УДК 002.53+004.65+004.62/.63+338.2
ББК 32.816, 65.05.0.2

МЕТОД СЕТЕВОГО ПРОГРАММИРОВАНИЯ В ЗАДАЧАХ УПРАВЛЕНИЯ ПРОЕКТАМИ

Бурков В. Н.[1], Буркова И. В.[2]
(*Учреждение Российской академии наук*
Институт проблем управления РАН, Москва)

Метод сетевого программирования разработан для получения точных решений или верхних (*нижних*) *оценок задач многоэкстремальной* (*в частном случае – дискретной*) *оптимизации. Идея метода заключается в представлении задачи в виде суперпозиции более простых задач. Такое представление удобно изображать в виде сети* (*сетевое представление*)*, вершины которой соответствуют задачам, входящим в суперпозицию. В каждой вершине решаются простые задачи оптимизации. Решение задачи в конечной вершине сети дает верхнюю* (*нижнюю*) *оценку для исходной задачи. Если сетевое представление является деревом, то решение задачи в конечной вершине сети дает оптимальное решение исходной задачи. В статье дается обзор применения метода для решения различных задач управления проектами.*

Ключевые слова: сетевое программирование, управление проектами, дискретная оптимизация.

1. Введение

Задачи оптимизации в управлении проектами в основном связаны с определением предметной области проекта (т. е. состава работ проекта) и с построением календарного плана при ограниченных ресурсах. Как правило, это задачи многоэкстре-

[1] *Владимир Николаевич Бурков, д.т.н., профессор* (*vlab17@bk.ru*)
[2] *Ирина Владимировна Буркова, к.т.н.* (*irbur27@mail.ru*)

мальной (в частном случае – дискретной) оптимизации. В общем случае для их решения применяются эвристические алгоритмы. Из точных методов следует отметить метод ветвей и границ и метод динамического программирования. Эффективность метода ветвей и границ в большой степени зависит от качества верхних или нижних оценок (границ). Для получения таких оценок достаточно универсальным является метод множителей Лагранжа.

В 2004 году авторами был предложен метод дихотомического программирования [3], а несколько позднее его обобщение – метод сетевого программирования (МСП) [4]. Как метод точного решения задач, МСП обобщает метод динамического программирования, а как метод получения верхних (нижних) оценок – метод множителей Лагранжа.

В статье дается обзор применения МСП к задачам управления проектами. Дадим краткое описание МСП.

2. Метод сетевого программирования

Рассмотрим следующую задачу дискретной оптимизации: определить вектор $x \in X$, обеспечивающий

(1) $\max\limits_{x \in X} f(x)$

при ограничении

(2) $\varphi(x) \le b$.

Далее будем предполагать, что $X = \prod\limits_i X_i$,где X_i – дискретное множество чисел.

Любая функция дискретных переменных допускает сетевое представление, такое, что вычисление значений функции сводится к последовательному вычислению значений более простых функций.

***Пример* 1.** Рассмотрим функцию четырех переменных

(3) $f(x) = (x_1x_2 + x_1^2)x_3 + x_3x_4 + x_1x_4$.

Ее сетевое представление приведено на рис. 1, где

$$y_1 = x_1x_2 + x_1^2,\ y_2 = y_1x_3,\ y_3 = x_1x_4,\ y_4 = x_3x_4,\ y_5 = y_2 + y_3 + y_4.$$

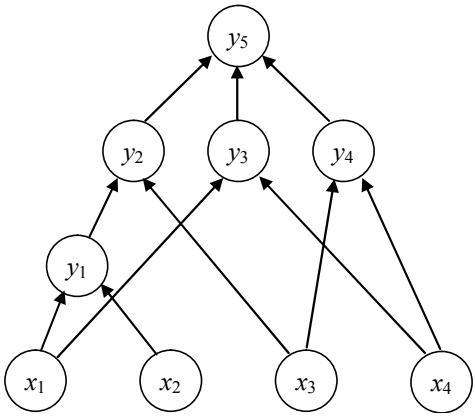

Рис. 1. Пример сетевого представления функции

***Определение* 1.** Функции $f(x)$ и $\varphi(x)$ называются структурно-подобными (с-подобными) в заданном классе сетевых представлений, если существуют сетевые представления этих функций такие, что соответствующие сетевые структуры совпадают.

