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 №1
"Решение задачи линейного программирования. Симплекс - Метод. Метод перебора крайних точек"
дисциплина "Методы оптимизации"

Выполнили студенты гр. 5030102/00201 Гвоздев С.Ю., Золин И.М., Хламкин Е.В.

Преподаватель: Родионова Е.А.

Санкт-Петербург

2023

Содержание

1 Постановка задачи

Поставлена задача линейного программирования, состоящая из 6 переменных, 3 равенств, 2 неравенств " $\leq$ ", 1 неравенства " $\geq$ ". Также поставлены ограничения на знаки для 3 переменных:

$$\begin{cases} -23x_1 + 3x_2 + x_3 - 2x_4 - 3x_5 - 5x_6 \leq 17 \\ -18x_1 + 3x_2 + x_3 - 3x_4 - 3x_5 - 4x_6 \leq 12 \\ 13x_1 - 3x_2 - x_3 + 3x_4 + 3x_5 + 3x_6 \geq -13 \\ -8x_1 + 3x_2 + x_3 - 2x_4 - 2x_5 - 2x_6 = 23 \\ -4x_1 + 2x_2 + x_3 - x_4 - x_5 - x_6 = 20 \\ -4x_1 + 2x_2 + 2x_3 - x_4 - x_5 - x_6 = 29 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Функция цели:

$$F(x) = 4x_1 + 6x_2 + 2x_3 + 6x_4 + 5x_5 + 3x_6 \longrightarrow min \tag{1}$$

Необходимо:

1. Решить данную задачу Симплекс - Методом
2. Решить данную задачу методом перебора крайних точек.
3. Провести анализ и сравнение полученных результатов. Привести качественный разбор скорости сходимости Симплекс - Метода и метода перебора крайних точек
4. Построить двойственную задачу к данной

2 Исследование применимости метода

Алгоритм **симплекс-метода** применим к задачам линейного программирования на поиск минимума путем перебора вершин выпуклого многранника в многогранном пространстве.

Метод является универсальным и применим к любой задаче, работает на задачах в канонической форме при всяких вещественных значениях компонент $A \in \mathbb{R}_{m\times n}, b \in \mathbb{R}_m, c \in \mathbb{R}_n$. Матрица A должно иметь ранг m, что гарантирует наличие хотя бы одного опорного вектора.

Данная задача соответствует вышеуказанным условиям, так как формировалась изначально с помощью единичной матрицы и положительного вектора b, а потом с помощью эквивалентных преобразований (сложение строк и умножение строк на число) приводилась к нетривиальной форме.

3 Описание алгоритма

3.1 Алгоритм симплекс метода

3.1.1 Описание

Input: задача линейного программирования в стандартной форме
Output: вектор $x[N]$, который является оптимальным решением задачи линейного программирования, или сообщение о неразрешимости или неограниченности задачи

3.1.2 Блок-схема

3.1.3 Правило Блэнда

Правило Блэнда - усовершенстовввание симплекс-метода, направленное на нахождение решения без зацикливания.(Если Симплекс метод не может завершиться более чем за C_N^M, то он зацикливается) [1]

Обычно целевая функция действительно уменьшается на каждом шаге, и таким образом после ограниченного числа шагов находится оптимальное решение. Однако есть примеры вырожденных линейных программ, на которых исходный симплексный алгоритм зацикливается вечно.

Алгоритм

1. Выбираем небазисный столбец с наименьшим номером
2. Среди всех строк выбираем ту, для которой минимум отношения правой части и коэффициента вводимого столбца в таблице(при условии, что этот коэффицинт больше 0). Если такой минимум достигается на нескольких строках, выбираем ту, которая соотвествует столбцу с наименьшим индексом.

