МЕТОД ДИХОТОМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

В.Н. Бурков, И.В. Буркова, М.В. Попок
(*Институт проблем управления РАН, Москва*)

1. Введение

Многие задачи дискретной оптимизации сводятся к следующей постановке: определить вектор $x = \{x_i\}$ с дискретными компонентами, минимизирующий аддитивную функцию

(1) $$j(x) = \sum_{i=1}^{n} j_i(x_i)$$

при ограничении

(2) $$f(x) \geq b.$$

Широкий класс функций $f(x)$ допускает дихотомическое представление, такое, что вычисление значений функции сводится к последовательному вычислению значений функций двух переменных. Так функция

$$f(x) = f_0[f_1(x_1,x_2),\ f_2(x_2,x_3)]$$

допускает дихотомическое представление (рис. 1). При этом соответствующие функции f_0, f_1, f_2 удобно представлять в матричном виде (рис. 2).

Такое представление широко используется в методах комплексного оценивания программ развития предприятий, регионов, результатов деятельности подразделений, уровня безопасности объектов и др.

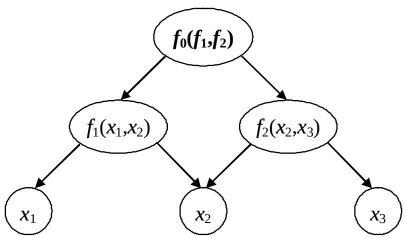

Рис. 1.

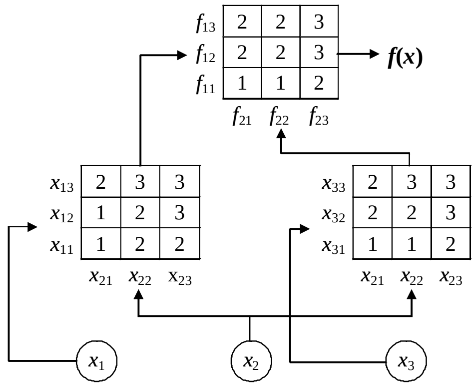

Рис. 2.

Колмогоровым А.Н. и Арнольдом В.И. [1, 3] доказаны теоремы о представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных (в частности, двух переменных). Так, например, любая непрерывная функция трех переменных представима в виде [3]

$$f(x_1,x_2,x_3) = h^1(x_1, j_1(x_2,x_3)) + h^2(x_1, j_2(x_2,x_3)) + h^3(x_1, j_3(x_2,x_3))$$

Ее дихотомическое представление (в агрегированном виде) – на рис. 3.

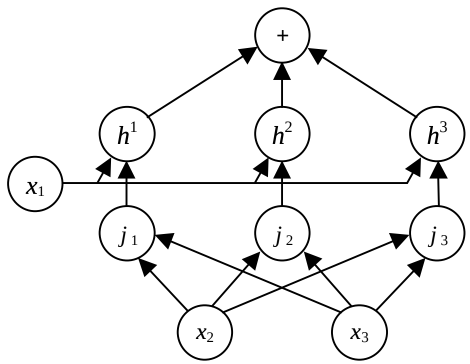

Рис. 3.

Поскольку функция дискретных переменных может быть продолжена до непрерывной функции, то, тем более, любая функция дискретных переменных представима в дихотомическом виде. В дихотомическом виде можно представить и систему неравенств. Рассмотрим, например, систему неравенств

(3) $$f_j(x) \le b_j, \quad j = \overline{1,m}$$

Без ограничения общности можно принять, что b_j – положительные и одинаковые числа, $b_j = b > 0$. В этом случае систему неравенств (3) можно заменить одним неравенством

$$f(x) \le b$$

где

$$f(x) = \max_j f_i(x)$$

Очевидно, что функция $f(x)$ допускает дихотомическое представление, если все функции f_j допускают такое представление.

2. Дихотомическое представление типа дерева

В задачах комплексного оценивания функция $f(x)$, дающая интегральную оценку объекта, как правило, допускает дихотомическое представление в виде дерева. В этом случае можно предложить эффективный метод решения задачи (1), (2). На рис. 4 приведен пример построения интегральной оценки трех показателей, имеющей вид $f(x_1,x_2,x_3) = j_0[f_1(x_1,x_2), x_3] = j_0(y,x_3)$

Значения функций $j_i(x_i)$ даны в нижних половинах квадратов, соответствующих переменным x_1, x_2 и x_3. Дадим описание алгоритма на примере рис. 4.

