Симплекс-метод—алгоритм решения оптимизационной задачи линейного программирования путём перебора вершин выпуклого многогранника в многомерном пространстве.

Сущность метода: построение базисных решений, на которых монотонно убывает линейный функционал, до ситуации, когда выполняются необходимые условия локальной оптимальности.

В работе Л. В. Канторовича "Математические методы организации и планирования производства" (1939 г.) были впервые изложены принципы новой отрасли математики, которая позднее получила название линейного программирования.

Исторически общая задача линейного программирования была впервые поставлена в 1947 г. Дж. Б. Данцигом, Маршаллом Вудом и их сотрудниками в департаменте военно-воздушных сил США. В то время эта группа занималась исследованием возможности использования математических и смежных с ними методов для военных задач и проблем планирования. В дальнейшем для развития этих идей в ВВС была организована исследовательская группа под названием Project SCOOP. Первое успешное решение задачи линейного программирования на ЭВМ SEAC было проведено в январе 1952 г.

Данный метод является методом целенаправленного перебора опорных решений задачи линейного программирования. Он позволяет за конечное число шагов либо найти оптимальное решение, либо установить, что оптимальное решение отсутствует.

Основное содержание симплексного метода заключается в следующем:

1. Указать способ нахождения оптимального опорного решения
2. Указать способ перехода от одного опорного решения к другому, на котором значение целевой функции будет ближе к оптимальному, т.е. указать способ улучшения опорного решения
3. Задать критерии, которые позволяют своевременно прекратить перебор опорных решений на оптимальном решении или следать заключение об отсутствии оптимального решения.

Алгоритм симплексного метода решения задач линейного программирования

Для того, чтобы решить задачу симплексным методом необходимо выполнить следующее:

1. Привести задачу к каноническому виду
2. Найти начальное опорное решение с "единичным базисом" (если опорное решение отсутствует, то задача не имеет решение ввиду несовместимости системы ограничений)
3. Вычислить оценки разложений векторов по базису опорного решения и заполнить таблицу симплексного метода

4. Если выполняется признак единственности оптимального решения, то решение задачи заканчивается

5. Если выполняется условие существования множества оптимальных решений, то путем простого перебора находят все оптимальные решения

Пример решения задачи симплексным методом

Пример 26.1

Решить симплексным методом задачу:

$$Z(x) = 9x_1 + 5x_2 + 4x_3 + 3x_4 + 2x_5 \to \max,$$

$$\begin{cases} x_1 - 2x_2 + 2x_3 \le 6, \\ x_1 + 2x_2 + x_3 + x_4 = 24, \\ 2x_1 + x_2 - 4x_3 + x_5 = 30, \end{cases}$$

$$x_j \ge 0 \ \ \forall j.$$

Решение:

Приводим задачу к каноническому виду.

Для этого в левую часть первого ограничения-неравенства вводим дополнительную переменную x_6 с коэффициентом +1. В целевую функцию переменная x_6 входит с коэффицентом ноль (т.е. не входит).

Получаем:

$$Z(x) = 9x_1 + 5x_2 + 4x_3 + 3x_4 + 2x_5 + 0x_6 \to \max,$$

$$\begin{cases} x_1 - 2x_2 + 2x_3 + x_6 = 6, \\ x_1 + 2x_2 + x_3 + x_4 = 24, \\ 2x_1 + x_2 - 4x_3 + x_5 = 30, \end{cases}$$

$$x_j \ge 0 \ \ \forall j.$$

Находим начальное опорное решение. Для этого свободные (неразрешенные) переменные приравниваем к нулю x1 = x2 = x3 = 0.

Получаем **опорное решение** X1 = (0,0,0,24,30,6) с единичным базисом Б1 = (A4, A5, A6).

