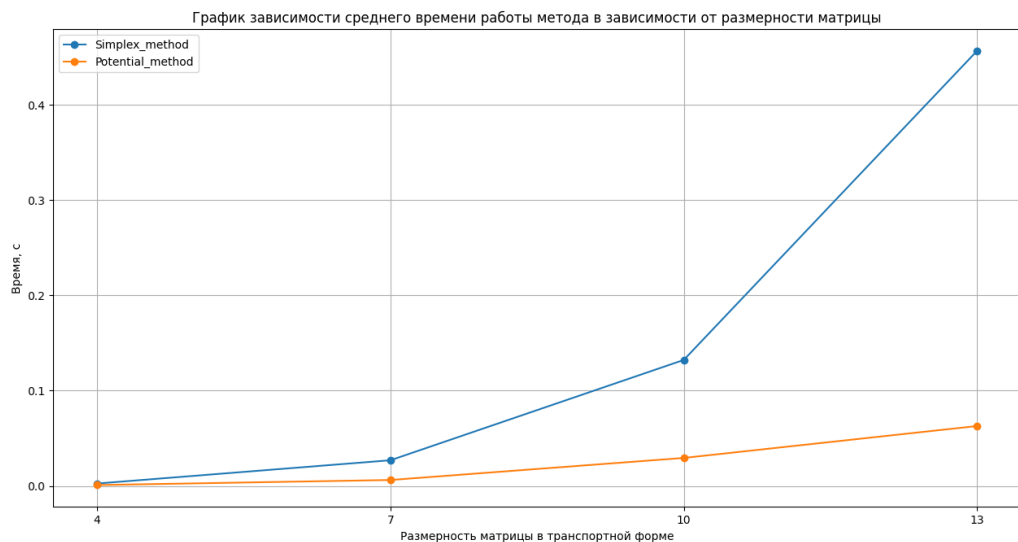
4.3 Решение поставленной задачи методом перебора крайних точек

Метод крайних точек дает тот же верных результат, что и оба вышеупомянутых метода: 381.

5 Обоснование достоверности полученного решения

5.1 Сравнительный анализ

Графики зависимости времени работы методов от размерности матрицы.



Для начала заметим: коль скоро транспортная задача имеет размеры $m * n$, соответствующая задача линейного программирования в канонической форме имеет размеры $(m + n) * (m * n)$. А это весьма внушительные размеры даже для небольших m и n .

Данные два графика изображают собой зависимости времени работы Метода Потенциалов и Симплекс - Метода для разных размерностей матриц - для небольших, и для тех, что побольше. Метод потенциалов решает поставленную задачу существенно

быстрее. Действительно, хоть в худшем случае оба метода и работают за C_{n*m}^{n+m+1} шагов (см. оценки далее), и хоть время, необходимое на каждую из итераций одинаково оценивается в $O((n)^2)$, n для Симплекс - Метода и для Метода Потенциалов различны. На каждой итерации Метода Потенциалов мы пересчитываем матрицу $m * n$, на каждой итерации Симплекс - Метода мы пересчитываем матрицу $(m + n) * (m * n)$. Ровно этим и обусловлено такое поведение графиков.

5.2 Теоретическая оценка алгоритмов

Рассмотрим временные затраты метода потенциалов на каждом из этапов:

1. Проверка оптимальности.
 - (а) Нахождение значений потенциалов. (вычислили выше - $O((m + n)^2)$)
 - (б) Пересчет разности потенциалов: проход почти по всей матрице: $O((m + n)^2)$
2. Поиск цикла пересчета (вычислили выше - $O((m + n)^2)$)
3. Обновление базиса: к каждой базисной переменной прибавляем или вычитаем одно число. $O(m + n)$

Итого: временные затраты на одну итерацию: $O((m + n)^2)$

Всего же возможных итераций не более чем C_{n*m}^{n+m+1} (число возможных базисов). Конечно, в худшем случае при оценке скорости сходимости придется ориентироваться именно на значение C_{n*m}^{n+m+1} . Однако практически число итераций подчиняется чему-то вроде линейного закона (это нестрогое утверждение). И в этом случае уже можно ориентироваться на $O((m + n)^2)$

6 Дополнительные исследования. Пример приведения задачи в открытой форме к задаче в закрытой форме

До проверки на закрытость:

Объем груза, хранящийся у поставщиков:

[23. 25. 5. 15.]