***Пример* 2.** Рассмотрим функцию

(4) $\varphi(x) = (x_1 + x_2)x_3(x_1 + x_4) + x_3x_4$.

Нетрудно показать, что одно из сетевых представлений этой функции имеет вид рис. 1.

Пусть функции $f(x)$ и $\varphi(x)$ в задаче (1), (2) являются с-подобными. Построим их общее сетевое представление. Пусть далее целевая функция f является монотонной функцией обобщенных переменных y (без ограничения общности можно принять, что f – возрастающая функция y). Аналогично примем, что функция φ также является возрастающей функцией y. В сетевом представлении выделим вершины нулевого уровня, которым соответствуют переменным x_i. Вершинам первого уровня соответствуют задачи оптимизации следующего вида: максимизировать

(5) $y_i = f_i(x)$ при ограничении

$z_i = \varphi_i(x) \le p$,

где p принимает все допустимые значения.

Как уже отмечалось выше, в сетевом представлении те задачи, которые соответствуют вершинам сетевого представления (за исключением вершин первого уровня) имеют более простой вид, такой, что существуют эффективные алгоритмы их решения. В частности, если сетевое представление является дихотомическим, то каждая задача является задачей оптимизации функции двух переменных и в дискретном случае легко решается на основе матричного представления. Решив задачи первого уровня, переходим к решению задач второго уровня, и т.д. Последней решается задача, соответствующая выходу сети. Обозначим $y_k(b)$ – значение целевой функции в оптимальном решении задачи, соответствующей выходу сети.

***Теорема* 1.** Величина $y_k(b)$ является верхней оценкой для исходной задачи (1), (2).

3. Задача выбора портфеля проектов

Имеется n проектов, каждый из которых характеризуется затратами a_i на реализацию и эффектом c_i от реализации проекта. Задача заключается в выборе портфеля проектов с максимальным суммарным эффектом при заданной величине R инвестиционного фонда. Формально задача сводится к классической задаче о ранце [6].

Математическая постановка задачи имеет вид:

(6) $f(x) = \sum_j c_j x_j \to \max$,

(7) $\varphi(x) = \sum_j a_j x_j \le R$,

(8) $x_j \in \{0, 1\}, \quad j = \overline{1, n}$.

Структура сетевого представления является деревом, поэтому метод сетевого или дихотомического программирования дает оптимальное решение.

Различные обобщения этой задачи связаны с учетом ограничений на совместимость проектов, с учетом риска или ограничений на число проектов в тех или иных областях рассмотрены в [1, 2].

4. Задача календарного планирования

Рассмотрим задачу формирования календарного плана реализации проекта, состоящего из n работ.

Примем, что плановый период разбит на T интервалов определенной длины Δ (недели, месяцы, кварталы и т. д.)

Обозначим R_i – множество интервалов, в которых может выполняться работа i; P_{sj} – множество работ j-го вида которые могут выполняться в s-ом интервале. Заданы ограничения Q_{tj} на объем проектных работ каждого вида в каждом интервале. Для каждой проектной работы, в свою очередь, задан объем работ, выполняемый ресурсами каждого вида. Более того, примем, что каждая работа выполняется только одним видом ресурсов. Таким образом, все работы разбиты на m подмножеств так, что работы j-го подмножества выполняются ресурсами j-го вида. Обозначим через x_{is} объем i-ой работы, выполняемый в s-ом интервале; c_{is} – максимальный объем i-ой работы, который можно выполнить в s-ом интервале. Задача заключается в определении $\{x_{is}\}$, $i=\overline{1,n}$, $s=\overline{1,T}$, так, чтобы

(9) $x_{is} \le C_{is}$, $i \in P_s$, $s=\overline{1,T}$,

$$\sum_{s\in R_1} x_{is} \le W_i,\ i=\overline{1,n},$$

где W_i – объем i-ой работы,

(10) $$\sum_{i\in P_{sj}} x_{is} \le Q_{sj},\ j=\overline{1,m},$$

а суммарный объем выполненных работ j-го вида

(11) $$\sum_{s=1}^{T}\sum_{i\in P_{sj}} x_{is}$$

был максимален.

