# Тема 1 Целочисленное линейное программирование

Целочисленное линейное программирование (ЦЛП) ориентировано на решение задач линейного программирования (ЛП), в которых на все или часть переменных наложены условия целочисленности. Различают полностью целочисленные задачи и частично целочисленные задачи.

К необходимости рассматривать целочисленные задачи мы приходим по двум причинам:

существует много практических задач, которые изначально по своей природе являются целочисленными; существуют задачи, которые явно не содержат требований целочисленности, но содержат условия, которые трудно или невозможно (в исходной постановке) записать в виде формул (т.е. математически формализовать). Это так называемые «некорректные» задачи. Анализ таких задач, введение новых целочисленных переменных позволяют преодолеть эту трудность и свести задачу к стандартному виду.

Рассмотрим ряд примеров задач второго типа.

## 1.1 Примеры прикладных задач, содержащих условия целочисленности. Постановка задачи целочисленного программирования

### 1.1.1 Задача с постоянными элементами затрат

Во многих типах задач планирования производства рассматривается следующая ситуация. Предприятие производит $N$ видов продукции. Затраты на производство продукции $j$ -го вида складываются из постоянных затрат в объеме $K_j$ , не зависящем от количества произведенной продукции, и текущих издержек на производство единицы продукции.

Таким образом, если $x_j$ -- объем выпуска продукции $j$ -го вида, то функцию суммарных затрат можно представить в виде

$$C_j(x_j) = \begin{cases} K_j + c_j x_j, & \text{if } x_j > 0; \\ 0 \quad \text{if} \quad x_j = 0. \end{cases}$$

Естественно стремление минимизировать общие затраты по всем типам продукции:

$$Z = \sum_{j=1}^{N} C_j(x_j) \to \min. \tag{1}$$

Критерий (1) не является линейным вследствие разрыва в начале координат. Поэтому

он практически не пригоден для дальнейшего исследования. Введем новые булевы переменные, которые позволят нам записать критерий (1) в лучшем (с аналитической точки зрения) виде.

Пусть

$$y_j = \begin{cases} 0, & \text{if } x_j = 0, \\ 1, & \text{if } x_j > 0. \end{cases} \tag{2}$$

Последнее условие можно переписать в виде

$$x_j \leq M y_j, \tag{3}$$

где $M$ -- достаточно большое число. Оно должно быть таким, чтобы $x_j \leq M$ при любом допустимом значении $x_j$. Строго говоря, пока условие (3) не эквивалентно условию (2)!

Теперь рассмотрим критерий

$$Z = \sum_{j=1}^{N} (c_j x_j + K_j y_j) \to \min \tag{4}$$

при дополнительных ограничениях

$$0 \leq x_j \leq M y_j, \quad y_j = 0 \vee 1, \quad j = \overline{1, N}. \tag{5}$$

Покажем, что (4), (5) эквивалентны (1). Действительно, при $x_j > 0$ имеем $y_j = 1$ и

$$c_j x_j + K_j y_j = c_j x_j + K_j.$$

Пусть $x_j = 0$. Из (5) следует, что $y_j$ может быть равным 0 или 1. Однако, согласно (4), наша цель -- минимизировать критерий (4), следовательно, с учетом (4) мы обязательно должны выбрать $y_j = 0$ .

С аналитической точки зрения, функция (4) является хорошей, так как она является линейной и непрерывной.

Следует подчеркнуть, что исходная задача с постоянными элементами затрат не имела никакого отношения к целочисленному программированию. Однако преобразованная задача является частично целочисленной с булевыми переменными. Необходимость такого преобразования была вызвана чисто аналитическими причинами.

### 1.1.2 Задача планирования производственной линии

Предположим, что имеется один станок, на котором надо произвести *n* различных операций. На порядок выполнения некоторых операций наложены ограничения

технологического характера и, кроме того, есть ограничения, согласно которым каждая операция должна быть выполнена к заданному сроку. Таким образом, в задаче имеются ограничения трех типов:

1. ограничения на последовательность выполнения операций,
2. условие, что все операции выполняются на одном станке, т.е. операции нельзя распараллелить,
3. наличие временных сроков на осуществление операций

Рассмотрим первый тип ограничений. Пусть $x_j$ -- момент начала выполнения *j*-й операции, $a_j$ -- время, требуемое на выполнение $j$ -й операции. Если операция $i$ должна предшествовать операции $j$ и операция $j$ должна начаться не ранее, чем через $c_{ij}$, где $c_{ij} > a_i$ , после начала выполнения операции $i$ , то должно выполняться неравенство

$$x_i + c_{ij} \le x_j, \quad (i,j) \in U_*. \tag{6}$$

Пусть $i$ и $j$ --- две операции, для которых нет ограничений на последовательность их выполнения (т.е. $(i,j) \notin U_*$ ). Рассмотрим ограничение второго типа. Пусть $i$ и $j$ --- две операции, для которых нет ограничений на последовательность их выполнения (т.е. $(i,j) \notin U_*$ . Поскольку операции $i$ и $j$ , не могут выполняться на станке одновременно, то для $(i,j) \notin U_*$, должно выполняться одно из соотношений:

$x_i - x_j \ge a_j$ , если в оптимальном решении $j$ предшествует $i$ ,

или

$x_j - x_i \ge a_i$ , если в оптимальном решении $i$ предшествует $j$ .