Вычисляем **оценки разложений векторов** условий по базису опорного решения по формуле:

$$\mathbf{\Delta_k = C_б X_k — c_k}$$

Где:

- $C_б = (c_1, c_2, \dots, c_m)$ — вектор коэффициентов целевой функции при базисных переменных
- $X_k = (x_{1k}, x_{2k}, \dots, x_{mk})$ — вектор разложения соответствующего вектора $A_к$ по базису опорного решения
- $C_к$ — коэффициент целевой функции при переменной $x_к$.

$$\Delta_1 = C_б \cdot X_1 - c_1 = (0,3,2)\cdot(1,1,2) - 9 = 0+3+4-9 = -2;$$
$$\Delta_2 = C_б \cdot X_2 - c_2 = (0,3,2)\cdot(-2,2,1) - 5 = 0+6+2-5 = 3;$$
$$\Delta_3 = C_б \cdot X_3 - c_3 = (0,3,2)\cdot(2,1,-4) - 4 = 0+3-8-4 = -9;$$
$$\Delta_4 = C_б \cdot X_4 - c_4 = (0,3,2)\cdot(0,1,0) - 3 = 0+3+0-3 = 0;$$
$$\Delta_5 = C_б \cdot X_5 - c_5 = (0,3,2)\cdot(0,0,1) - 2 = 0+0+2-2 = 0;$$
$$\Delta_6 = C_б \cdot X_6 - c_6 = (0,3,2)\cdot(1,0,0) - 0 = 0+0+0-0 = 0.$$

Оценки векторов входящих в базис всегда равны нулю. Опорное решение, коэффициенты разложений и оценки разложений векторов условий по базису опорного решения записываются в **симплексную таблицу**:

Таблица 26.1

			9	5	↓4	3	2	0		
	Б	$C_б$	A_0	A_1	A_2	A_3	A_4	A_5	A_6	θ_1 θ_3
←	A_6	0	6	1	−2	[2]	0	0	1	6 3
	A_4	3	24	1	2	1	1	0	0	24 24
	A_5	2	30	2	1	−4	0	1	0	15 –
	Δ_k		132	−2	3	−9	0	0	0	

Сверху над таблицей для удобства вычислений оценок записываются коэффициенты целевой функции. В первом столбце "Б" записываются векторы, входящие в базис опорного решения. Порядок записи этих векторов соответствует номерам разрешенных неизвестных в уравнениях ограничениях. Во втором столбце таблицы "$C_б$" записываются коэффициенты целевой функции при базисных переменных в том же порядке. При правильном расположении коэффициентов целевой функции в столбце "$C_б$" оценки единичных векторов, входящих в базис, всегда равных нулю.

В последней строке таблицы с оценками Δ_k в столбце "A_0" записываются значения целевой функции на опорном решении $Z(X_1)$.

Начальное опорное решение не является оптимальным, так как в задаче на максимум оценки $\Delta_1 = -2$, $\Delta_3 = -9$ для векторов A_1 и A_3 отрицательные.

По теореме об улучшении опорного решения, если в задаче на максимум хотя бы один вектор имеет отрицательную оценку, то можно найти новое опорное решение, на котором значение целевой функции будет больше.

Определим, введение какого из двух векторов приведет к большему приращению целевой функции.

Приращение целевой функции находится по формуле: $\Delta Z_k = -\theta_{0k}\Delta_k$.

Вычисляем значения параметра θ_{01} для первого и третьего столбцов по формуле:

Получаем $\theta_{01} = 6$ при $l = 1$, $\theta_{03} = 3$ при $l = 1$ (таблица 26.1).

Находим приращение целевой функции при введении в базис первого вектора $\Delta Z_1 = — 6*(- 2) = 12$, и третьего вектора $\Delta Z_3 = — 3*(- 9) = 27$.

Следовательно, для более быстрого приближения к оптимальному решению необходимо ввести в базис опорного решения вектор А3 вместо

первого вектора базиса А6, так как минимум параметра θ_{03} достигается в первой строке (l = 1).

Производим преобразование Жордана с элементом X13 = 2, получаем второе опорное решение X2 = (0,0,3,21,42,0) с базисом Б2 = (А3, А4, А5). (таблица 26.2)

Таблица 26.2

	Б	$C_б$	A_0	A_1	A_2	A_3	A_4	A_5	A_6	θ_2
	A_3	4	3	1/2	−1	1	0	0	1/2	–
←	A_4	3	21	1/2	[3]	0	1	0	−1/2	7
	A_5	2	42	4	−3	0	0	1	2	–
	Δ_k		159	5/2	−6	0	0	0	9	

Это решение не является оптимальным, так как вектор А2 имеет отрицательную оценку Δ2 = — 6. Для улучшение решения необходимо ввести вектор А2 в базис опорного решения.