Ограничимся случаем независимых работ. Для решения задачи определим двудольный граф $G(X, Y)$. Вершины $i \in X$ соответствуют работам, а вершины $s \in Y$ соответствуют интервалам. Пропускные способности вершин $i \in X$ равны объектам W_i соответствующих работ, а пропускные способности вершины

$s \in Y$ равны объему работ, который можно будет выполнить в соответствующем интервале, т. е. Q_s.

Пропускные способности дуг (i, s), $i = \overline{1,n}$, $s \in R_i$, равны C_{is}. Задача свелась к определению максимального потока в полученной сети, что соответствует минимальному объему работ, отдаваемых на субподряд. Опишем алгоритм определения потока максимальной величины, основанный на методе сетевого программирования.

ЗАДАЧА О МАКСИМАЛЬНОМ ПОТОКЕ

Рассмотрим произвольную сеть без контуров на дугах (i, j), которой заданы пропускные способности $C_{ij} > 0$. Как известно, потоком в сети называется совокупность чисел $0 \le x_{ij} \le c_{ij}$, $(i, j) \in U$ (U – множество дуг сети), таких что

(12) $\sum_j x_{ij} = \sum_k x_{ki}$

для всех $i \ne 0$, n, где 0 – вход сети, а n – выход сети. Величиной потока называется

(13) $\varphi_0 = \sum_i x_{0i} = \sum_j x_{jn}$.

Задача заключается в определении потока максимальной величины. Для решения этой задачи, как правило, применяется алгоритм Форда-Фалкерсона. Мы рассмотрим другой подход, в основе которого лежит метод сетевого программирования.

Сначала дадим определение агрегируемой сети.

***Определение* 2.** Последовательным множеством называется подмножество дуг сетевого графика, образующих путь такой, что любая вершина, за исключением начальной, имеет степень захода 1, и любая вершина, за исключением конечной, имеет степень исхода 1 (рис. 2).

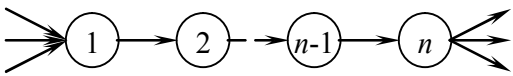

Рис. 2. К определению последовательного множества

Заметим, что последовательное множество дуг можно агрегировать в одну дугу, пропуская способность которой равна минимальной из пропускных способностей дуг подмножества.

***Определение* 3.** Параллельным множеством дуг называется подмножество дуг сети, у которых начальная и конечная вершины совпадают.

Параллельное множество дуг можно агрегировать в одну дугу, пропуская способность которой равна сумме пропускных способностей дуг подмножества.

***Определение* 4.** Сеть называется агрегируемой, если путем агрегирования последовательных и (или) параллельных множеств дуг ее можно свести к одной дуге.

Задача о максимальном потоке для агрегируемой сети эффективно решается. Как известно, задача о максимальном потоке для агрегируемой сети может быть представлена сетевой структурой типа дерева. На рис. 3. приведен пример агрегируемой сети (числа у дуг равны пропускным способностям), а на рис. 4. сетевое представление задачи о максимальном потоке. Числа в нижних половинах вершин равны пропускным способностям соответствующих дуг, а в верхних половинах указаны операции, которые проводятся над числами вершин нижних уровней для определения величины максимального потока. Операции взятия минимума выполняются для последовательного множества дуг, а операция суммирования для параллельного множества дуг. Число в корневой вершине дерева равно величине максимального потока.