1 шаг. Рассматриваем нижнюю матрицу и для каждого элемента этой матрицы записываем в нижней половине соответствующей клетки сумму функций $j_1(x_1)$ и $j_2(x_2)$ для соответствующих значений x_1 и x_2. Так, например, клетке $(x_1,x_2) = (3, 2)$ соответствует сумма $j_1(3) + j_2(2) = 20 + 10 = 30$.

Далее будем называть эту сумму затратами на достижение соответствующего состояния.

f(x)	1	2	3	4
j (x)	6	25	67	120

4 / 70	2 / 71	3 / 78	4 / 120	4 / 170
3 / 30	2 / 31	2 / 38	3 / 80	3 / 130
2 / 17	1 / 18	2 / 25	3 / 67	3 / 117
1 / 5	1 / 6	1 / 13	2 / 55	3 / 105
y / x₃	1 / 1	2 / 8	3 / 50	4 / 100

4 / 50	2 / 52	3 / 57	4 / 70	4 / 110
3 / 35	1 / 37	2 / 42	3 / 55	3 / 95
2 / 10	1 / 12	2 / 17	3 / 30	3 / 70
1 / 3	1 / 5	1 / 10	2 / 23	2 / 63
x_2 / x_1	1 / 2	2 / 7	3 / 20	4 / 60

Рис. 4.

2 шаг. Из всех элементов матрицы имеющих одно и то же значение $y = f_1(x_1,x_2)$ выбираем элемент с минимальной суммой $j_1(x_1)+j_2(x_2)$. Минимальную сумму записываем в нижнюю половину клетки, соответствующей этому значению y в верхней матрице. Так, например, значению $y = 3$ соответствуют 5 элементов нижней матрицы: (3;2), (4;2), (3;3), (4;3) и (2;4). Из них элемент (3;2) имеет минимальную сумму 30 (это число записано в нижней половине соответствующей клетки). Поэтому в верхней матрице значению $y = 3$ соответствует число 30, записанное в нижней половине соответствующей клетки.

Далее шаги 1 и 2 повторяются для верхней матрицы. В результате для каждого значения $f(x)$ мы получаем минимальную величину $j(x)$.

Несложно обобщить описанный алгоритм на случай производльного дихотомического представления функции $f(x)$ в виде дерева. Шаги 1 и 2 алгоритма повторяются, начиная с висячих вершин дерева дихотомического представления.

Заметим, что дихотомическое представление Рис. 4 имеет тип ветви дерева. В этом случае метод дихотомического программирования переходит в метод динамического программирования. Таким образом, метод дихотомического программирования в случае, когда дихотомическое представление имеет вид дерева, является обобщением метода динамического программирования, расширяя круг задач, решаемых на основе данного подхода (Рис. 5).

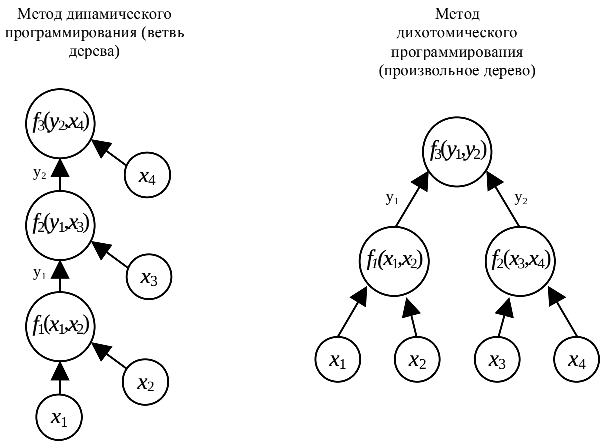

Рис. 5.

Если в методе динамического программирования решением задачи является путь в некоторой специальным образом построенной сети, то в методе дихотомического программирования решением задачи является частичное дерево в некотором специально построенном дереве. Соответственно, принцип оптимальности в методе дихотомического программирования можно сформулиро-

вать следующим образом: любое поддерево оптимального дерева должно быть оптимальным.

Формально этот принцип оптимальности можно записать следующим образом: $j_k(y) = \min\limits_{(i,j)\in p(y)} \left[j_i(y_i) + j_j(y_j) \right]$, где $P(y)$ – множество пар (i,j), такие что $f_k(y_i, y_j) = y$.