Логическое ограничение вида «или--или» не укладывается в рамки обычной задачи ЛП, что вызывает существенные трудности при построении математической модели задачи. Эти трудности можно обойти путем введения вспомогательных булевых переменных
>br>

$$y_{ij} = \begin{cases} 0, & \text{если } j \text{ предшествует } i, \\ 1, & \text{если } i \text{ предшествует } j. \end{cases} \qquad (i,j) \notin U_*$$

Пусть $M > 0$ достаточно велико, тогда ограничение вида «или--или» эквивалентно следующей системе неравенств для $(i,j) \notin U_*$ :

$$My_{ij} + (x_i - x_j) \ge a_j, \tag{7}$$

$$M(1-y_{ij})+(x_j-x_i)\geq a_i,\ \ y_{ij}=0\vee 1. \tag{8}$$

Действительно, если на оптимальном решении $y_{ij}=0$ , то ограничение (8) становится избыточным, а ограничение (7) -- активным:

$$(x_i-x_j)\geq a_j,\quad M+(x_j-x_i)\geq a_i.$$

Если на оптимальном решении $y_{ij}=1$ , то ограничение (7) -- избыточное, а ограничение (8) -- активное:

$$M+(x_i-x_j)\geq a_j,\qquad (x_j-x_i)\geq a_i.$$

Таким образом, введение булевых переменных позволило избежать логического «или--или» с помощью **линейных** неравенств. Ограничения третьего типа учитываются следующим образом. Пусть операция $j$ должна быть завершена к моменту времени $d_j$ , тогда

$$x_j+a_j\leq d_j,\quad j\in J=\{1,2,...,n\}. \tag{9}$$

Если обозначить через $t$ суммарное время, требуемое для выполнения всех $n$ операций, то рассматриваемая задача принимает вид

$$Z=t\ \rightarrow\ \min$$

при ограничениях (6) -- (9) и

$$x_j+a_j\leq t,\quad j=\overline{1,n};y_{ij}=0\vee 1,\ i=\overline{1,n}\,;j=\overline{1,n},\ \ (i,j)\notin U_*.$$

### 1.1.3 Задачи с дихотомией

Предположим, что в некоторой ситуации требуется выполнение $k$ ограничений из общего числа $m$ ограничений. При этом заранее не известно, какие именно ограничения должны выполняться. Запишем ограничения в виде

$$g_i(x_1,...,x_n)\leq b_i,\quad i=1,2,...,m.$$

Определим новую переменную

$$y_i=\begin{cases}0\text{ , если }i\text{-е ограничение должно обязательно выполняться;}\\ 1\text{ , если }i\text{-е ограничение может нарушаться.}\end{cases}$$

Если $M>0$ достаточно велико, то выполнение по крайней мере $k$ ограничений из $m$ обеспечивается соблюдением соотношений:

$$g_i(x_1,...,x_n)\leq b_i+My_i,\quad y_i=0\vee 1,\quad i=\overline{1,m},$$

$$\sum_{i=1}^{m} y_i = m - k.$$

Аналогичная ситуация возникает, когда правая часть некоторого ограничения может принимать **одно** из нескольких значений:

$$g(x_1, x_2, ..., x_n) \le b, \quad b \in \{b_1, b_2, ..., b_k\}.$$

Данное ограничение можно записать в виде

$$g(x_1, ..., x_n) \le \sum_{s=1}^{k} b_s y_s, \quad \sum_{s=1}^{k} y_s = 1, \quad y_s = 0 \vee 1, \quad s = \overline{1,k},$$

где

$$y_s = \begin{cases} 1, & \text{if } b = b_s, \\ 0 & \text{otherwise} \end{cases}$$

Есть еще много типов задач, которые по постановке не относятся к целочисленным, но построение оптимального решения в этих задачах требует перехода к моделям целочисленного программирования: задача коммивояжера, задача составления расписания обработки деталей на станках, задача баланса сборочной линии и др. Некоторые из этих задач мы еще будем рассматривать далее подробно. Построение моделей целочисленного программирования для таких ситуаций в некоторой степени является искусством.

