Санкт-Петербургский политехнический университет
Петра Великого

Физико-Механический Институт

Высшая школа прикладной математики и вычислительной физики
Кафедра «Прикладной Математики и Информатики»

ОТЧЁТ ПО ЛАБОРАТОРНЫМ РАБОТАМ ПО ДИСЦИПЛИНЕ «МЕТОДЫ ОПТИМИЗАЦИИ» «РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ СИМПЛЕКС-МЕТОДОМ»

Выполнили
студенты группы 5030102/00101

КУАССИ СИЕМО Т. Г.

Руководитель
к. ф.-м. н., доц.

Родионова Елена Александровна

Санкт-Петербург
2023

Содержание

1 Постановка задачи

Поставлена задача линейного программирования, состоящая из три переменных, включающая ограничения в виде равенств и неравенств.

$$\begin{cases} x_1 + 2x_2 + 3x_3 \geq 1 \\ 2x_1 + 3x_2 + x_3 = 2 \\ 3x_1 + x_2 + 4x_3 \leq 3 \\ x_1 \geq x_2 \geq x_3 \\ 2x_1 + x_2 + 3x_3 = min \end{cases} \tag{1}$$

Функция цели:

$$F(x) = 2x_1 + x_2 + 3x_3 \longrightarrow min \tag{2}$$

1. Привести задачу к виду, необходимому для применения симплекс-метода.
2. Построить к данной задаче двойственную и также привести к виду, необходимому для применения симплекс-метода.
3. Автоматизировать привидение исходной задачи к каноническому виду.
4. Решить обе задачи симплекс-методом
5. Решить обе задачи методом перебора крайних точек.

Симплекс-метод является классическим методом решения задач линейного программирования, который на практике зачастую бывает очень быстрым.

2 Исследование применимости метода

Алгоритм **симплекс-метода** применим к задачам линейного программирования на нахождение минимума. Метод 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.

Проверим применимость **симплекс-метода** к нашей выбранной задаче. Для вычисления ранга приведем матрицу к ступенчатому виду, используя элементарные преобразования над строками и столбцами матрицы:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 4 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & -5 \\ 0 & -5 & -5 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 0 & -5 & -5 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 20 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

от 2 строки отнимаем 1 строку, умноженную на 2; от 3 строки отнимаем 1 строку,

умноженную на 3 2-ую строку делим на -1 2-ую строку делим на -1 3-ую строку делим на 20 3-ую строку делим на 20 Так как ненулевых строк 3, то $rang(A) = 3$, столько же, сколько строк в матрице.
Можно сделать вывод, что симплекс-метод применим к нашей задаче.

3 Описание алгоритма

3.1 Алгоритм перевода из общей в каноническую форму

Вход: система уравнений

1. Проверяем знаки в системе
2. Если «$\le$», то к левой части добавляем $w[i]$, если «$\ge$», то из левой части вычитаем $w[i]$, $w[i] \ge 0$.
3. Знаки неравенства в системе заменяем на равенство.
4. Производим замену переменных:
   если $x[i] \le 0$, то $x'[i] = -x[i] \ge 0$;
   если $x[i]$ любого знака, то $x[i] = u[i] - v[i]$, $v[i], u[i] \ge 0$.

3.2 Алгоритм построения двойственной задачи

Рассмотрим задачу минимума:

$$\begin{array}{l} (x[N], c[N]) \longrightarrow \min_{x[N]}, x[N] \in S, x[N] \ge 0 \\ S := \{x[N] | A[M, N] \cdot x[N] \gtreqless b[M]\}, x[N] \ge 0 \end{array} \qquad (4)$$

Если же перед нами стоит задача максимума, то домножим вектор коэффициентов матрицы цели на -1.