3. Общий случай

Рассмотрим произвольное дихотомическое представление функции $f(x)$, задаваемое сетью, входом которой является вершина, соответствующая функции $f(x)$, а выходами – вершины, соответствующие переменным x_i, $i = \overline{1,n}$. Рассмотрим множество конечных вершин, которые не являются висячими, то есть их степень захода больше 1. Разделим произвольным образом затраты $j_i(x_i)$ на k_i частей, где k_i – число заходящих дуг. Фактически мы как бы разделили вершину i на k_i висячих вершин с соответствующей частью затрат. Далее применяем описанный выше алгоритм. При этом каждый раз, когда встречается вершина, имеющая степень захода больше 1, мы делим затраты на соответствующее число частей. В результате применения алгоритма мы получим оптимальное решение для модифицированной сети. Однако это решение может не быть решением исходной задачи. Тем не менее, имеет место следующая теорема.

Теорема. Полученное с помощью вышеописанного алгоритма решение дает нижнюю оценку оптимального решения исходной задачи.

Доказательство. Заметим, что множество решений модифицированной сети содержит все решения исходной задачи. Эти решения имеют следующий вид. Если в вершину, соответствующую переменной x_{ik} заходит хотя бы одна дуга полученного решения, то все дуги, заходящие в эту вершину, также принадлежат полученному решению. Отсюда следует, что полученное оптимальное решение модифицированной задачи дает нижнюю оценку для оптимального решения исходной задачи.

Пример 1. Рассмотрим сеть рис. 1, 2. На рис. 6 приведено решение задачи. При этом затраты $j_2(x_2)$ разделены на две части, поскольку переменная x_2 используется и при вычислении f_1, и при вычислении f_2. В данном случае общие затраты, равные 8, 12 и 20 при значениях переменной x_2 равной 1, 2 и 3, соответственно, поделены пополам.

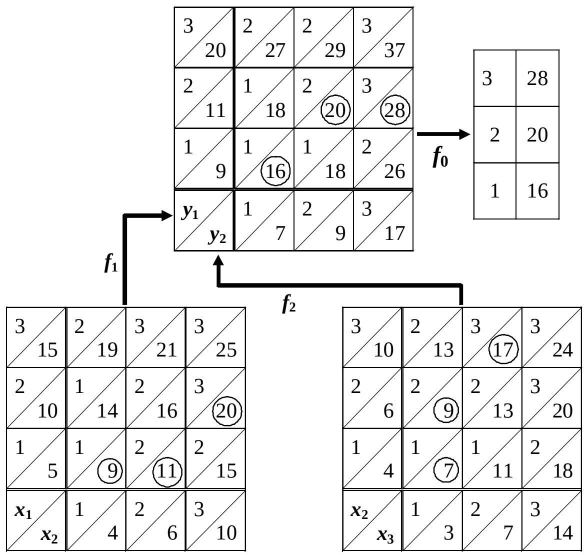

Рис. 6.

В каждой матрице выделены клетки, соответствующие минимальным затратам на получение того или иного значения функций (f_1, f_2 и f_0). В результате получены минимальные затраты $j(f_0)$, требуемые для получения значений функции f_0. Если $f_0 = 1$, то $j(1) = 16$, если $f_0 = 2$, то $j(2) = 20$, если $f_0 = 3$, то $j(3) = 28$.

Рассмотрим случай $f_0 = 2$. Ему соответствует оптимальное решение модифицированной задачи: $x_1 = 1$, $x_{21} = 2$, $x_{22} = 2$, $x_3 = 1$.

Здесь x_{21} соответствует значению x_2 в левой нижней матрице, а x_{22} – в правой нижней матрице. Поскольку оба значения $x_{21} = 2$, $x_{22} = 2$ вошли в оптимальное решение модифицированной задачи, то полученное решение является допустимым для исходной задачи, а значит мы получили оптимальное решение исходной задачи.

Другая ситуация возникает в случае $f_0 = 3$. Оптимальное решение модифицированной задачи имеет вид: $x_1 = 1$, $x_{21} = 2$, $x_{22} = 3$, $x_3 = 2$ с величиной затрат $j_0 = 28$. Это решение не является допус-

тимым для исходной задачи, поэтому $j_0 = 28$ является нижней оценкой минимальных затрат для исходной задачи. Здесь возможны два варианта действий. Первый заключается в попытке улучшить нижнюю оценку, изменяя разбиение затрат $c_2 = j_2(x_2)$ на две части – c_{21} и c_{22}. Очевидно, что для улучшения оценки следует c_{21} увеличить, а c_{22} уменьшить. Возьмем, например, $c_{21} = 10$, а $c_{22} = 2$. В этом случае оптимальное решение модифицированной задачи будет иметь вид: $x_1 = 1$, $x_{21} = 2$, $x_{22} = 2$, $x_3 = 3$ с величиной затрат $j_0 = 31$. Это решение является допустимым для исходной задачи, а значит оптимальным. Однако, изменение разбиения затрат на части может и не привести к получению допустимого решения для исходной задачи.