Определяем номер вектора, выводимого из базиса. Для этого вычисляем параметр θ_{02} для второго столбца, он равен 7 при l = 2. Следовательно, из базиса выводим второй вектор базиса А4. Производим преобразование Жордана с элементом x_{22} = 3, получаем третье опорное решение X3 = (0,7,10,0,63,0) Б2 = (А3, А2, А5) (таблица 26.3).

Таблица 26.3

Б	$C_б$	A_0	A_1	A_2	A_3	A_4	A_5	A_6
A_3	4	10	2/3	0	1	1/3	0	1/3
A_2	5	7	1/6	1	0	1/3	0	−1/6
A_5	2	63	9/2	0	0	1	1	3/2
Δ_k		201	7/2	0	0	2	0	7/2

Это решение является единственным оптимальным, так как для всех векторов, не входящих в базис оценки положительные

$\Delta_1 = 7/2$, $\Delta_4 = 2$, $\Delta_6 = 7/2$.

Ответ: max Z(X) = 201 при X = (0,7,10,0,63).

Алгоритм симплекс-метода

Усиленная постановка задачи

Рассмотрим следующую задачу линейного программирования:

$$c^T x \to \max, Ax \leqslant b, x \geqslant 0, b \geqslant 0.$$

Теперь поставим эту задачу в эквивалентной *усиленной* форме.

Необходимо максимизировать *Z*, где:

$$\begin{bmatrix} 1 & -c^T & 0 \\ 0 & A & E \end{bmatrix} \begin{bmatrix} Z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, x, x_s \geqslant 0$$

Здесь x — переменные из исходного линейного функционала, x_s — новые переменные, дополняющие старые таким образом, что неравенство переходит в равенство, c — коэффициенты исходного линейного функционала, Z — переменная, которую необходимо максимизировать. Полупространства $x \geqslant 0$ и $x_s \geqslant 0$ в пересечении образуют многогранник, представляющий множество допустимых решений. Разница между числом переменных и уравнений даёт нам число степеней свободы. Проще говоря, если мы рассматриваем вершину многогранника, то это число рёбер, по которым мы можем продолжать движение. Тогда мы можем присвоить этому числу переменных значение 0 и назвать их *«непростыми»*. Остальные переменные при этом будут вычисляться однозначно и называться *«простыми»*. Полученная точка будет вершиной в пересечении соответствующих непростым переменным гиперплоскостей. Для того, чтобы найти т. н. *начальное допустимое решение* (вершину, из которой мы начнём движение), присвоим всем изначальным переменным x значение 0 и будем их считать непростыми, а все новые будем считать простыми. При этом *начальное допустимое решение* вычисляется однозначно : $\mathbf{x}_{si} = \mathbf{b}_i$.

Алгоритм

Теперь приведём шаги алгоритма. На каждом шаге мы будем менять множества простых и непростых векторов (двигаться по рёбрам), и матрица будет иметь следующий вид:

$$\begin{bmatrix} 1 & c_B^T B^{-1} A - c^T & c_B^T B^{-1} \\ 0 & B^{-1} A & B^{-1} \end{bmatrix} \begin{bmatrix} Z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} c_B^T B^{-1} b \\ B^{-1} b \end{bmatrix},$$

где c_B — коэффициенты вектора c, соответствующие простым переменным (переменным x_s соответствуют 0), B — столбцы $[\mathbf{AE}]$, соответствующие простым переменным. Матрицу, образованную оставшимися столбцами обозначим D. Почему матрица будет иметь такой вид поясним в описании шагов алгоритма.

Первый шаг.

Выбираем начальное допустимое значение, как указано выше. На первом шаге B — единичная матрица, так как простыми переменными являются x_s. c_B — нулевой вектор по тем же причинам.