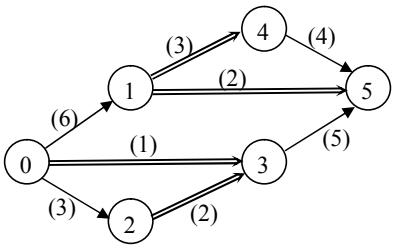

Рис. 3. Пример агрегируемой сети

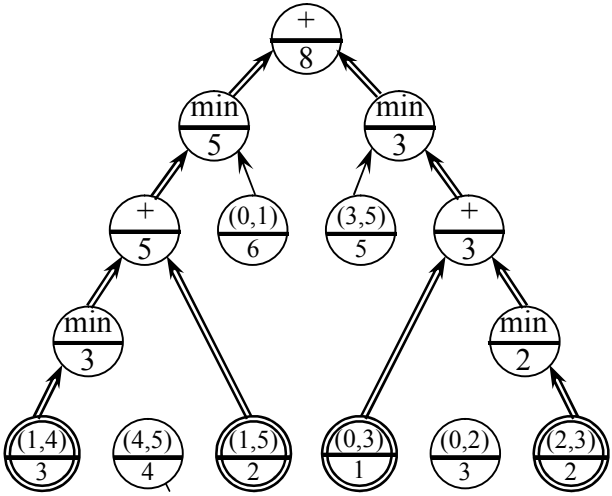

Рис. 4. Сетевое представление задачи о максимальном потоке

Числа в нижних половинах вершин равны пропускным способностям соответствующих дуг, а в верхних половинах указаны операции, которые проводятся над числами вершин нижних уровней для определения величины максимального потока. Операции взятия минимума выполняются для последовательного множества дуг, а операция суммирования для параллельного множества дуг. Число в корневой вершине дерева равно величине максимального потока.

Двигаясь сверху вниз, отмечаем двойными линиями все дуги, заходящие в вершину со знаком суммирования, и отмечаем двойной линией дугу, имеющую минимальную пропускную способность среди дуг, заходящих в вершину с операцией взятия минимума (в случае равенства пропускных способностей для нескольких вершин берем любую из них). В результате определяется разрез, имеющий минимальную пропускную способность (дуги разреза выделены двойными линиями на рис. 3 и двойными кружками на рис. 4). Заметим, что при движении сверху вниз одновременно определяются и потоки по дугам.

Рассмотрим произвольную сеть. Произвольную сеть всегда можно преобразовать в агрегируемую сеть путем разделения ряда вершин на несколько вершин. Так, если в сети (рис. 5) вершину 4 разделить на две вершины 4 и 4′, то получим агрегируемую сеть, приведенную на рис. 6.

Поскольку вместо одной дуги (4,5) мы получили две дуги (4,5) и (4′,5), то разделим пропускную способность $C_{45} = 5$ на две части произвольным образом. Возьмем, например, $C_{45} = 4$, $C_{4'5} = 1$. Определим поток максимальной величины и соответственно разрез минимальной пропускной способности в полученной сети, применяя описанный выше алгоритм для агрегируемых сетей.

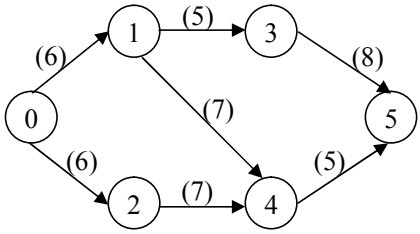

Рис. 5. Исходная сеть

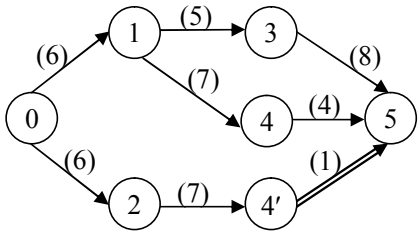

Рис. 6. Преобразованная агрегируемая сеть

Нетрудно видеть, что величина потока равна 7, а в разрез заходят дуги (0,1) и (4′,5). Имеет место

***Теорема* 2.** Величина максимального потока в агрегируемой сети меньше или равна величине максимального потока в исходной сети.

Доказательство следует из очевидного факта, что любому потоку в агрегируемой сети однозначно соответствует некоторый поток в исходной сети.

***Теорема* 3** (двойственности). Существует разбиение пропускных способностей дуг, исходящих из разделенных вершин, такое, что величина максимального потока в агрегируемой сети равна величине максимального потока в исходной сети.

Следствие. Если (i, j) – дуга, заходящая в разрез минимальной пропускной способности в исходной сети, то все дуги $(i_s j)$ также заходят в разрез минимальной пропускной способности в агрегируемой сети.