1. Транспонируем заданную матрицу A^T
2. Новый вектор коэффицентов, стоящий в системе справа, равен вектору коэффициентов функции цели (2).
3. Новый вектор коэффициентов функции цели равен вектору коэффицентов, стоящему в системе (1) справа.
4. Если ограничение на $x[i] \ge 0$, то i-ая строка новой системы имеет знак «$\le$». Если нет ограничения на знак, то i-ая строка новой системы имеет знак «$=$».
5. Если ограничение i-ой строки в исходной системе «$\ge$» (тк рассматриваем задачу минимума), то ограничение на знак новой переменной $y[i] \ge 0$. Если ограничение i-ой строки в исходной системе «$=$», то $y[i]$ любого знака.
6. Если исходная задача на поиск минимума, то двойственная на поиск максимума.

3.3 Алгоритм симплекс-метода

Рассмотрим алгоритм в качестве псевдокода:

Algorithm 1: Симплекс-метод решения задачи линейного программирования

Data: задача линейного программирования в стандартной форме

Result: n-мерный вектор $\bar{x} = (\bar{x}_j)$, который является оптимальным решением задачи линейного программирования

```
1  Simplex(A, b, c) :
2  (N, B, A, b, c, v) = Initialize − Simplex(A, b, c)
3   Пусть Δ - новый вектор длиной m
4  while c_j > 0 для некоторого индекса j ∈ N do
5      Выбрать индекс e ∈ B, для которого c_e > 0
6      for каждого индекса i ∈ B do
7          if a_ie > 0 then
8              Δ_i = b_i/a_ie
9          else
10             Δ_i = ∞
11         end
12     end
13     Выбрать индекс l ∈ B, который минимизирует Δ_l
14     if Δ_l == ∞ then
15         return задача неограничена
16     else
17         (N, B, A, b, c, v) = Pivot(N, B, A, b, c, v, l, e)
18     end
19     for i = 1 to n do
20         if i ∈ B then
21             x̄_i = b_i
22         else
23             x̄_i = 0
24         end
25     end
26     return (x̄_i, x̄_2, ..., x̄_n)
27 end
```

Процедура **Simplex** работает следующим образом:

- В строке 2 выполняется процедура $Initialize - Simplex(A, b, c)$, которая или определяет, что предложенная задача неразрешима, или возвращает каноническую форму, базисное решение которой является допустимым.

  Если в системе имеется единичная матрица, то в качестве начальных базисных переменных принимают те компоненты, которым соответствует её столбцам

Algorithm 2: Поиск начального базисного допустимого решения задачи линейного программирования L, заданной в стандартной форме

Data: задача линейного программирования в стандартной форме

Result: или определяет, что предложенная задача неразрешима, или возвращает каноническую форму, базисное решение которой является допустимым

$Initialize - Simplex(A, b, c)$:

Пусть k - является индексом минимального b_i

if $b_k \geq 0$ *// Допустимо ли начальное базисное решение?* **then**

  return $(\{1, 2, ..., n\}, \{n+1, n+2, ..., n+m\}, A, b, c, 0)$

end

Образуем L_{aux} путем добавления — x_0 к левой части каждого ограничения и задаем целевую функцию — x_0

Пусть (N, B, A, b, c, v) представляет собой результирующую каноническую форму для L_{aux}

$l = n + k$

// L_{aux} имеет $n + 1$ небазисную и m базисных переменных

$(N, B, A, b, c, v) = Pivot(N, B, A, b, c, v, l, 0)$

// Базисное решение является допустимым для L_{aux}

Выполняем итерации цикла while в строках 3-12 процедуры $Simplex$, пока не будет найдено оптимальное решение задачи L_{aux}

if *в оптимальном решении* $L_{aux}\bar{x}_0 = 0$ **then**

  if $\bar{x}_0$ *является базисной переменной* **then**

    выполнить одно (вырожденное) замещение, чтобы сделать её небазисной

    В окончательной канонической форме для L_{aux} удалить из ограничений $_0$ и восстановить исходную целевую функцию L, но заменить в этой целевой функции каждую базисную переменную правой частью связанного с ней ограничения

    return полученную окончательную каноническую форму

  end

else

  return “задача неразрешима”

end

- Главная часть алгоритма содержится в цикле while в строках $4 - 16$.