Второй шаг

Покажем, что в выражении $(c_B^T B^{-1} A - c^T)x + (c_B^T B^{-1})x_s$ только непростые переменные имеют ненулевой коэффициент. Заметим, что из выражения $Ax + x_s = b$ простые переменные однозначно выражаются через непростые, так как число простых переменных равно числу уравнений. Пусть x' — простые, а x'' — непростые переменные на данной итерации. Уравнение $Ax + x_s = b$ можно переписать, как $Bx' + Dx'' = b$. Умножим его на B^{-1} слева: $x' + B^{-1}Dx'' = B^{-1}b$. Таким образом мы выразили простые переменные через непростые, и в выражении $B^{-1}Ax + B^{-1}x_s$, эквивалентному левой части равенства, все простые переменные имеют единичные коэффициенты. Поэтому, если прибавить к равенству $Z - c^T x = 0$ равенство $c_B^T B^{-1} Ax + c_B^T B^{-1} x_s$, то в полученном равенстве все простые переменные будут иметь нулевой коэффициент — все простые переменные вида x сократятся, а простые переменные вида x_s не войдут в выражение $c_B^T B^{-1} x_s$.

Выберем ребро, по которому мы будем перемещаться. Поскольку мы хотим максимизировать Z, то необходимо выбрать переменную, которая будет более всех уменьшать выражение

$$(c_B^T B^{-1} A - c^T)x + (c_B^T B^{-1})x_s.$$

Для этого выберем переменную, которая имеет наибольший по модулю отрицательный коэффициент. Если таких переменных нет, то есть все коэффициенты этого выражения неотрицательны, то мы пришли в искомую вершину и нашли оптимальное решение. В противном случае начнём увеличивать эту непростую переменную, то есть перемещаться по соответствующему ей ребру. Эту переменную назовём *входящей*.

Третий шаг

Теперь необходимо понять, какая простая переменная первой обратится в ноль по мере увеличения входящей переменной. Для этого достаточно рассмотреть систему:

$$\left[B^{-1}AB^{-1}\right]\begin{bmatrix}x\\x_s\end{bmatrix}=B^{-1}b$$

При фиксированных значениях непростых переменных система однозначно разрешима относительно простых, поэтому мы можем определить, какая из простых переменных первой достигнет нуля при увеличении входящей. Эту переменную назовем *выходящей*. Это будет означать, что мы натолкнулись на новую вершину. Теперь входящую и выходящую переменную поменяем местами — входящая «войдёт» в простую, а выходящая из них «выйдет» в непростые. Теперь перепишем матрицу B и вектор c_B в соответствии с новыми наборами простых и непростых переменных, после чего вернёмся ко второму шагу. x''

Поскольку число вершин конечно, то алгоритм однажды закончится. Найденная вершина будет являться оптимальным решением.

Двухфазный симплекс-метод

Причины использования

Если в условии задачи линейного программирования не все ограничения представлены неравенствами типа «≤», то далеко не всегда нулевой вектор будет допустимым решением. Однако каждая итерация симплекс-метода является переходом от одной вершины к другой, и если неизвестно ни одной вершины, алгоритм вообще не может быть начат.

Процесс нахождения исходной вершины не сильно отличается от однофазного симплекс-метода, однако может в итоге оказаться сложнее, чем дальнейшая оптимизация.

Модификация ограничений

Все ограничения задачи модифицируются согласно следующим правилам:

• ограничения типа «≤» переводятся на равенства созданием дополнительной переменной с коэффициентом «+1». Эта модификация проводится и в однофазном симплекс-методе, дополнительные переменные в дальнейшем используются как исходный базис.

• ограничения типа «≥» дополняются одной переменной с коэффициентом «**−1**». Поскольку такая переменная из-за отрицательного коэффициента не может быть использована в исходном базисе, необходимо создать ещё одну, **вспомогательную**, переменную. Вспомогательные переменные всегда создаются с коэффициентом «+1».

• ограничения типа «=» дополняются одной вспомогательной переменной.