Второй вариант состоит в применении метода ветвей и границ. Разобьем множество всех решений исходной задачи на два подмножества. В первом $x_2 \le 2$, а во втором $x_2 = 3$ и применим описанный выше алгоритм. Получим оценку снизу для первого подмножества. Получаем следующее решение: $x_1 = 1$, $x_{21} = 2$, $x_{22} = 2$, $x_3 = 3$ с величиной затрат $j_0 = 31$.

Получим оценку снизу для второго подмножества. Оптимальное решение модифицированной задачи имеет вид: $x_1 = 1$, $x_2 = 3$, $x_3 = 2$ с величиной затрат $j_0 = 32$.

Выбираем первое подмножество с минимальной оценкой. Поскольку полученное решение модифицированной задачи является допустимым для исходной задачи, то оно является оптимальным.

Рассмотрим на ряде задач построение оценочной задачи и метод ветвей и границ на основе полученной оценки.

4. Задача целочисленного линейного программирования

Рассмотрим следующую постановку задачи целочисленного линейного программирования. Определить целочисленный неотрицательный вектор $x = \{x_1, x_2, \ldots, x_4\}$ максимизирующий

(4) $$j(x) = \sum_{i=1}^{n} c_i x_i$$

при ограничениях

(5) $$\sum_i a_{ij} x_i \le b_j, \quad j = \overline{1, m}$$

Для построения оценочной задачи разделим c_i на m частей s_{ij}, так что

(6) $$\sum_j s_{ij} = c_i,\ i = \overline{1, n}$$

и рассмотрим m задач целочисленного линейного программирования следующего вида: определить целочисленный вектор x, максимизирующий

(7) $$s_j(x) = \sum_i s_{ij} x_i$$

при ограничениях

(8) $$\sum_i a_{ij} x_i \le b_j .$$

Обозначим через $\Phi_j(s_j)$ – оптимальное решение j-ой задачи ($s_j = \{s_{ij}\}$). Согласно теореме, величина

(9) $$\Phi(\mathrm{s}) = \sum_{j=1}^{m} \Phi_{\mathrm{j}}(s_j)$$

является оценкой сверху оптимального решения исходной задачи. Окончательно получаем следующую формулировку оценочной задачи: определить $\{s_{ij}\}$, $i = \overline{1, n}$, $j = \overline{1, m}$, максимизирующие $\Phi(s)$ при ограничениях (6). Оценочную задачу назовем двойственной к исходной задаче целочисленного линейного программирования. Обоснованием этого названия служит следующая интересная связь. Рассмотрим обычную задачу линейного программирования (4)-(5) (без требования целочисленности). Для упрощения выводов примем, что все параметры системы ограничений – положительные числа. Заметим, что если не требовать целочисленности решений, то задача (7)-(8) легко решается. Ее оптимальное решение:

(10) $$x_{ij} = \begin{cases} \dfrac{b_j}{a_{kj}}, \text{ если } \dfrac{s_{kj}}{a_{kj}} = \max\limits_q \dfrac{s_{qj}}{a_{qj}}, \\ 0, \text{ если } i \ne k. \end{cases}$$

Оптимальная величина (7) составит

(11) $$\Phi_j(\mathrm{s_j}) = b_j \max_q \frac{s_{qj}}{a_{qj}} .$$

Обозначим $y_j = \max\limits_q \dfrac{s_{qj}}{a_{qj}}$, $j = \overline{1, m}$. Заметим, что $y_j \ge \dfrac{s_{qj}}{a_{qj}}$ для всех q. Увеличим s_{qj} так, чтобы $s_{qj} = y_j a_{qj}$. Тогда оценочная задача (6), (9) запишется в следующем виде: определить $y_j \ge 0$, $j = \overline{1, m}$, минимизирующие

(12) $$B(y) = \sum_j b_j y_j$$

при ограничениях

(13) $$\sum_j a_{ij} y_j \ge c_i,\ i = \overline{1, n}.$$

Таким образом, в непрерывном случае оценочная задача становится двойственной задачей линейного программирования.

Рассмотрим на примере применение метода ветвей и границ для решения задачи целочисленного линейного программирования.