### 1.1.4 Постановка задачи линейного целочисленного программирования и ее связь с задачами линейного программирования

В общей постановке задача ЦЛП имеет вид:

$$c'x \to \max, \tag{10}$$

$$Ax = b, d_{*j} \le x_j \le d_j^*, \ j \in J, \tag{11}$$

$$x_j \text{ -- целое}, \ j \in I, \tag{12}$$

здесь $x = (x_j, j \in J), \ J = \{1, 2, ..., n\}, \ I \subset J; \ A \in R^{m \times n}, \text{rank A} = m, \ b \in R^m, \ c \in R^n.$

Встает вопрос: что сложнее -- задача ЦЛП (10) -- (12) или задача ЛП (10) -- (11)?

Многие ответят, что задача ЦЛП проще, так как в этой задаче меньше допустимых планов, например, если $-\infty < d_* \le d^* < \infty$, то количество допустимых планов конечно.

Однако справедливо обратное. Проиллюстрируем это на примере.

**Пример.** Рассмотрим задачу

$$21x_1 + 11x_2 \to \max, \tag{13}$$

$$7x_1 + 4x_2 \le 13, \quad x_1 \ge 0,\ x_2 \ge 0, \tag{14}$$

$$x_2,\ x_2 \text{ -- целые.} \tag{15}$$

Эта задача имеет единственный оптимальный план $x^0 = (x_1 = 0,\ x_2 = 3)$ (см. рис.~1.1).

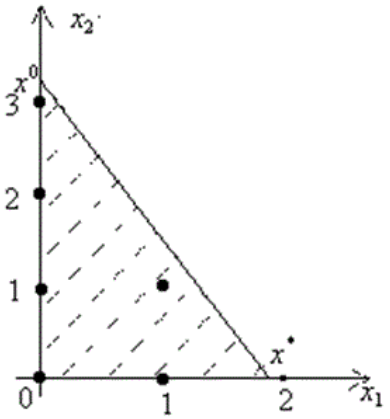


Рис. 1.1

Существует такое мнение, что задачи ЦЛП можно решать следующим образом: надо решить задачу без предположения целочисленности, а потом «округлить» полученное нецелочисленное решение до целочисленного.

Посмотрим, что мы получим, если воспользуемся этим приемом в нашей задаче. Оптимальное решение нецелочисленной задачи (13), (14) имеет вид $x^* = (x_1 = \frac{13}{7},\ x_2 = 0)$.

Очевидное округление приводит к вектору $(x_1 = 2, x_2 = 0)$ , который не является даже планом.

Округление в «меньшую сторону» приводит к плану $(x_1 = 1,\ x_2 = 0)$.

Этот план допустимый, но очень далек от оптимального.

Таким образом, мы видим, что «метод округления» нельзя считать универсальным.

Еще более неприятная особенность, свойственная задачам целочисленного программирования, состоит в том, что нет простого способа, позволяющего определить, является ли данный допустимый план оптимальным. В этом одно из главных отличий задач ЦЛП от задач ЛП. Чтобы проиллюстрировать это, предположим, что в задаче (13) -- (15) надо проверить, является ли план

$(x_1 = 1,\ x_2 = 1)$ оптимальным.

По аналогии с ЛП приходит мысль, что для этого надо проверить, соответствует ли точка $(x_1 = 1,\ x_2 = 1)$ какому-либо локальному оптимуму, в том смысле, что значение целевой функции не улучшится при переходе в любую соседнюю допустимую целочисленную точку:

$$x_1 = 1 + d,\ x_2 = 1 + l, \quad 345 \quad d, \ \ l = -1 \vee 0 \vee 1.$$

В нашем примере допустимые соседние точки имеют вид

$$(x_1 = 0, x_2 = 0), (x_1 = 0,\ x_2 = 1), (x_1 = 0,\ x_2 = 2), (x_1 = 1,\ x_2 = 0). \tag{16}$$

Легко проверить, что значение критерия качества в точке $(x_1 = 1, x_2 = 1)$ лучше, чем в точках (16). Однако вектор $(x_1 = 1, x_2 = 1)$ не является оптимальным планом!

Отметим, что для задач с булевыми переменными $x_j = 0 \vee 1$ проверка всех соседних дискретных точек сводится к полному перебору всех возможных планов. Возникает вопрос: существуют ли методы решения задач ЦЛП, отличные от полного перебора, либо полный перебор -- это единственный способ решения?

Если задача имеет небольшую размерность, то метод полного перебора можно использовать. Однако, если размеры большие, то полный перебор по времени выливается приблизительно «в целую жизнь». Например, при рассмотрении задачи ЦЛП, в которой 100 переменных и $x_j = 0 \vee 1$ , для полного перебора надо перебрать $2^{100}$ вариантов. Следовательно, нужны другие методы, отличные от полного перебора.

К настоящему времени наиболее популярными методами целочисленного программирования являются методы двух типов:

1. методы возврата
2. методы отсекающих плоскостей

Ниже рассматриваются метод ветвей и границ, относящийся к первой группе, и метод Гомори, относящийся ко второй группе методов.