Соответственно, будет создано некоторое количество дополнительных и вспомогательных переменных. В исходный базис выбираются дополнительные переменные с коэффициентом «+1» и все вспомогательные.*Осторожно: решение, которому соответствует этот базис, не является допустимым.*

Различия между дополнительными и вспомогательными переменными

Несмотря на то, что и дополнительные, и вспомогательные переменные создаются искусственно и используются для создания исходного базиса, их значения в решении сильно отличаются:

• дополнительные переменные сообщают, насколько соответствующее им ограничение «недоиспользовано». Значение дополнительной переменной, равное нулю, соответствует равенству значений правых и левых частей ограничения.

• вспомогательные переменные сообщают, насколько данное условие далеко от допустимого (относительно конкретного ограничения). Если значение вспомогательной переменной больше нуля, то данное решение не выполняет определённое ограничение, а значит не является допустимым.

То есть ненулевое значение дополнительной переменной может (но не должно) сигнализировать о*неоптимальности* решения. Ненулевое значение вспомогательной переменной сигнализирует о *недопустимости*решения.

Фазы решения

После того, как было модифицировано условие, создаётся **вспомогательная целевая функция**. Если вспомогательные переменные были обозначены, как y_i, $i \in \{1, .., k\}$, то вспомогательную функцию определим, как

$$z' = \sum_{i=1}^{k} y_i \to min$$

.

После этого проводится обыкновенный симплекс-метод относительно вспомогательной целевой функции. Поскольку все вспомогательные переменные увеличивают значение z', в ходе алгоритма они будут поочерёдно выводится из базиса, при этом после каждого перехода новое решение будет всё ближе к множеству допустимых решений.

Когда будет найдено оптимальное значение вспомогательной целевой функции, могут возникнуть две ситуации:

• оптимальное значение z' больше нуля. Это значит, что как минимум одна из вспомогательных переменных осталась в базисе. В таком случае можно сделать вывод, что допустимых решений данной задачи линейного программирования не существует.

• оптимальное значение z' равно нулю. Это означает, что все вспомогательные переменные были выведены из базиса, и текущее решение является допустимым.

Во втором случае мы имеем допустимый базис, или, иначе говоря, исходное допустимое решение. Можно проводить дальнейшую оптимизацию с учётом исходной целевой функции, при этом уже не обращая внимания на вспомогательные переменные. Это и является второй фазой решения.

Модифицированный симплекс-метод

В модифицированном методе матрица

$$\begin{bmatrix} 1 & c_B^T B^{-1} A - c^T & c_B^T B^{-1} \\ 0 & B^{-1} A & B^{-1} \end{bmatrix}$$

не пересчитывается, хранится и пересчитывается только матрица B^{-1}. В остальном алгоритм похож на вышеописанный.

1. Вычисляем двойственные переменные $d = c_B^T B^{-1}$

2. Проверка оптимальности. $c_B^T B^{-1} A - c^T$ преобразуется в $d^T A - c^T$.

Проверка заключается в вычислении $d^T A_n - c_n$ для всех столбцов $n \in N$. Столбец со значением < 0 можно вводить в базис.

Часто выбирают минимальное значение, но для этого нужно перебрать все столбцы.

Чаще выбирают значение, меньшее некоторого заданного значения $-\varepsilon$

Если такого столбца не обнаружится, за ε принимается максимальное найденное абсолютное значение и соответствующий столбец A_J вводится в базис.

3. Определение выводимого.

Пусть A_J - вводимый столбец, соответствующий переменной x_J Базисный план - это решение системы $A_B p = b$Увеличиваем $A_B p + \vartheta A_J = b$.

Умножим слева на B^{-1}, т.е. $B^{-1} A_B p + \vartheta B^{-1} A_J = B^{-1} b$.

Здесь $B^{-1} b$ - базисный план, $q = B^{-1} A_J$ - разложение вводимого столбца по базису.

Находим максимальное значение ϑ, при котором все значения не отрицательны. Если ϑ может быть взято как угодно велико, решение не ограничено. В противном случае один из элементов выйдет на нулевое значение. Выводим соответствующий столбец из базиса.

4. Пересчет опорного(базисного) плана.

Вычисляем новый опорный план по уже приведенной формуле $x = p + \vartheta B^{-1} A_J$ с найденным значением ϑ.