Пример. Определить $x_i = \{0;1\}$, $i = \overline{1, n}$, максимизирующие

$$\Phi(x) = 10x_1+8x_2+6x_3+7x_4$$

при ограничениях

$$6x_1+3x_2+2x_3+5x_4 \le 11,$$
$$3x_1+5x_2+6x_3+3x_4 \le 11.$$

Для построения оценочных задач возьмем $s_{i1} = a_{i1}$, $s_{i2} = c_i - a_{i1}$, $i = \overline{1, 4}$. Получаем две задачи о ранце.

Задача 1.

$$j_1 = \max(6x_1+3x_2+2x_3+5x_4),$$
$$6x_1+3x_2+2x_3+5x_4 \le 11.$$

Она имеет два решения:

1) $x_1 = x_4 = 1,\quad x_2 = x_3 = 0,$
2) $x_1 = x_2 = x_3 = 1,\quad x_4 = 0.$

В обоих случаях $\varphi 1 = 11$.

Задача 2.

$$j_2 = \max(4x_1+5x_2+4x_3+2x_4),$$
$$3x_1+5x_2+6x_3+3x_4 \le 11.$$

Ее решение:

$$x_1 = x_2 = x_4 = 1,\quad x_3 = 0,\ j_2 = 11.$$

Оценка сверху исходной задачи

$$j_0 = j_1 + j_2 = 22.$$

Для улучшения оценки увеличим s_{21} и s_{41} на единицу, уменьшив на единицу s_{22} и s_{42}. Получаем две новые оценочные задачи:

Задача 1.

$$j_1 = \max(6x_1+4x_2+2x_3+6x_4),$$

(14) $$6x_1+3x_2+2x_3+5x_4 \le 11.$$

Ее решения:

$$1)\ x_1 = x_4 = 1,\quad x_2 = x_3 = 0,$$
$$2)\ x_1 = x_2 = x_3 = 1,\quad x_4 = 0,$$
$$3)\ x_2 = x_3 = x_4 = 1,\quad x_1 = 0,$$
$$j_1 = 12.$$

Задача 2.

$$j_2 = \max(4x_1+4x_2+4x_3+1x_4),$$

(15) $$3x_1+5x_2+6x_3+3x_4 \le 11.$$

Ее решение:

$$x_1 = x_2 = x_4 = 1,\quad x_3 = 0,$$
$$j_2 = 9.$$

Оценка сверху исходной задачи уменьшилась на единицу:

$$j_0 = j_1 + j_2 = 21.$$

Применим метод ветвей и границ. Разобьем множество всех решений на два подмножества. В первом подмножестве $x_1 = 1$, а во втором $x_1 = 0$.

Оценим первое подмножество. Положив в (14) и (15) $x_1 = 1$, получаем следующие две задачи:

Задача 1.

$$j_1 = \max(4x_2+2x_3+6x_4),$$
$$3x_2+2x_3+5x_4 \le 5.$$

Ее решения:

$$1)\ x_4 = 1,\quad x_2 = x_3 = 0,$$
$$2)\ x_2 = x_3 = 1,\quad x_4 = 0,$$
$$j_1 = 6.$$

Задача 2.

$$j_2 = \max(4x_2+4x_3+1x_4),$$
$$5x_2+6x_3+3x_4 \le 8.$$

Ее решение:

$$x_2 = x_4 = 1,\quad x_3 = 0,$$

$$j_2 = 5.$$

Оценка сверху первого подмножества:

$$j_0 = j_1 + j_2 + c_1 = 21.$$

Оценим второе подмножество ($x_1 = 0$). Заметим, что при $x_1 = 0$ любое решение является допустимым для первой оценочной задачи. Поэтому достаточно решить вторую задачу, положив $s_{i2} = c_i$, $i = 2, 3, 4$. Ее решение

$$x_2 = x_3 = 1,\quad x_4 = 0$$

является оптимальным во втором подмножестве со значением целевой функции $j_0 = 14$. Выбираем первое подмножество, имеющее большую оценку. Разбиваем первое подмножество на два. В одном из них $x_2 = 1$, а в другом – $x_2 = 0$.

Оценим первое подмножество ($x_2 = 1$). Рассматривая два ограничения

$$2x_3+5x_4 \le 2,$$
$$6x_3+3x_4 \le 3,$$

видим, что единственное решение

$$x_3 = x_4 = 0,$$

следовательно оно является оптимальным решением в данном подмножестве со значением целевой функции $j_0 = 18$.