Объем груза, который необходимо доставить каждому из производителей:

[20. 10. 12. 7. 12.]

Весовая матрица:

```
+-----+-----+-----+-----+-----+
| 14.0 | 7.0 | 4.0 | 8.0 | 3.0 |
| 9.0 | 2.0 | 2.0 | 12.0 | 10.0 |
| 17.0 | 7.0 | 9.0 | 11.0 | 10.0 |
| 13.0 | 7.0 | 12.0 | 14.0 | 5.0 |
+-----+-----+-----+-----+-----+
```

После проверки на закрытость:

Объем груза, хранящийся у поставщиков:

[23. 25. 5. 15.]

Объем груза, который необходимо доставить каждому из производителей:

[20. 10. 12. 7. 12. 7.]

Весовая матрица:

```
+-----+-----+-----+-----+-----+-----+
| 14.0 | 7.0 | 4.0 | 8.0 | 3.0 | 0.0 |
| 9.0 | 2.0 | 2.0 | 12.0 | 10.0 | 0.0 |
| 17.0 | 7.0 | 9.0 | 11.0 | 10.0 | 0.0 |
| 13.0 | 7.0 | 12.0 | 14.0 | 5.0 | 0.0 |
+-----+-----+-----+-----+-----+-----+
```

7 Выводы

В работе были проведены исследования по сравнению трёх методов нахождения оптимального плана задачи транспортного типа. (Метод потенциалов, Симплекс-метод, метод перебора крайних точек). Последний показал себя не с лучшей стороны, вынудив авторов ждать решения поставленной задачи три часа (167960 вариантов решения, для каждого решается СЛАУ = грусть). Первые же два заставляют проявлять к себе интерес. Графическая картина результатов работы методов внушает доверие, и предпочтение (не без удовольствия) приходится отдавать методу Потенциалов. Однако заметим вкратце, что транспортная задача - лишь частный случай задачи линейного программирования. Так что, в плане универсальности Симплекс - Метод смело может дать фору методу Потенциалов.

Итог: для задач транспортного типа использовать метод Потенциалов. Симплекс - Метод в обычной интерпретации использовать не рекомендуется

8 Приложения

Реализация программы находится в репозитории GitHub по ссылке:
<https://github.com/IMZolin/transport-task>

9 Библиографический список

1. Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клиффорд. "Алгоритмы. Построение и анализ, 2-е издание" Издательский дом "Вильямс", 2011. – 892–918 с.
URL: https://vk.com/doc191450968_561608466?hash=HUwStWS0yzrW9SaXn8P0Ztaz3gTyMTm&dl=U9ivclLJBeeYQbs3MMhGtwYZ7Mx4nGJe1Tv0Hv56E4z / [Электронный ресурс]. Режим доступа: (Дата обращения: 04.03.2023)
2. Родионова Е.А., Петухов Л.В., Серёгин Г.А. "Методы оптимизации. Задачи выпуклого программирования" Издательство Политехнического университета, Санкт-Петербург, 2014
URL: <https://elib.spbstu.ru/dl/2/i17-98.pdf/info> / [Электронный ресурс]. Режим доступа: (Дата обращения: 10.03.2023)
3. Моисеев Н.Н. "Методы оптимизации"
URL: <https://avidreaders.ru/book/metody-optimizacii-1.html> / [Электронный ресурс]. Режим доступа: (Дата обращения: 07.03.2023)