Санкт-Петербургский политехнический университет
Петра Великого

Институт прикладной математики и механики
Кафедра «Прикладная математика»

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

Выполнили
студенты группы 3630102/80201

Деркаченко А. О.
Хрипунков Д. В.
Войнова А. Н.

Руководитель
к. ф.-м. н., доц.

Родионова Елена Александровна

Санкт-Петербург
2021

1 Постановка задачи

Поставлена задача линейного программирования, состоящая из пяти переменных, включающая три равенства и два неравенства разных знаков. На знаки для четырёх переменных поставлены ограничения:

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 \geq 1 \\ 2x_1 + 3x_2 + 8x_3 + x_5 = 2 \\ x_1 + 4x_2 + 5x_4 + x_5 = 3 \\ 3x_1 + 7x_2 + 4x_3 + 2x_5 = 4 \\ 2x_1 + 3x_2 + 5x_3 + 6x_4 + x_5 \leq 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases} \tag{1}$$

Функция цели:

$$F(x) = 4x_1 + 3x_2 + 2x_3 \longrightarrow min \tag{2}$$

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 с выбором начального приближения методом искусственного базиса.
5. Решить обе задачи методом перебора крайних точек.

Симплекс-метод является классическим методом решения задач линейного программирования, который на практике зачастую бывает замечательно 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:

$$\begin{pmatrix}1&2&3&4&0\\2&3&8&0&1\\1&4&0&5&1\\3&7&4&0&2\\2&3&5&6&1\end{pmatrix} \Longrightarrow \begin{pmatrix}1&2&3&4&0\\0&-1&2&-8&1\\0&2&-3&1&1\\0&1&-5&-12&2\\0&-1&-1&-2&1\end{pmatrix} \Longrightarrow \begin{pmatrix}1&2&3&4&0\\0&1&-2&8&-1\\0&0&1&-15&3\\0&0&-3&-20&3\\0&0&-3&6&0\end{pmatrix} \Longrightarrow$$
$$\begin{pmatrix}1&2&3&4&0\\0&1&-2&8&-1\\0&0&1&-15&3\\0&0&0&-65&12\\0&0&0&-39&9\end{pmatrix} \Longrightarrow \begin{pmatrix}1&2&3&4&0\\0&1&-2&8&-1\\0&0&1&-15&3\\0&0&0&1&-\frac{12}{65}\\0&0&0&0&-\frac{9}{39}\end{pmatrix} \tag{3}$$

Так как ненулевых строк 5, то $rang(A) = 5$, 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 «$\leq$», 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 $x[i] = u[i] - v[i]$, $v[i], u[i] \geq 0$.

3.2 Алгоритм построения двойственной задачи

Рассмотрим задачу минимума:

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

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 $x[i] \geq 0$, то i-ая строка новой системы имеет знак «$\leq$». Если нет ограничения на знак, то i-ая строка новой системы имеет знак «$=$».

5. Если ограничение i-ой строки в исходной системе «$\geq$» (тк рассматриваем задачу минимума), то ограничение на знак новой переменной $y[i] \geq 0$. 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.
- 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.

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 2: 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 тполучения крайней точки мы просто пополняем полученное решение нулевыми значениями соответсвующих компонент.
- Находим значение функции цели в крайней точке и запоминаем его.
- Генерируем следущую матрицу и продолжаем вышеперечисленные шаги.
- Сравниваем сохраненные значения между собой и выбираем то решение, которое соответсвует меньшему значению функции цели.

4 Практическое решение задач

4.1 Результат нахождения задачи двойственной к заданной

Найдём двойствунную задачу для прямой задачи (1):

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 \geq 1 \\ 2x_1 + 3x_2 + 8x_3 + x_5 = 2 \\ x_1 + 4x_2 + 5x_4 + x_5 = 3 \\ 3x_1 + 7x_2 + 4x_3 + 2x_5 = 4 \\ 2x_1 + 3x_2 + 5x_3 + 6x_4 + x_5 \leq 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases} \Longrightarrow \begin{cases} x_1 + 2x_2 + 1x_3 + 3x_4 + 2x_5 \leq 4 \\ 2x_1 + 3x_2 + 4x_3 + 7x_4 + 3x_8 \leq 3 \\ 3x_1 + 8x_2 + 4x_3 + 5x_5 \leq 2 \\ 4x_1 + 5x_3 + 6x_5 \leq 0 \\ x_2 + x_3 + 2x_4 + x_5 \leq 0 \\ x_1 \geq 0, x_5 = 0 \end{cases} \tag{5}$$