  Если все коэффициенты целевой функции отрицательны, цикл while завершается. В противном случае в строке 5 мы выбираем в качестве вводимой переменной некоторую переменную x_e, коэффициент при которой в целевой функции положителен.

- Затем, в строках $6 - 12$, выполняется проверка каждого ограничения и выбирается то, которое более всего лимитирует величину увеличения x_e.

  Базисная переменная, связанная с этим ограничением, выбирается в качестве выводимой переменной x_i.

- Если ни одно из ограничений не лимитирует возможность увеличения вводимой переменной, алгоритм выдает сообщение "задача неограниченная" (строка 15).

  В противном случае в строке 17 роли вводимой и выводимой переменных меняются путем вызова описанной выше процедуры $Pivot(N, B, A, b, c, v, l, e)$

- В строках 19−25 вычисляется решение $(\bar{x}_i, \bar{x}_2, ..., \bar{x}_n)$ исходной задачи линейного программирования путем присваивания
  небазисным переменным нулевого значения,
  базисным переменным $\bar{x}_i$ — соответствующих значений b_i,
  а строка 26 возвращает эти значения.

- Существует Лемма:

  Если процедура Simplex не завершается не более чем за $\begin{pmatrix} n+m \\ m \end{pmatrix}$ итераций,она зацикливается.

  Использование в программе данной леммы, позволяет избежать зацикливания алгоритма.

Симплекс - метод позволяет эффективно найти оптимальное решение, избегая простой перебор всех возможных угловых точек. Основной принцип метода: вычисления начинаются с какого-то «стартового» базисного решения, а затем ведется поиск решений, «улучшающих» значение целевой функции.

обзор Алгоритм метода (симплекс-таблица):

0. Привести задачу к канонической форме.

1. Составить симплекс-таблицу для начальной допустимой точки.

2. С помощью допустимых операций в строке ЦФ выразить все базисные переменные через внебазисные.

3. Выбрать ведущий столбец с наименьшим отрицательным коэффициентом ЦФ. Соответствующая столбцу внебазисная переменная будет включена в базис.

4. Свободный член каждой строки делится на элемент, стоящий в столбце включаемой переменной. Если элемент неположительный, строка пропускается.

5. Выбирается ведущая строка, у которой найденное отношение положительное и наименьшее. Соответствующая строке базисная переменная будет исключена из базиса. Элемент, стоящий на пересечении ведущей строки и столбца, называется разрешающим.