Пример

1. Целевая функция: $c = 10x_1 - 57x_2 - 9x_3 - 24x_4$
   $x_5 = -0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4$
   $x_6 = -0.5x_1 + 1.5x_2 + 0.5x_3 - x_4$
   $x_7 = 1 - x_1$
   (0, 0, 0, 0, [0], [0],[1]) x_1-вх, x_5-вых
2. Целевая функция: $c = 53x_2 + 41x_3 - 204x_4 - 20x_5$
   $x_1 = 11x_2 + 5x_3 - 18x_4 - 2x_5$
   $x_6 = -4x_2 - 2x_3 + 8x_4 + x_5$
   $x_7 = 1 - 11x_2 - 5x_3 + 18x_4 + 2x_5$
   ([0], 0, 0, 0, 0, [0],[1]) x_2-вх, x_6-вых
3. Целевая функция: $c = 14.5x_3 - 98x_4 - 6.75x_5 - 13.25x_6$
   $x_1 = -0.5x_3 + 4x_4 + 0.75x_5 - 2.75x_6$
   $x_2 = -0.5x_3 + 2x_4 + 0.25x_5 - 0.25x_6$
   $x_7 = 1 + 0.5x_3 - 4x_4 - 0.75x_5 - 13.25x_6$
   ([0], [0], 0, 0, 0, 0,[1]) x_3-вх, x_1-вых
4. Целевая функция: $c = -29x_1 + 18x_4 + 15x_5 - 93x_6$
   $x_3 = -2x_1 + 8x_4 + 1.5x_5 - 5.5x_6$
   $x_2 = x_1 - 2x_4 - 0.5x_5 + 2.5x_6$
   $x_7 = 1 - x_1$
   (0, [0], [0], 0, 0, 0,[1]) x_4-вх, x_2-вых
5. Целевая функция: $c = -20x_1 - 9x_2 + 10.5x_5 - 70.5x_6$
   $x_3 = 2x_1 - 4x_2 - 0.5x_5 + 4.5x_6$
   $x_4 = 0.5x_1 - 0.5x_2 - 0.25x_5 + 1.25x_6$
   $x_7 = 1 - x_1$
   (0, 0, [0], [0], 0, 0,[1]) x_5-вх, x_3-вых

6. Следующая поворотная точка позволяет нам выйти из цикла (цикл возникает, если x_6 выбирается для входа).

   Целевая функция: $c = 22x_1 - 93x_2 - 21x_3 + 24x_6$
   $x_5 = 4x_1 - 8x_2 - 2x_3 + 9x_6$
   $x_4 = -0.5x_1 + 1.5x_2 + 0.5x_3 - x_6$
   $x_7 = 1 - x_1$
   (0, 0, 0, [0], [0], 0,[1]) x_1-вх, x_4-вых

7. Целевая функция: $c = -27x_2 + x_3 - 44x_4 - 20x_6$
   $x_5 = 4x_2 + 2x_3 - 8x_4 + x_6$
   $x_1 = 3x_2 + x_3 - 2x_4 - 2x_6$
   $x_7 = 1 - 3x_2 - x_3 + 2x_4 + 2x_6$
   ([0], 0, 0, 0, [0], 0,[1]) x_3-вх, x_7-вых

8. Целевая функция: $c = 1 - 30x_2 - 42x_4 - 18x_6 - x_7$
   $x_5 = 2 - 2x_2 - 4x_4 + 5x_6 - 2x_7$
   $x_1 = 1 - x_7$
   $x_3 = 1 - 3x_2 + 2x_4 + 2x_6 - x_7$
   ([1], 0, [1], 0, [2], 0, 0)

Целевая функция имеет максимум = 1, достигаемый в точке (1, 0, 1, 0, 2, 0, 0)
Заметим, что в вышеописанном процессе решения участвуют только 8 из C_7^3= 35

3.2 Алгоритм перевода из общей в каноническую форму

Input: система уравнений A[M,N] в общей форме
Output: система уравнений A[M,N] в канонической форме

1. Проверяем знаки в системе

2. Если «$\leq$», то к левой части добавляем $w[i]$, если «$\geq$», то из левой части вычитаем $w[i]$, $w[i] \geq 0$.

3. Знаки неравенства в системе заменяем на равенство.

4. Производим замену переменных:
   если $x[i] \leq 0$, то $x'[i] = -x[i] \geq 0$;
   если $x[i]$ любого знака, то $x[i] = u[i] - v[i]$, $v[i], u[i] \geq 0$.

3.3 Алгоритм восстановления решения двойственной задачи по решению прямой

Рассмотрим задачу минимума:

$$min\ c^T[N] \cdot x[N]$$

$$S = \{\ x[N] |\ A[M,N] \cdot x[N] = b[M],\ \ x[N] \geq 0\} \tag{2}$$

В случае задачи максимума, то домножаем целевой вектор на -1.
Для получения двойственной задачи будем использолвать теоремы двойственности.
Найдя решение прямой задачи(с оптимальным значением целевой функции F_{max} и оптимальным планом $x = x_1, x_2, ..., x_n$), подставив в систему ограничений, проверим:
Если