Функция цели:

$$F(x) = x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 \longrightarrow \max$$

4.2 Результат приведения задач линейного программирования к каноническому виду

- Приведём задачу (1) к каноническому виду:

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 \geq 1 \\ 2x_1 + 3x_2 + 8x_3 + x_5 = 2 \\ x_1 + 4x_2 + 5x_4 + x_5 = 3 \\ 3x_1 + 7x_2 + 4x_3 + 2x_5 = 4 \\ 2x_1 + 3x_2 + 5x_3 + 6x_4 + x_5 \leq 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases} \Longrightarrow \begin{cases} -x_1 - 2x_2 - 3x_3 - 4x_4 \leq -1 \\ 2x_1 + 3x_2 + 8x_3 + x_5 = 2 \\ x_1 + 4x_2 + 5x_4 + x_5 = 3 \\ 3x_1 + 7x_2 + 4x_3 + 2x_5 = 4 \\ 2x_1 + 3x_2 + 5x_3 + 6x_4 + x_5 \leq 5 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases} \tag{6}$$

Функция цели:

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

- Приведём двойственную задачу (5) к каноническому виду:

$$\begin{cases} x_1 + 2x_2 + x_3 + 3x_4 + 2x_5 \leq 4 \\ 2x_1 + 3x_2 + 4x_3 + 7x_4 + 3x_8 \leq 3 \\ 3x_1 + 8x_2 + 4x_3 + 5x_5 \leq 2 \\ 4x_1 + 5x_3 + 6x_5 \leq 1 \\ x_2 + x_3 + 2x_4 + x_5 = 0 \\ x_1 \geq 0, x_5 \leq 0 \end{cases} \Longrightarrow$$

$$\begin{cases} x_1 + x_2 + 2x_3 - 2x_4 - 3x_5 - 3x_6 + 2x_7 - 2x_8 + x_9 = 4 \\ 2x_1 + 4x_2 + 3x_3 - 3x_4 - 7x_5 - 7x_6 + 3x_7 - 3x_8 + x_{10} = 3 \\ 3x_1 + 5x_3 - 8x_4 - 4x_5 - 4x_6 + 8x_7 - 8x_8 + x_{11} = 2 \\ 4x_1 + 5x_2 + 6x_3 - x + 12 = 1 \\ x_2 + x_3 - x_4 - 2x_5 - 2x_6 + 1x_7 - x_8 = 0 \\ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 = 0 \end{cases} \tag{7}$$

Функция цели:

$$F(x) = x_1 + 3x_2 + 5x_3 - 2x_4 - 4x_5 - 4x_6 + 2x_7 - 2x_8 - 2x_9 \longrightarrow \max$$

4.3 Результат решения прямой и двойственной задач линейного программирования

- Решение прямой задачи методом перебора крайних точек:

$$x^* = (0, 0.10084403, 0.00084034, 0.19327731, 1.63025211, 0, 0, 1.86554621)$$

- Решение двойственной задачи методом перебора крайних точек:

$$x^* = (0, 0, 0, 0, 0, 0, 0, 0, 4, 3, 2, 0)$$

- Решение прямой задачи симплекс - методом:

$$x^* = (0, 0.10084403, 0.00084034, 0.19327731, 1.63025211, 0, 0, 1.86554621)$$

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:

$$C^T[N] - Y_*^T[M]A[M,N] \geq 0$$

$$(C^T[N] - Y_*^T[M]A[M,N]) * X_*[N] = 0)$$

Проверим полученные результаты:

6 Выводы

Симплекс метод был предложен американским математиком Р.Данцигом в 1947 году, с тех пор не утратил свою акиуальность, 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.

7 Приложения

URL: Выполненная лабораторная работа на GitHub
https://github.com/ThinkingFrog/OptimizationMethods/tree/main/Simplex