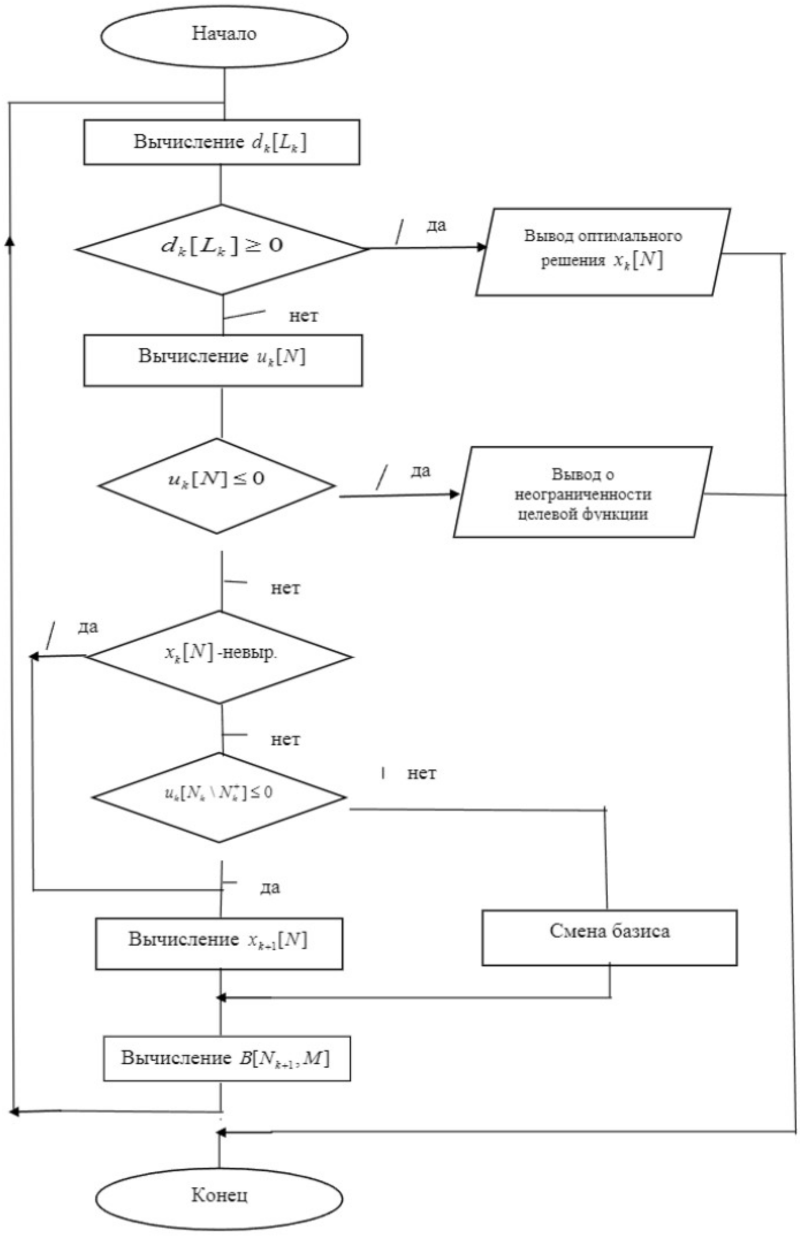
6. Внести изменение в симплекс-таблицу по правилу прямоугольника: ведущий элемент заменяется на себя в степени -1, элементы на ведущей строке делятся на ведущий элемент, элементы на ведущем столбце делятся на ведущий элемент с противоположным знаком, из остальных элементов таблицы вычитается произведение элементов ведущего столбца и ведущей строки соответствующих координат, делённое на разрешающий элемент.

7. Шаги 3-6 повторяются, пока удается найти отрицательный коэффициент ЦФ.

8. Найденное решение оптимально.

Обычно симплекс-таблицы используют для наглядности. В нашем же случае удобнее представить алгоритм в несколько ином виде.

Алгоритм метода (блок-схема):



3.4 Алгоритм перебора опорных векторов

Опорные векторы можно искать прямо по определению, перебирая все возможные базисы и находя соответствующие ненулевые коэффициенты из решения СЛАУ.

Algorithm 3: Метод перебора опорных векторов решения задачи линейного программирования в канонической форме

Data: $A[M, N], b[M], c[N]$ – параметры задачи линейного программирования, поставленной в канонической форме $(m = |M|, n = |N|)$

Result: опорный вектор $x_*[N]$, минимизирующий целевую функцию $(x[N], c[N])$

$V := \emptyset$ – будущий список опорных векторов;
for i *в диапазоне* $\{0; C_m^n\}$ **do**
    $A[M, N_k] := \text{extractMatrix}(i)$;
    if $|det(A[M, N_k])| > eps$ **then**
        $x[N_k] := \text{inv}(A[M, N_k], b[M])$;
        Дополняем нулями до $x[N]$;
        Добавляем $x[N]$ в V;
    end
end
Выбираем x_* – любой вектор из V;
for $v \in V$ **do**
    if $(v, c) < (x_*, c)$ **then**
        $x_* := v$;
    end
end

Метод перебора крайних точек заключается в следующем:

- Рассматривается матрица $A[M, N]$, где число строк матрицы меньше, чем число столбцов $(M < N)$.
- Генерируются квадратные матрицы, выделяемые из матрицы $A[M, N]$, таких матриц получится C_M^N.
- Для каждой такой квадратной матрицы проверяется, что определитель отличен от нуля $|det(A[M, N_k])| > eps$ Если это не так, то эта матрица к рассмотрению не принимается, иначе решается соответсвенно система $A[M, N_k]x[N] = b[M]$ и находится решение.
- Если оказывается, что все компоненты решения удовлетворяют неравенству ≥ 0, то эта точка является полученной частью компонент крайней точки. Для получения крайней точки мы просто пополняем полученное решение нулевыми значениями соответсвующих компонент.
- Находим значение функции цели в крайней точке и запоминаем его.
- Генерируем следующую матрицу и продолжаем вышеперечисленные шаги.
- Сравниваем сохраненные значения между собой и выбираем то решение, которое соответсвует меньшему значению функции цели.

Обзор Алгоритм Метод перебора крайних точек

0. Привести задачу к канонической форме.

1. Получаем матрицу коэффициентов ограничений А размерности $m \times n$, где $n \geq m$.

2. Из получившейся матрицы выбираем всевозможные квадратные матрицы размерности $m \times m$ с сохранением порядка столбцов.

3. Решаем все СЛАУ с полученными матрицами и столбцом b.

4. Из всех полученных решений выбираем те, где все коэффициенты неотрицательны.

5. Подставляем каждое получившееся решению в целевую функцию.

6. Решение, при котором функция будет иметь максимальное значение и будет ответом.

3.5 Про выбор алгоритма решения СЛАУ

Для решения систем линейных алгебраических уравнений в методе перебора крайних точек был выбрал метод Гаусса по дальнейшим причинам:

1. Отсутствие ошибки вычислений.

2. Понятность и простота реализации.

3. Из-за того, что поставленная задача имеет малую размерность мы можем не брать во внимание факт большого времени работы алгоритма.

4 Практическое решение задач

4.1 Результат нахождения задачи двойственной к заданной

Найдём двойствунную задачу для прямой задачи (1):

$$\begin{cases} x_1 + 2x_2 + 3x_3 \geq 1 \\ 2x_1 + 3x_2 + x_3 = 2 \\ 3x_1 + x_2 + 4x_3 \leq 3 \\ x_1 \geq x_2 \geq x_3 \end{cases} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3x_3 \leq 2 \\ 2x_1 + 3x_2 + 1x_3 \leq 1 \\ 3x_1 + x_2 + 4x_3 = 1 \\ x_1 \geq 0; x_3 \leq 0; \end{cases} \tag{5}$$

Функция цели:

$$F(x) = x_1 + 2x_2 + 3x_3 \longrightarrow \max$$

4.2 Результат приведения задач линейного программирования к каноническому виду

- Приведём задачу (1) к каноническому виду:

$$\begin{cases} x_1 + 2x_2 + 3x_3 \geq 1 \\ 2x_1 + 3x_2 + x_3 = 2 \\ 3x_1 + x_2 + 4x_3 \leq 3 \\ x_1 \geq x_2 \geq x_3 \end{cases} \Longrightarrow \begin{cases} x_1 + 2x_2 + 3x_3 - 3x_4 - x_5 = 1 \\ 2x_1 + 3x_2 + 1x_3 - x_4 = 2 \\ 3x_1 + x_2 + 4x_3 - 4x_4 + x_6 = 3 \\ x_1 >= 0; x_2 >= 0; x_3 >= 0; x_4 >= 0; x_5 >= 0; x_6 >= 0; \end{cases} \tag{6}$$

Функция цели:

$$F(x) = 2x_1 + 1x_2 + 1x_3 - x_4 \longrightarrow min$$

- Приведём двойственную задачу (5) к каноническому виду:

$$\begin{cases} x_1 + 2x_2 + 3x_3 \leq 2 \\ 2x_1 + 3x_2 + 1x_3 \leq 1 \\ 3x_1 + x_2 + 4x_3 = 1 \\ x_1 \geq 0; x_3 \leq 0; \end{cases} \Longrightarrow$$

$$\begin{cases} x_1 + 2x_2 - 2x_3 - 3x_4 + x_5 = 2 \\ 2x_1 + 3x_2 - 3x_3 - x_4 + x_6 = 1 \\ 3x_1 + x_2 - x_3 - 4x_4 = 1 \\ x_1 \geq 0; x_2 \geq 0; x_3 \geq 0; x_4 \geq 0; x_5 \geq 0; x_6 \geq 0; \end{cases} \tag{7}$$