1. i-ое неравенство не является равенством $(a_{1i}x_1 + a_{2i}x_2 + ... + a_{mi}x_n \neq c_i)$, то $y_i = 0$

2. $x_k \neq 0$, то k-ая строка системы ограничений является равенством: $a_{1k}y_1 + a_{2k}y_2 + ... + a_{mk}y_k = c_k$

Система ограничений:

$$\begin{cases} a_{11}y_1 + a_{21}y_2 + ... + a_{m1}y_m \geq c_1 \\ a_{12}y_1 + a_{22}y_2 + ... + a_{m2}y_m \geq c_2 \\ ... \\ a_{1n}y_1 + a_{2n}y_2 + ... + a_{mn}y_m \geq c_n \\ y_1 \geq 0, y_2 \geq 0, ..., y_m \geq 0 \\ x_1, x_2 \geq 0 \end{cases}$$

Составим все строки ограничений, для которых $x_k \neq 0$, получичм систему уравнений, из которых можно найти ненулевые значения переменных y.

По 1-ой теореме двойственности: минимальное значение целевой функции $z_{min} = F_{max} \Rightarrow$ для получения решения двойственной задачи достаточно решить полученную систему линейных уравнений.

Исходная задача, переведенная в двойственную форму:

$$\begin{cases} -23y_1 + -18y_2 + 13y_3 - 8y_4 - 4y_5 - 4y_6 \leq 4 \\ 3y_1 + 3y_2 - 3y_3 + 3y_4 + 2y_5 + 2y_6 \leq 6 \\ y_1 + y_2 + 1y_3 - y_4 + y_5 + 2y_6 = 2 \\ -2y_1 - 3y_2 + 3y_3 - 2y_4 - y_5 - y_6 = 6 \\ -3y_1 - 3y_2 + 3y_3 - 2y_4 - y_5 - y_6 = 5 \\ -5y_1 - 4y_2 + 3y_3 - 2y_4 - y_5 - y_6 = 3 \\ y_1, y_2 \geq 0, y_3 \leq 0 \end{cases}$$

$$F(y) = 17y_1 + 12y_2 - 13y_3 + 23y_4 + 20y_5 + 29y_6 \longrightarrow max \quad (3)$$

3.4 Алгоритм перебора крайних точек

Оптимальный вектор в алгоритме перебора крайних точек является угловой точкой ОДР(области допустимых решений). Если же имеется множество решений, то среди них найдутся угловые точки.

Находя угловые точки - вычисляем в них значяения целевой функции. Далее определяем наименьшее/наибольшее значение функции.

Input:

1. A[M,N] - матрица коэффициентов задачи в каноничесокой форме
2. b[M] - вектор свободных коэффициентов задачи
3. c[N] - вектор коэффициентов функции цели задачи

Output: опорный вектор x[N], минимизирующий целевую функцию

Алгоритм:

1. Генерирование квадратных мтриц, выделямых из A[M,N], где M < N: их количество C_N^M
2. Проверка для каждой матрицы отличие определителя от нуля. Далее, если проверка пройдена, находим решение системы: $A[M, N_k]x[N_k] = b[M]$

3. Проверка на положительность компонент решения. Если проверка пройдена, то дополняем $x[N]$ нулевыми значениями соответсвующих компонент и получаем крайнюю точку.

4. Находим значение функции цели в крайней точке и запоминаем его

5. Повторяем предыдущие шаги

6. Сравниваем значения между собой и выбираем то решение, которое соотвествует наименьшему значению функции цели.

3.5 Решение поставленной задачи

Basic_indexes	0	1	2	3	4	5	6	7	8	9	10	11	12	
10	0	0	0	0	0	0	-1	1	-3	3	1	0	-15	69
11	0	0	0	0	0	0	0	0	-1	1	0	1	-6	24
5	0	0	0	0	-1	1	-1	1	-1	1	0	0	-4	15
0	1	0	0	0	0	0	0	0	0	0	0	0	-1	5
1	0	1	0	0	0	0	0	0	0	0	0	0	0	8
2	0	0	1	-1	0	0	0	0	0	0	0	0	0	9
	0	0	0	8	5	3	0	0	0	0	0	0	4	0