Оценим второе подмножество ($x_2 = 0$). В данном случае достаточно сравнить два варианта:

$$1)\ x_3 = 1,\ x_4 = 0\ \ \varphi_1 = 6$$
$$2)\ x_3 = 0,\ x_4 = 1,\ \varphi_1 = 7$$

Оценка второго подмножества:

$$\varphi_o = 10 + 7 = 17.$$

Ей соответствует оптимальное решение в этом подмножестве

$$x_1 = x_4 = 1,\ x_2 = x_3 = 0,$$

со значением целевой функции $\varphi_o = 17$.

Выбираем первое подмножество, а следовательно, и оптимальное решение $x_1 = x_2 = 1$, $x_3 = x_4 = 0$, $\varphi_o = 18$. Дерево ветвлений приведено на Рис. 7.

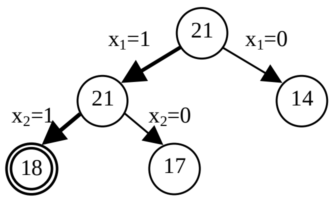

Рис. 7.

Интересно сравнить описанный способ получения оценок для задач целочисленного линейного программирования с известными способами. В основном применяются два способа. В первом решаются m задач о ранце с каждым ограничением отдельно. Очевидно, что наихудшее решение (по значению целевой функции) определяет оценку сверху исходной задачи. Для нашего примера имеем две задачи о ранце

Задача 1.

$$\max\,(10x_1 + 8x_2 + 6x_3 + 7x_4)$$
$$6x_1 + 3x_2 = 2x_3 + 5x_4 \le 11$$

Ее решение

$$x_1 = x_2 = x_3 = 1,\ x_4 = 0,\ \varphi_1 = 24$$

Задача 2.

$$\max\,(10x_1 + 8x_2 + 6x_3 + 7x_4)$$
$$3x_1 + 5x_2 = 6x_3 + x_4 \le 11$$

Ее решение

$$x_1 = x_2 = x_4 = 1,\ x_3 = 0,\ \varphi_1 = 25.$$

Таким образом, оценка сверху оптимального решения исходной задачи в данном случае равна 24, что больше чем оценка 21, полученная методом дихотомического программирования. Более того, поскольку обе рассмотренные задачи о ранце получаются в методе дихотомического программирования (первая – при $s_{i1} = c_i$, $s_{i2} = 0$, а вторая наоборот, при $s_{i2} = c_i$, $s_{i1} = 0$), то можно утверждать, что применение метода дихотомического программирования дает лучшие (или такие же) оценки. Во втором способе решается задача линейного программирования без требования целочисленности. Рассмотрим простой пример.

Пример. Определить $x_i = \{0; 1\}$, $i = 1,2$ максимизирующее

$$j\,(x) = 42x_1 + 14x_2$$

При ограничениях

$$3x_1 + 4x_2 \le 6$$
$$5x_1 + 2x_2 \le 6$$

Получим оценку методом дихотомического программирования. Для этого достаточно положить $s_{11} = 42$, $s_2 = 14$, $s_{12} = s_{22} = 0$. Решение первой оценочной задачи $x_1 = 1$, $x_2 = 0$, $\varphi_1 = 42$ является допустимым для второй задачи и, следовательно, является оптимальным.

Получим оценку, решая нецелочисленную задачу линейного программирования. Ее решение

$$x_1 = \frac{6}{7};\quad x_2 = \frac{6}{7};\quad j_0 = 48 > 42$$

Как видим, оценка существенно хуже.

5. Решение «Задачи о камнях» методом дихотомического программирования

Рассмотрим постановку «задачи о камнях». Имеется n «камней» разного веса. Требуется разбить их на m групп (куч) так, чтобы максимальный вес камней в группе был минимален. Задача о камнях имеет многочисленные варианты применения (равномерное распределение работ между исполнителями, функций по подразделениям организационной структуры и т.д.). Дадим формальную постановку задачи.

Задача 1. Обозначим через a_i – вес i-го камня, $x_{ij} = 1$ если камень i попал в j-ю кучку, $x_{ij} = 0$ в противном случае. Суммарный вес камней в j-й группе равен

(16) $$T_j = \sum_i a_i x_{ij}\,.$$

Максимальный вес группы

(17) $$T = \max_j \sum_i a_i x_{ij} \to \min\,.$$

Поскольку каждый камень должен быть помещен только в одну группу, имеем ограничения:

(18) $$\sum_j x_{ij} = 1,\ i = \overline{1,n}\,.$$

Задача заключается в минимизации (17) при ограничениях (18). Мы будем рассматривать вспомогательную задачу следующего вида:

Задача 2. Фиксируем допустимый вес каждой группы T и сформулируем следующую задачу: максимизировать сумму весов размещенных в ящики вместимостью T камней:

(19) $$\Phi = \sum_{i,j} a_i x_{ij} \to \max$$

при ограничениях (18) и (20):

(20) $$\sum_i a_i x_{ij} \le T, \quad j = \overline{1,m}.$$

Связь между задачами (17)-(18) и (18)-(20) очевидна. Минимальное T, при котором в оптимальном решении задачи 2 размещены все камни, определяет оптимальное решение задачи 1.