Функция цели:

$$F(x) = -x_1 - 2x_2 + 2x_3 + 3x_4 \longrightarrow \max$$

4.3 Результат решения прямой и двойственной задач линейного программирования

- Решение прямой задачи методом перебора крайних точек:

$$x^* = (0, 0.714286, -0.142857)$$

целевая функция в точке решения:

$$f(x^*) = 0.571429$$

- Решение двойственной задачи методом перебора крайних точек:

$$x^* = (0.285714, 0.142857, 0)$$

целевая функция в точке решения:

$$f(x^*) = 0.571429$$

- Решение прямой задачи симплекс - методом:

$$x^* = (0, 0.714286, -0.142857)$$

целевая функция в точке решения:

$$f(x^*) = 0.571429$$

- Решение двойственной задачи симплекс - методом:

$$x^* = (0.285714, 0.142857, 0)$$

целевая функция в точке решения:

$$f(x^*) = 0.571429$$

Можно заметить, что мы поличили одинаковый ответ, решая разными методами. Что указывает на корректность запрограммированного алгоритма и вычислений.

5 Обоснование результатов

Теорема
Чтобы вектор $X_*[N]$ был решением исходной задачи в канонической форме, необходимо и достаточно, чтобы существовал положительный вектор $Y_*[M]$, являющийся решением двойственной задачи и удовлетворяющий следующим условиям:

$$Y_*[M_1] \geq 0 \tag{8}$$

$$C^T[N_1] - Y_*^T[M]A[M, N_1] \geq 0 \tag{9}$$

$$C^T[N_2] - Y_*^T[M]A[M, N_2] = 0 \tag{10}$$

$$Y_*^T[M_1](A[M_1]X_*[N] - b[M_1] = 0 \tag{11}$$

$$(C^T[N_1] - Y_*^T[M]A[M, N_1]) * X_*[N_1] = 0) \tag{12}$$

Проверим полученные результаты:

Подставив полученные решения прямой

$$x^* = (0, 0.714286, -0.142857)$$

и двойственной

$$y^* = (0.285714, 0.142857, 0)$$

задачи в уравнения, левые части обоих условий обнуляются, а значит x^* и y^* являются оптимальными решениями соответствующих задач.

6 Выводы

Симплекс метод был предложен американским математиком Р.Данцигом в 1947 году, с тех пор не утратил свою актуальность, для нужд промышленности этим методом нередко решаются задачи линейного программирования с тысячами переменных и ограничений.

Основные преимущества метода:

- Симплекс-метод является универсальным методом, которым можно решить любую задачу линейного программирования, в то время, как графический метод пригоден лишь для системы ограничений с двумя переменными.
- Решение будет гарантировано найдено за $O(2^n)$ операций, где n - это количество переменных.
- Не так хорош для больших задач, но есть множетсво улучшений базового симплекс-метода, которые компенсируют эту проблему.

Метод перебора - простейший из методов поиска значений действительно-значных функций по какому-либо из критериев сравнения (на максимум, на минимум, на определённую константу). Применительно к экстремальным задачам является примером прямого метода условной одномерной пассивной оптимизации.

Основные преимущества метода:

- Достаточно прост в реализации.
- Показывает отличные результаты с определенной точностью. Не уступает симплекс-методу.

Но есть и занчительный недостаток:

- Данный метод не является удачным для решения объемных задач. Перебор больших матриц будет рассматривать слишком много комбинации, что приведет к значительному замедлению процесса решения задачи.

Список литературы

[1] Кормен, Томас Х. и др. Алгоритмы: построение и анализ, 3-е изд. : Пер. с англ. — Москва : ООО “И. Д. Вильямс”, 2013. — 1328 с. : ил. — Парал. тит. англ.