

Алгоритмы и вычислительные методы оптимизации

Лекция 2

1.4. Базисные и опорные решения (продолжение)

Базисные решения, в которых все переменные неотрицательны, называются *опорными решениями*.

Опорные решения можно находить путем перебора базисных решений, но можно внести изменения в уже рассмотренный алгоритм поиска базисных решений для более быстрого нахождения опорных решений.

При нахождении опорных решений будем считать, что правая часть исходной системы не содержит отрицательных коэффициентов. Этого можно всегда достичь, умножив нужные уравнения на -1 . Далее при преобразованиях матрицы, необходимо следить за тем, чтобы правая часть сохраняла неотрицательность.

Рассмотрим дополнительные условия, которые будут накладываться на выбор разрешающего элемента при выполнении жордановых исключений.

Пусть a_{ij} – разрешающий элемент j -го столбца, a_{kj} – элемент в j -ом столбце и k -ой строке ($a_{kj} \neq 0$), на месте которого после элементарных преобразований будет получен 0. Рассмотрим один шаг жордановых исключений:

$$\left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & \boxed{a_{ij}} & \dots & b_i \\ \dots & \dots & \dots & \dots \\ \dots & a_{kj} & \dots & b_k \\ \dots & \dots & \dots & \dots \end{array} \right) \sim \left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & 1 & \dots & b'_i \\ \dots & \dots & \dots & \dots \\ \dots & 0 & \dots & b'_k \\ \dots & \dots & \dots & \dots \end{array} \right)$$

$$\text{где } b_i \geq 0, b_k \geq 0, b'_i = \frac{b_i}{a_{ij}}, b'_k = b_k - \frac{b_i \cdot a_{kj}}{a_{ij}} = \frac{b_k \cdot a_{ij} - b_i \cdot a_{kj}}{a_{ij}}.$$

$$\left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & \boxed{a_{ij}} & \dots & b_i \\ \dots & \dots & \dots & \dots \\ \dots & a_{kj} & \dots & b_k \\ \dots & \dots & \dots & \dots \end{array} \right) \sim \left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & 1 & \dots & b'_i \\ \dots & \dots & \dots & \dots \\ \dots & 0 & \dots & b'_k \\ \dots & \dots & \dots & \dots \end{array} \right) \quad \begin{aligned} b'_i &= \frac{b_i}{a_{ij}} \\ b'_k &= b_k - \frac{b_i \cdot a_{kj}}{a_{ij}} = \frac{b_k \cdot a_{ij} - b_i \cdot a_{kj}}{a_{ij}} \end{aligned}$$

Хотим сохранить неотрицательность правой части. Посмотрим, какие условия будут наложены на коэффициенты исходной матрицы

$$1) \ b'_i = \frac{b_i}{a_{ij}} \geq 0, \text{ откуда } a_{ij} > 0$$

Откуда заключаем, что разрешающий элемент должен быть положительным

$$\left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & \boxed{a_{ij}} & \dots & b_i \\ \dots & \dots & \dots & \dots \\ \dots & a_{kj} & \dots & b_k \\ \dots & \dots & \dots & \dots \end{array} \right) \sim \left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & 1 & \dots & b'_i \\ \dots & \dots & \dots & \dots \\ \dots & 0 & \dots & b'_k \\ \dots & \dots & \dots & \dots \end{array} \right) \quad \begin{aligned} b'_i &= \frac{b_i}{a_{ij}} \\ b'_k &= b_k - \frac{b_i \cdot a_{kj}}{a_{ij}} = \frac{b_k \cdot a_{ij} - b_i \cdot a_{kj}}{a_{ij}} \end{aligned}$$

$$2) b'_k = \frac{b_k \cdot a_{ij} - b_i \cdot a_{kj}}{a_{ij}} \geq 0$$

Из предыдущих рассуждений знаменатель положителен, следовательно, должно выполняться:

$$b_k \cdot a_{ij} - b_i \cdot a_{kj} \geq 0$$

$$b_k \cdot a_{ij} - b_i \cdot a_{kj} \geq 0$$

Возможны 2 случая: $a_{kj} < 0$ и $a_{kj} > 0$. Рассмотрим их.