Сначала получим дихотомическое представление задачи 2. Оно в агрегированном виде представлено на рис. 8 для случая $n = 3$, $m = 2$.

Поскольку структура дихотомического представления имеет вид сети, а не дерева, то для построения оценочной задачи разделяем каждую вершину нижнего уровня на две вершины. Преобразованная структура приведена на рис. 9. Все a_i также делим на 2 части u_{ij} и v_{ij} для каждой вершины нижнего уровня так, что

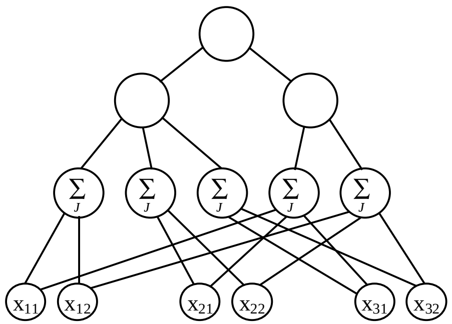

Рис. 8.

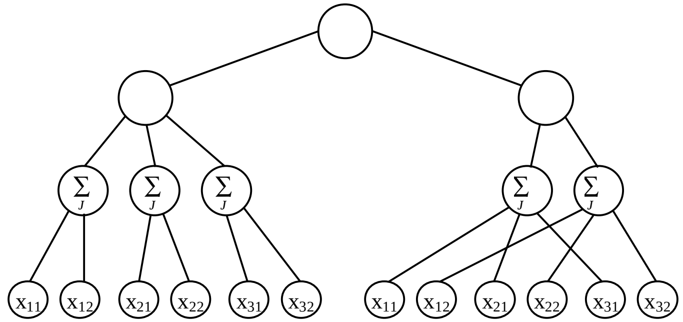

Рис. 9.

(21) $$u_{ij} + v_{ij} = a_i \quad \text{для всех } i, j.$$

Рассмотрим следующие две задачи.

Задача 1. Определить x_{ij} так, чтобы максимизировать

(22) $$\sum_{i,j} u_{ij} x_{ij}$$

при ограничениях (3).

Вторая задача. Максимизировать

(23) $$\sum_{i,j} v_{ij} x_{ij}$$

при ограничениях (20).

Обозначим $S_m(u)$ и $L_m(v)$ оптимальные решения первой и второй задач при заданных u и v. Оценочная задача заключается в определении $\{u_{ij}\}$ и $\{v_{ij}\}$, минимизирующих

(24) $$F(u,v) = S_m(u) + L_m(v)$$

при ограничении (21).

Заметим, во-первых, что в оптимальных решениях первой и второй задач можно принять

$$u_{ij} = y_i, \quad v_{ij} = a_i - y_i, \quad j = \overline{1,m}.$$

Во-вторых, решение первой задачи очевидно:

(25) $$S_m(x) = \sum_i y_i.$$

В третьих, решение m вторых задач при заданных $\{y_i\}$ сводится к решению одной задачи о ранце: определить $x_i = \overline{0,1}$, максимизирующие

(26) $$\sum_i x_i(a_i - y_i)$$

при ограничении

(27) $$\sum_i x_i a_i \le T .$$

Решим задачу (26)-(27) при $y_i = 0,\ i = \overline{1,n}$.

Обозначим через $Q = \{Q_j\}$ множество векторов x, удовлетворяющих (27) и упорядоченных по убыванию $M_j = \sum_{i \in Q_j} a_i$, $Y_j = \sum_{i \in Q_j} y_i$, а

(28) $$Z = \max_j (M_j - Y_j).$$

Заметим, что при заданных $\{y_i\}$ Z определяет оптимальное решение каждой из m вторых задач. Оценка (24) при этом равна

(29) $$F(y) = mZ + \sum_i y_i ,$$

где $y_i \ge 0$ удовлетворяют неравенствам

(30) $$\sum_{i \in Q_j} y_i + Z \ge M_j ,\quad j = \overline{1,N} ,$$

где N – число различных решений неравенства (27). Таким образом, оценочная задача свелась к определению $0 \le y_i \le a_i$, $i = \overline{1,n}$ и $0 \le Z \le M_j$, максимизирующих (29) при ограничениях (30). Это обычная задача линейного программирования.