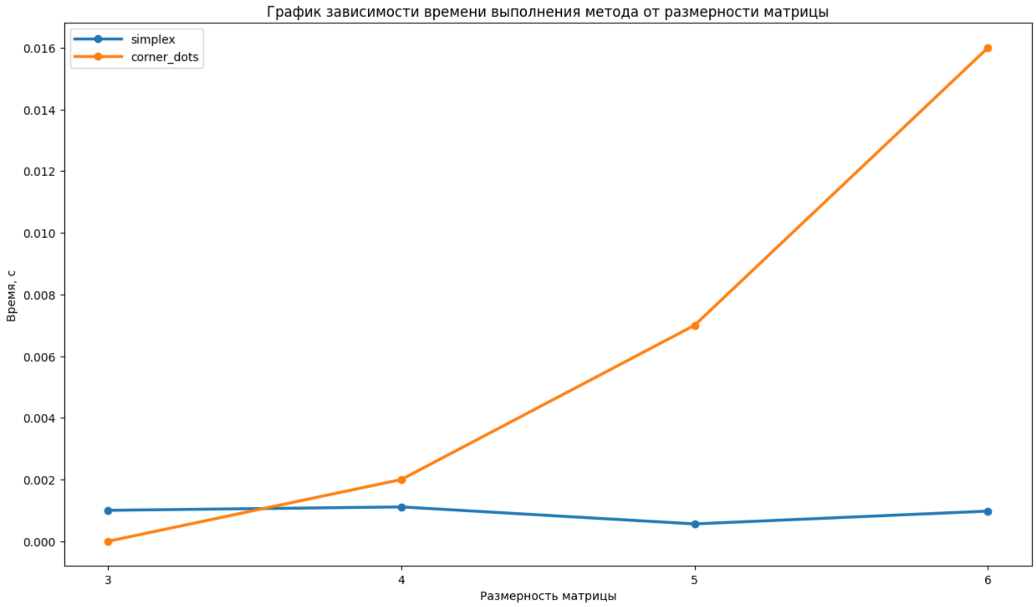
Basic_indexes	0	1	2	3	4	5	6	7	8	9	10	11	12	
10	0	0	0	0	1	-1	0	0	-2	2	1	0	-11	54
11	0	0	0	0	0	0	0	0	-1	1	0	1	-6	24
7	0	0	0	0	-1	1	-1	1	-1	1	0	0	-4	15
0	1	0	0	0	0	0	0	0	0	0	0	0	-1	5
1	0	1	0	0	0	0	0	0	0	0	0	0	0	8
2	0	0	1	-1	0	0	0	0	0	0	0	0	0	9
	0	0	0	8	5	3	0	0	0	0	0	0	4	-86

Здсь первая таблица соответствует процедуре Инициализации Симплекс-Метода (поиск допустимого базиса). Вторая таблица - переход от найденного допустимого базисного решения к новому допустимому базису. Оказывается, что в данном примере Симплекс-Метод решает задачу за одну итерацию.

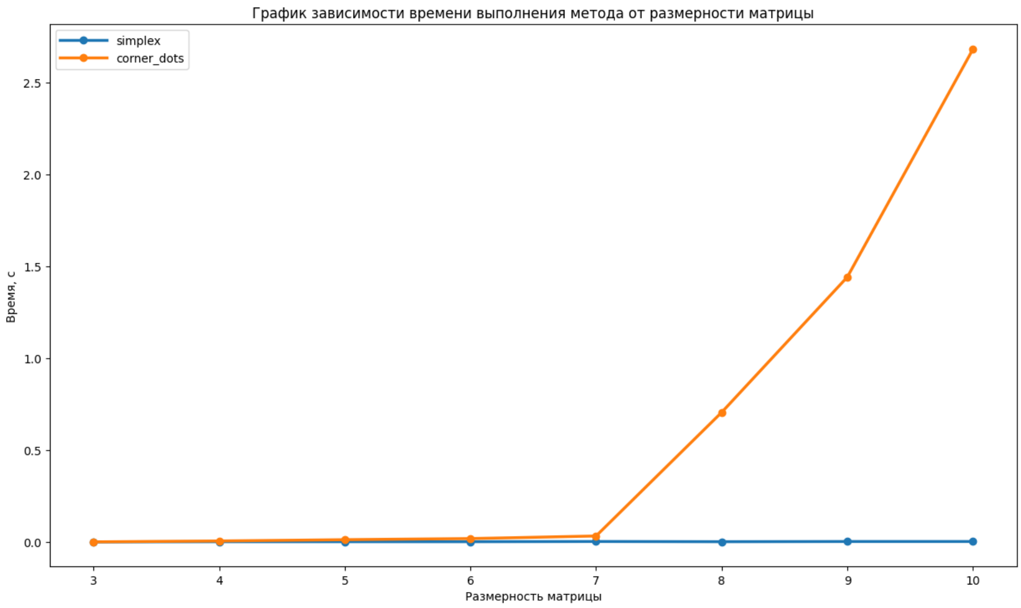
4 Обоснование достоверности полученного решения