$$a) a_{kj} < 0 \Rightarrow b_k \cdot a_{ij} - b_i \cdot a_{kj} \geq 0 \quad (b_i \geq 0)$$

$$б) a_{kj} > 0$$

$$b_k \cdot a_{ij} \geq b_i \cdot a_{kj} \Rightarrow \frac{b_k}{a_{kj}} \geq \frac{b_i}{a_{ij}} \quad (a_{kj} > 0, a_{ij} > 0)$$

$$\left(\begin{array}{ccc|c} \dots & \dots & \dots & \dots \\ \dots & \boxed{a_{ij}} & \dots & b_i \\ \dots & \dots & \dots & \dots \\ \dots & a_{kj} & \dots & b_k \\ \dots & \dots & \dots & \dots \end{array} \right)$$

Следовательно, при поиске опорных решений на каждом шаге метода Жордана-Гаусса для включения в базис следует выбирать переменную с положительным коэффициентом a_{ij} , для которого в j -ом столбце достигается минимум отношения свободных членов к положительным элементам столбца:

$$\frac{b_i}{a_{ij}} = \min_{k, a_{kj} > 0} \frac{b_k}{a_{kj}}$$

Пример 2

Найти все опорные решения системы.

$$\begin{cases} 2x_1 - x_2 + x_3 - x_4 = 3 \\ 2x_1 - x_2 + x_4 = 2 \\ 3x_1 - x_3 - x_4 = -1 \end{cases}$$

Решение примера 2 – в файле [lecture2.pdf](#).

$X = (2/3; 0; 7/3; 2/3)$ - единственное опорное решение

1.5. Генерирование сочетаний без повторений из n по k

Без ограничения общности можно считать, что $A = \{1, 2, \dots, n\}$.

Каждому k -элементному подмножеству взаимно однозначно соответствует возрастающая последовательность длины k с элементами из A .

Сформулируем алгоритм, который генерирует такие последовательности в лексикографическом порядке.

Пусть p – номер элемента, с которого будут начинаться изменения в следующем сочетании.

1. Первая последовательность $(1, \dots, k)$, $p = k$.
2. Если последний элемент сгенерированной последовательности не равен n , то $p = k$, в новой последовательности меняется только последний элемент.

Если последний элемент сгенерированной последовательности равен n , то уменьшаем p на 1 и новая сгенерированная последовательность $(b_1, \dots, b_p, \dots, b_i, \dots, b_k)$ будет отличаться от предыдущей $(a_1, \dots, a_p, \dots, a_i, \dots, a_k)$, начиная с p -го элемента по следующему правилу: p -ый элемент увеличиваем на 1, а все последующие элементы на 1 больше предыдущего.

Правило можно сформулировать по-другому: $b_i = a_p + (i - p + 1), i = p, \dots, k$.

$$\begin{array}{ll}
 b_p = a_p + 1, & (a_1, \dots, a_p, \dots, a_i, \dots, a_k) \\
 b_{p+1} = a_p + 2, & (b_1, \dots, b_p, \dots, b_i, \dots, b_k) \\
 \dots & \underbrace{\hspace{10em}}_{i-p+1} \\
 b_k = a_p + (k - p + 1) &
 \end{array}$$

Пункт 2 повторяем до тех пор, пока p не станет равным 0.

Пример: Генерирование сочетаний из 5 по 3.

$(1, 2, 3), p = 3;$

$(1, 2, 4), p = 3;$

$(1, 2, 5), p = 2;$

$(1, 3, 4), p = 3;$

$(1, 3, 5), p = 2;$

$(1, 4, 5), p = 1;$

$(2, 3, 4), p = 3;$


$(2, 3, 5), p = 2;$


$(2, 4, 5), p = 1;$

$(3, 4, 5), p = 0.$

1.6. Различные формы записи задачи линейного программирования. Переход от одной формы записи к другой

Рассмотрим задачу линейного программирования (ЗЛП) *в общем виде*:

Целевая функция 
$$Z = \sum_{j=1}^n c_j x_j \rightarrow \max(\min) \quad (1.9)$$

Система ограничений 
$$\left\{ \begin{array}{l} \sum_{j=1}^n a_{ij} x_j \leq b_i \quad (i = 1, \dots, m_1), \\ \sum_{j=1}^n a_{ij} x_j \geq b_i \quad (i = m_1 + 1, \dots, m_2), \\ \sum_{j=1}^n a_{ij} x_j = b_i \quad (i = m_2 + 1, \dots, m), \\ x_j \geq 0 \quad (j = 1, \dots, k, k \leq n). \end{array} \right. \quad (1.10)$$

где a_{ij}, b_i, c_j – заданные постоянные величины и $m_1 \leq m_2 \leq m$.