Фиксируем величину Z и определяем максимальный номер k такой, что $Z < M_k$. Рассматриваем следующую задачу линейного программирования: определить $0 \le y_i \le a_i$, $i = \overline{1,n}$, минимизирующие

(31) $$Y(Z) = \sum_i y_i ,$$

при ограничениях (30), где $j = \overline{1,k}$. Двойственная задача имеет вид: определить $u_j \ge 0$, $j = \overline{1,k}$, максимизирующие

(32) $$\sum_{j=1}^{k} (M_j - Z) u_j ,$$

при ограничениях

(33) $$\sum_{j \in R_i} u_j \le 1,\quad i = \overline{1,n} ,$$

где R_i – множество j, содержащих камень i.

Обозначим через $Y_0(Z)$ минимальное значение $Y(Z)$. Оценочная задача сводится к минимизации функции одного переменного

(34) $$Y_0(Z) + mZ \to \min.$$

Берем $T_0 = A/m$, где $A = \sum_i a_i$, и решаем задачу 2. Если $\Phi_{\max}(T_0) < A$, то увеличиваем T_0 до T_1 так, чтобы появился хотя бы один новый вектор Q_j. Если $\Phi_{\max}(T_1) < A$, то продолжаем увеличение T до тех пор, пока не получим величину T_k такую, что $\Phi_{\max}(T_k) \ge A$. Величина T_k является нижней оценкой для задачи 1. Далее можно применить метод ветвей и границ на основе полученной оценки.

Пример. Пусть $m = 3$ и имеется 7 камней следующего веса:

i	1	2	3	4	5	6	7
a_i	10	12	13	14	18	19	22

1 шаг. Имеем $A = 108$, $T_0 = 36$. Имеется только одно решение: $Q = (4,7)$ с величиной $M = 36$.

2 шаг. Увеличиваем T_0 до $T_1 = 37$. Имеются следующие решения: $Q_1 = (5,6)$, $Q_2 = (4,7)$, $Q_3 = (1,2,3)$, $Q_4 = (3,7)$. Соответственно $M_1 = 37$, $M_2 = 36$, $M_3 = 35$, $M_4 = 35$. Выпишем систему неравенств:

$$\begin{aligned} y_5 + y_6 + z &\ge 37 \\ y_4 + y_7 + z &\ge 36 \\ y_1 + y_2 + y_3 + z &\ge 35 \\ y_3 + y_7 + z &\ge 35 \end{aligned}$$

Имеем: $Z_0 = 37$, $y_i = 0$, $Y_0(37) = 111$,
$Z_1 = 36$, $y_5 = 1$, $Y_0(36) = 109$,
$Z_2 = 35$, $y_5 = 2$, $y_4 = 1$, $Y_0(35) = 108$.

Нетрудно показать, что дальнейшее уменьшение Z не приводит к уменьшению оценки. Поэтому оптимальное $Z_0 = Z_2 = 35$.

Берем $x_{51} = x_{61} = 1$, то есть, помещаем камни 5 и 6 в первую группу. Исключая эти камни, рассматриваем задачу меньшей размерности. Имеем для нее также $Y_0(35) = 71 = A - 37$. Получаем оптимальное решение: $x_{12} = x_{22} = x_{32} = 1$, $x_{43} = x_{73} = 1$ со значением $T_{\min} = 37$.

6. Заключение

Предложенный подход обобщается на случай любого сетевого представления системы ограничений. Главное, чтобы задачи в вершинах сетевого представления имели эффективные методы решения. Такой более общий подход можно назвать методом сетевого программирования. Заметим, что в «задаче о камнях» мы уже не использовали дихотомическое, а уже использовали сетевое представление.

Литература

1. АРНОЛЬД В.И. *О функциях трех переменных* // ДАН СССР, 1957. № 2.

2. БУРКОВ В.Н., БУРКОВА И.В. *Задачи дихотомической оптимизации* / Материалы международной научно-технической конференции «Системные проблемы качества, математического моделирования, информационных и электронных технологий». М.: Радио и связь, 2003. С. 23 – 28.

3. КОЛМОГОРОВ А.Н. *О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных* // ДАН СССР. 1956. Том 108. № 2.