4.1 Сравнительный анализ

1. График зависимости времени выполнения метода от размерности матрицы (3,6)

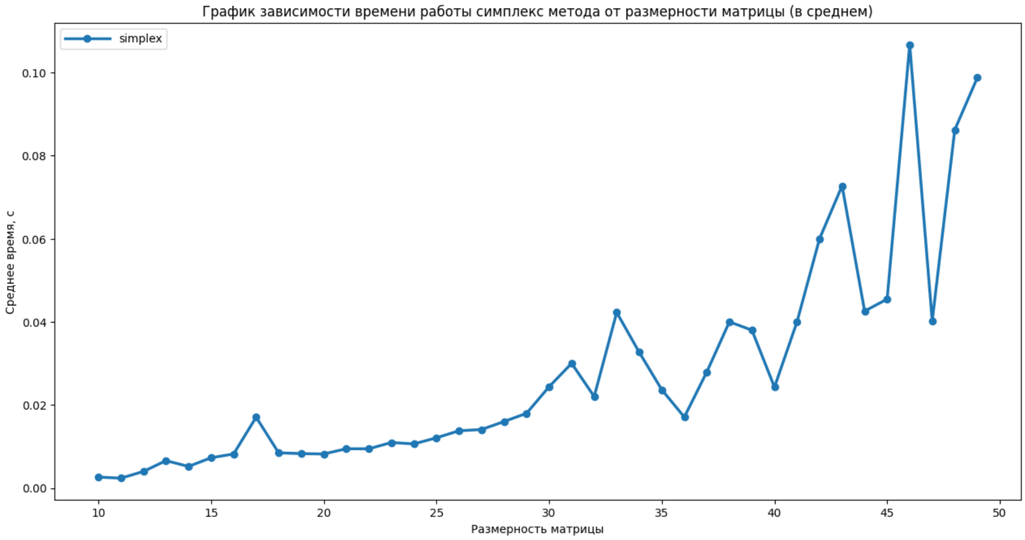


2. График зависимости времени выполнения метода от размерности матрицы (3,10)



Для матриц маленького (<4) ранга Симплекс метод и метод перебора крайних точек, дают похожие результаты по времени. Что логично, ведь методу перебора крайних точек нужно перебрать не так много матриц (максимум 16). В дальнейшем количество матриц для перебора быстро увеличивается и метод перебора крайних точек начинает существенно проигрывать симплекс методу в таблицах. Отрыв, хорошо видно на "графике 2".

3. График зависимости времени выполнения Симплекс метода от размерности матрицы в среднем (10,50)



Был произведён замер времени (10 раз). И взято среднее значение. Видно, что с ростом размерности матрицы время работы Симплекс метода увеличивается (не монотонно). Немонотонность обусловлена возможностью в некоторых тестовых запусках очень быстро получить целевую функцию, удовлетворяющую критериям остановки метода. (Все коэффициенты >0)

4.2 Теоретическая оценка алгоритмов

Инициализация Симплекс метода, зачастую более затратная чем реализация самого Симплекс метода (по времени и памяти) т.к при инициализации добавляются дополнительные переменные (количество зависит от количества отрицательных компонент вектора b) и приходится решать задачу более высокого порядка.

Ищем выводимую строку и вводимый столбец ($O(n)$). Выполняем процедуру pivot (Жордано Гаусс) ($O(n^2)$) Эти операции в худшем случае выполнятся C_n^m раз. Где n - число столбцов, m - число строк. Можно сделать вывод , что в худшем случае симплекс метод работает за $O(2^n)$

5 Дополнительные исследования

Приведем соответствие Симплекс-Метода и Табличного Симплекс-Метода
Каждому шагу Табличного Симплекс-Метода сопоставим шаг обычного Симплекс-Метода

1.1) Предподготовка. Данная в общей форме задача приводится к каноническому виду. После этого выделяется первый базис. Причем коль скоро этот базис состоит из линейно независимых векторов, мы будем приводить эти вектора к единичному виду (м. Жордано-Гаусса). Таким образм подматрица $A[M, N_k]$ будет являтся матрицей перестановок. Никто не мешает представить вектор c таким образом, чтобы он был выражен только через небазисные компоненты. Так и поступим.