5. Пересчитываем обратную к базисной B^{-1}.

Пусть $B^{-1} A_F$ - выводимый столбец.

Матрица B представима в виде $[B_G A_F]$

где B_G - базисная матрица без выводимого столбца.

После замены столбца базисная матрица будет иметь вид $[B_G A_J]$

Нам нужно найти матрицу B_1, такую что

$$[B_G, A_J]B_1^{-1} = E \Rightarrow [B^{-1}B_G B_1^{-1}, B^{-1}A_J] = B^{-1} \Rightarrow [B^{-1}B_G, q]B_1^{-1} = B^{-1}$$

=>

$$\begin{bmatrix} E & q' \\ 0 & q_f \end{bmatrix} B_1^{-1} = B^{-1}$$

Откуда $$B_1^{-1} = \begin{bmatrix} E & -q'/q_f \\ 0 & 1/q_f \end{bmatrix} B^{-1}$$

Замечание.

При пересчете матрицы B^{-1} накапливаются ошибки округления. Во избежание получения больших ошибок время от времени матрица пересчитывается полностью. Этот процесс называется "повторением".

Мультипликативный вариант симплекс-метода

В мультипликативном варианте матрица B^{-1} не хранится, хранятся лишь множители $\begin{bmatrix} E & -q'/q_f \\ 0 & 1/q_f \end{bmatrix}$

При решении экономических задач часто матрица ограничений разреженная, в таком случае мультипликативный вариант получает дополнительные преимущества - можно хранить мультипликаторы в сжатом виде (не хранить нули).

Другие варианты симплекс-метода

Во избежание накопления ошибок округления может использоваться LU-разложение матрицы.

При подавляющем числе ограничений типа "неравенство" может быть использован **метод переменного базиса**.

Метод основан на том, что базисная матрица может быть представлена в виде

$$\begin{bmatrix} A_B & 0 \\ A_E & E \end{bmatrix}$$

Обратная к ней имеет вид

$$\begin{bmatrix} A_B^{-1} & 0 \\ -A_B^{-1}A_E & E \end{bmatrix}$$

При относительно небольших размерах матрицы A_B^{-1} остальная часть матрицы может не храниться.

Таким подходом удается решить задачи с десятками миллионов строк ограничений (например, из теории игр).

Двойственный симплекс-метод

Для реализации двойственного метода необходимо перейти от задачи на минимум к задаче на максимум (или наоборот) путем транспонирования матрицы коэффициентов. При переходе от задачи на минимум целевая функция примет вид:

$$g = \sum_{i=1}^{n} b_i y_i \to max$$

при ограничениях

$$\sum_{i=1}^{n} a_{ij} y_i <= c_j$$

.

Теорема двойственности. Если из пары двойственных задач одна обладает оптимальным планом, то и другая имеет решение, причем экстремальные значения линейных функций этих задач равны.

Если линейная функция одной из задач не ограничена, то другая не имеет решения.

Вычислительная эффективность

Симплекс-метод удивительно эффективен на практике, но в 1972 Кли и Минти [4] привели пример, в котором симплекс-метод перебирал все вершины симплекса, что показывает экспоненциальную сходимость метода в худшем случае. С тех пор для каждого варианта метода был найден пример, на котором метод вел себя исключительно плохо.

Наблюдения и анализ эффективности метода в практических приложениях привело к развитию других способов измерения эффективности.

Симплекс-метод имеет среднюю полиномиальную сходимость при широком выборе распределения значений в случайных матрицах.

Вычислительная эффективность оценивается обычно при помощи двух параметров:

1) Числа итераций, необходимого для получения решения;

2) Затрат машинного времени.

В результате численных экспериментов получены результаты:

1) Число итераций при решении задач линейного программирования в стандартной форме с M ограничениями и N переменными заключено между M и $3M$. Среднее число итераций $2M$. Верхняя граница числа итераций равна $2M + N$.

2) Требуемое машинное время пропорционально M^3.

Число ограничений больше влияет на вычислительную эффективность, чем число переменных, поэтому при формулировке задач линейного программирования нужно стремиться к уменьшению числа ограничений пусть даже путём роста числа переменных.