Из теоремы и следствия мы получаем необходимые и достаточные условия оптимальности потока в агрегируемой сети: все дуги, исходящие из разделенной вершины, либо заходят в разрез минимальной пропускной способности, либо ни одна из них не заходит в разрез минимальной пропускной способности.

Рассмотрим пример сети (рис. 6). В разрез минимальной пропускной способности заходит дуга (4′,5), но не заходит дуга (4,5). Поэтому поток можно увеличить, если «перекинуть» часть пропускной способности дуги (4,5) на дугу (4′,5). Увеличим $C_{4',5}$ на 3 единицы, уменьшив $C_{4,5}$ на 3 единицы (рис. 7).

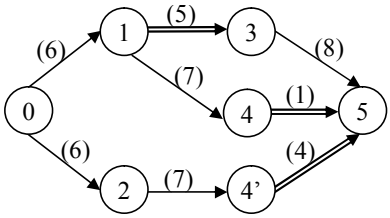

Рис. 7. Увеличение потока за счет переброски части пропускной способности дуги

Получаем разрез минимальной пропускной способности с дугами (1,3), (4,5) и (4’,5), заходящими в разрез.

Поскольку обе дуги (4,5) и (4′,5) заходят в разрез, то условия оптимальности выполнены. Следовательно, поток максимальной величины $\varphi_0 = 10$ получен.

5. Оптимальное размещение работ между подразделениями организации

Пусть в организации имеется ℓ подразделений, располагающих мощностями ресурсов одного вида. Обозначим Q_i объем работ, который может выполнить *i*-е подразделение; W_i – объем *i*-й работы, $i = \overline{1,n}$. Требуется распределить работы между подразделениями так, чтобы загрузка подразделений (или их перегрузка) была максимально равномерной. Обозначим $x_{ij} = 1$ если *i*-ая работа выполняется подразделением *j*, $x_{ij} = 0$ в противном случае. Тогда уровень загрузки (перегрузки) подразделения *i* можно оценить величиной

(14) $F_i = \sum_j w_i x_{ij} - Q_i$.

Задача заключается в распределении работ по подразделениям так, чтобы минимизировать

(15) $\max_i \left(\sum_j w_i x_{ij} - Q_i \right)$.

Рассмотрим сначала частный случай, когда $Q_i = Q$ для всех *i*. В этом случае задача сводится к классической «задаче о камнях».

Дадим постановку «задачи о камнях». Имеется *n* «камней» разного веса. Требуется разбить их на *m* групп 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 1-го камня, $x_{ij} = 1$, если камень i попал в j-ую кучку (группу), $x_{ij} = 0$ в противном случае. Суммарный вес камней в j-ой 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 (19):

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

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

Сначала получим сетевое представление задачи 2. Оно представлено на рис. 8 для случая $n = 3$, $m = 2$.

Поскольку структура сетевого 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

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

Рассмотрим следующие две задачи.

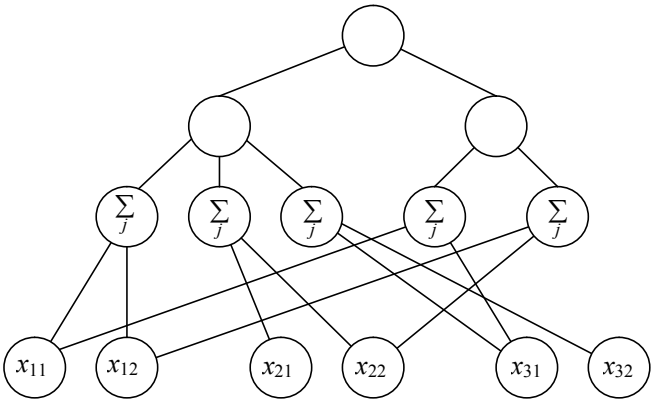

Рис. 8. Пример сетевого представления задачи 2

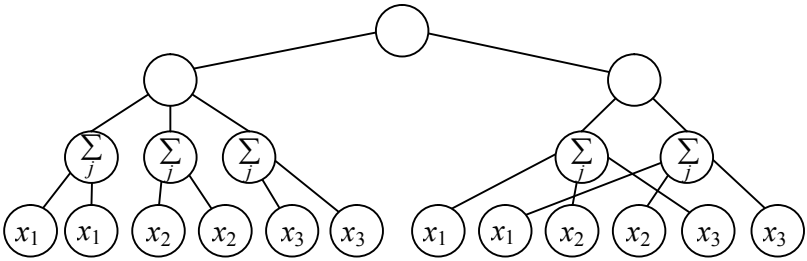

Рис. 9. Преобразованная структура

***Задача* 3.** Определить x_{ij} так, чтобы максимизировать

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

при ограничениях (18).