1.2) Все образования п.1.1 приводят к эквивалектной задача Л.П., поэтому на вход обоим Симплекс-Методам подается все та же исходная задача

2.1) Проверка условий оптимальности. В Табличном Симплекс-Методе при решении задачи минимизации условием прекращения алгоритма является условие $c^T[L_k] \geq 0$

2.2)

$$y^T[M] = c^T[N_k] * B[N_k, M] = 0$$
$$c^T[L_k] - y^T[M] * A[M, L_k] \geq 0$$
$$c^T[L_k] \geq 0 \Rightarrow \text{условия эквивалентны}$$

3.1) Замена базиса. В задаче минимизации по заданному правилу выбирается вводимый столбец j_k (среди отрицательных компонент вектора c выбирается наибольший по модулю. Координата данного столбца - это номер вводимого столбца). Среди компонент выбранного столбца либо не существует положительных компонент, и тогда решения задачи не ограничено, либо существуют положительные компоненты. Тогда среди отношений $b[i]/A[i, j_k]$ выбирается минимальное положительное (**). Базисная компонента, взятая по номеру найденного отношения - выводимый столбец. Далее - процедура pivot (см. Корман) и обновление вектора c по вышеуказанному правилу (эквивалентное преобразование).

3.2) j_k выбирается по тому же правилу. В обычном Симплекс - Методе:

$$x_{k+1}[N] = x_k[N] - \theta * u_k[N], \text{ где } (*)$$
$$u_k[j_k] = -1,$$
$$u_k[L_k\ j_k] = 0$$
$$u_k[N_k] = B[N_k, M] * A[M, j_k]$$

$B[N_k, M]$ - матрица перестановок, поэтому $x_k[N]$ - вектор, состоящий из нулей и перестановки вектора b (равенство "почти единичной"матрицы, умноженной на $x_k[N_k]$, вектору b). Поэтому правая часть (*) эквивалентна тому, что из вектора $x_k[N_k]$ мы вычитаем вектор $A[M, j_k]$ (дополненный нулями и минус единицей) с коэффиициентом θ. θ подбирается таким образом, чтобы при вычитании обнулить выводимую переменную, при этом оставив базис допустимым и не переведя вдруг одну из компонент вектора $x[N_k]$ в отрицательную область.
Выбор $\theta = min(x_k[i]/u_k[i]), u_k[i] > 0$ в обычном Симплекс-Методе соответствует решению (**) из п.3.1, потому что $x_k[N_k] = b[M]$ по построению, а $u_k[i] > 0$ только на коэфиициентах $A[M, j_k]$.
Если не существует $u_k[i] > 0$, то задача не ограничена. Это соответствует аналогичному выводу табличного метода, поскольку в таком случае $A[M, j_k] \leq 0$, остальные же компоненты - это нули и минус единица.
Процедура pivot обновляет вектор $b_k[M]$ таким образом, что $b_{k+1}[M] = b_k[M] - \theta * A[M, j_k]$, что эквивалентно (*). Вспомним, что по построению $x_k[N_k] = b[M]$. Это доказывает эквивалентность табличного и обычного Симплекс-Методов.

6 Выводы

В работе были проведены исследования по сравнению двух методов нахождения оптимального решения задачи линейного программирования. (Метод крайних точек и Симплекс метод) Исходя из полученных результатов, можно сделать следующие выводы:

1. Для матриц маленькой размерности (≤ 5) оба метода дают решение за примерно одинаковое время.
2. Для матриц большей размерности Симплекс метод существенно начинает выигрывать по скорости у метода перебора крайних точек . "Существенность"выигрыша растет при росте размерности матриц.
3. Время работы Симплекс метода прямо пропорционально размерности матрицы. Выбросы обусловлены "случайностью"входных данных. (Можем очень быстро получить целевую функцию с положительными коэффициентами)

7 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=HUwStWS0yzrW9SaXn8POZtaz3gTyMTm` `dl=U9ivclLJBeeYQbs3MMhGtwYZ7Mx4nGJelTv0Hv56E4z` / [Электронный ресурс]. Режим доступа: (Дата обращения: 17.02.2023)

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: 20.02.2023)

3. Моисеев Н.Н. "Методы оптимизации"

   URL: `https://avidreaders.ru/book/metody-optimizacii-1.html`/ [Электронный ресурс]. Режим доступа: (Дата обращения: 22.02.2023)