Задача линейного программирования задана *в канонической форме*, если требуется максимизировать целевую функцию, все ограничения системы – уравнения, и на все переменные наложено условие неотрицательности ($m_1 = m_2 = 0, k = n$).

Задача линейного программирования задана *в симметричной форме*, если требуется максимизировать целевую функцию, все ограничения системы – неравенства « \leq » (или минимизировать целевую функцию, все ограничения системы – неравенства « \geq ») и на все переменные наложено условие неотрицательности.

Набор чисел $X = (x_1, x_2, \dots, x_n)$ называется *допустимым решением (планом)*, если он удовлетворяет системе ограничений (1.10).

Множество всех допустимых решений называется *областью допустимых решений (ОДР)*.

Допустимое решение $X^* = (x_1^*, x_2^*, \dots, x_n^*)$, для которого достигается максимальное (минимальное) значение функции, называется *оптимальным планом*. Значение целевой функции при плане X будем обозначать $Z(X)$.

Следовательно, если X^* – оптимальный план задачи, то для любого X выполняется неравенство $Z(X) \leq Z(X^*)$ при нахождении максимума функции или $Z(X) \geq Z(X^*)$ при нахождении минимума функции.

Термины «план» и «оптимальный план» возникли из экономических приложений.

Все три формы записи ЗЛП являются эквивалентными в том смысле, что имеются алгоритмы перехода от одной формы к другой. Таким образом, если имеется способ решения задачи в одной из форм, то всегда можно определить оптимальный план задачи, заданной в любой другой форме.

Рассмотрим алгоритмы перехода от одной формы к другой.

- **Симметричная \rightarrow каноническая.**

Переход осуществляется путем добавления в левую часть каждого неравенства дополнительной неотрицательной переменной. Если неравенство было « \leq », то дополнительная переменная добавляется в левую часть неравенства со знаком « $+$ ». Если неравенство было « \geq », то дополнительная переменная добавляется в левую часть неравенства со знаком « $-$ ». Вводимые дополнительные переменные называются *балансовыми*. Задачу минимизации функции Z заменяют на задачу максимизации функции $(-Z)$ и используют, что $\min Z = -\max (-Z)$.

- **Каноническая \rightarrow симметричная.**

Для осуществления такого перехода находится общее решение системы уравнений – ограничений, целевая функция выражается через свободные переменные. Далее, воспользовавшись неотрицательностью базисных переменных, можно исключить их из задачи. Симметричная форма задачи будет содержать неравенства, связывающие только свободные переменные, и целевую функцию, зависящую только от свободных переменных. Значения базисных переменных находятся из общего решения исходной системы уравнений.

- **Общая → каноническая.**

Каждая переменная, на которую не было наложено условие неотрицательности, представляется в виде разности двух новых неотрицательных переменных. Неравенства преобразуются в уравнения путем введения в левую часть каждого неравенства балансовой переменной таким же образом, как это было описано при переходе от симметричной к канонической форме. Задачу минимизации функции Z заменяют на задачу максимизации функции $(-Z)$ таким же образом, как это было описано при переходе от симметричной к канонической форме.

Пример 1

Перейти от общей формы записи задачи линейного программирования к канонической форме.

$$Z = 3x_1 - 2x_2 - 3x_3 \rightarrow \min$$

$$\begin{cases} 2x_1 - x_2 + x_3 \geq 2 \\ 3x_1 + 2x_2 - 4x_3 \leq 6 \\ x_1 + x_2 + 5x_3 = 4 \\ x_1, x_2 \geq 0 \end{cases}$$



$$-Z = -3x_1 + 2x_2 + 3x_3' - 3x_4' \rightarrow \max$$

$$\begin{cases} 2x_1 - x_2 + x_3' - x_3'' - x_4 = 2 \\ 3x_1 + 2x_2 - 4x_3' + 4x_3'' + x_5 = 6 \\ x_1 + x_2 + 5x_3' - 5x_3'' = 4 \\ x_1, x_2, x_3', x_3'', x_4, x_5 \geq 0 \end{cases}$$

Решение примера 1 – в файле [lecture2.pdf](#).

Пример 2

Перейти от канонической формы записи задачи линейного программирования к симметричной форме.

$$Z = 2x_1 - x_2 - 3x_3 + 6x_4 - x_5 \rightarrow \max$$

$$\begin{cases} x_1 + 3x_2 + x_3 + 4x_4 - x_5 = 12 \\ x_1 + 2x_2 + 3x_4 - x_5 = 6 \\ 3x_1 + 3x_2 + 16x_4 - 2x_5 = 26 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$



$$Z = 2 + x_2 - 4x_4 \rightarrow \max$$

$$\begin{cases} -x_2 + 10x_4 \leq 14 \\ x_2 + x_4 \leq 6 \\ -3x_2 + 7x_4 \leq 8 \\ x_2, x_4 \geq 0 \end{cases}$$

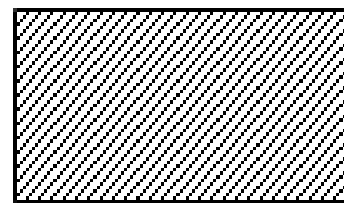
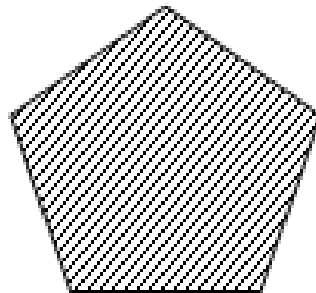
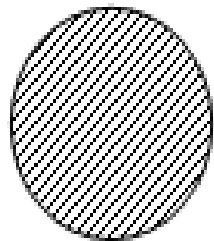
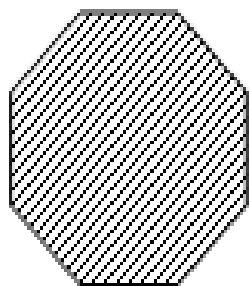
Решение примера 2 – в файле [lecture2.pdf](#).

1.7. Графическое решение задачи линейного программирования

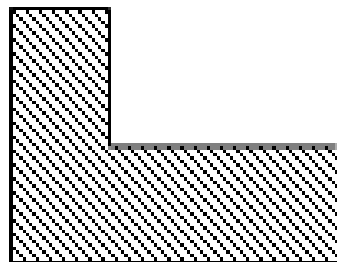
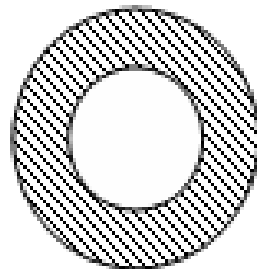
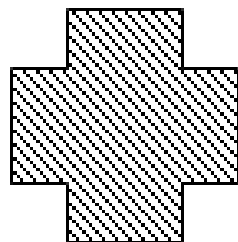
Графический метод решения ЗЛП применяется для решения задач, заданных в симметричной форме. Этот метод наиболее эффективно применяется для решения задач с двумя переменными, т.к. требует графических построений. В случае трех переменных необходимы построения в R^3 , в случае четырех переменных необходимы построения в R^4 и т.д.

Множество точек называется *выпуклым*, если для любых двух точек множества оно содержит отрезок, их соединяющий.

Примеры выпуклых множеств



Примеры множеств, которые не являются выпуклыми.



Теорема 4 Пересечение любого количества выпуклых множеств является выпуклым множеством.

Теорема 5 Пусть имеются две произвольные точки $P(p_1, p_2, \dots, p_n)$ и $Q(q_1, q_2, \dots, q_n)$ в пространстве R^n . Тогда для любой точки $X(x_1, x_2, \dots, x_n)$ отрезка $[PQ]$ должно выполняться: $x_i = \lambda q_i + (1 - \lambda) p_i$ ($i = 1, 2, \dots, n$), где $0 \leq \lambda \leq 1$.

Доказательство Теоремы 5 – в файле [lecture2.pdf](#).

Гиперплоскостью в пространстве R^n называется множество точек, удовлетворяющее уравнению $a_1 x_1 + a_2 x_2 + \dots + a_n x_n = b$, где a_i ($i = 1, 2, \dots, n$), b – константы.

В двумерном случае гиперплоскостью является прямая.

Полупространством называется множество точек, удовлетворяющее одному из неравенств $a_1 x_1 + a_2 x_2 + \dots + a_n x_n \leq b$ или $a_1 x_1 + a_2 x_2 + \dots + a_n x_n \geq b$.

Гиперплоскость делит точки пространства на два полупространства. В двумерном случае гиперплоскостью является полуплоскость.