***Задача* 4.** 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

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

Во-вторых, решение первой задачи очевидно:
(25) $S_m(u) = \sum_i y_i$

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) и (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 $Z = \max_j (M_j - Y_j)$.

Заметим, что при заданных $\{y_i\}$ Z определяет оптимальное решение каждой из m вторых задач. Оценка (24) при этом равна
(28) $F(y) = mZ + \sum_i y_i$.
где $y_i \ge 0$ удовлетворяют неравенствам
(29) $\sum_{i \in Q_j} y_i + Z \ge M_j$, $j = \overline{1,N}$.
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 (28) при ограничениях (29). 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

(30) $Y(Z) = \sum_i y_i$.

при ограничениях (29), где $j = \overline{1,k}$. Двойственная задача имеет вид: определить $u_j \geq 0,\ j = \overline{1,k}$, максимизирующие

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

при ограничениях

$$\sum_{j \in R_i} u_j \leq 1, \quad i = \overline{1,n}$$

где R_i - множество номеров Q_j, содержащих камень i.

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

(31) $Y_0(Z) + mZ \rightarrow \min$.

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 $\Phi_{max}(T_k) \geq A$. Величина T_k является нижней оценкой для задачи 1. Далее можно применить метод ветвей и границ на основе полученной оценки.

6. Распределение ресурсов с учетом времени их перемещения между работами проекта

При решении задач распределения ресурсов как правило не учитываются времена перемещения ресурсов с работы на работу. Однако в ряде случаев времена перемещения ресурсов сравнимы с продолжительностью работ и пренебрегать ими нельзя. Рассмотрим, например, задачу ремонта мостов или участков дорог. Ремонт производится одной бригадой. Необходимо определить очередность ремонта мостов (участков дорог) так, чтобы продолжительность ремонта была минимальной. Задача

сводится к задаче коммивояжера. Дадим ее постановку и решим на основе метода сетевого программирования [5].

Задан $(n + 1)$-вершинный граф с длинами дуг l_{ij}, которые интерпретируются как расстояние между городами i и j. Требуется найти кратчайший гамильтонов контур (маршрут коммивояжера). Обозначим $x_{ij} = 1$, если дуга (i, j) входит в маршрут, $x_{ij} = 0$ в обратном случае. Тогда задача сводится к минимизации

(32) $L(x) = \sum_{i,j} l_{ij} x_{ij}$

при ограничениях

(33) $\sum_{i} x_{ij} = 1 \quad \forall j$

(34) $\sum_{j} x_{ij} = 1 \quad \forall i$

(35) $u_i - u_j + n x_{ij} \le n - 1$,

где $u_i \ge 0$; $i, j = 1, 2, \ldots, n$; $i \ne j$. Условие (35) гарантирует получение гамильтонова контура [4].

Рассмотрим симметричную задачу коммивояжера, т. е. $l_{ij} = l_{ji} \; \forall \; i, j$.

Разобьем ограничения задачи на две группы. В одну группу включим ограничения (33) и (35), а во вторую – (34) и (35) Соответственно разделим на две части коэффициенты l_{ij}, т. е. представим l_{ij} в виде

(36) $l_{ij} = p_{ij} + q_{ij}$, $\forall i, j$.

Рассмотрим две оценочные задачи.

***Задача* 5.** Определить $\{x_{ij}\}$, удовлетворяющие ограничениям (33) и (35) и минимизирующие

(37) $P(x) = \sum_{ij} p_{ij} x_{ij}$.

***Задача* 6.** Определить $\{x_{ij}\}$, удовлетворяющие ограничениям (34) и (35) и минимизирующие

(38) $Q(x) = \sum_{ij} q_{ij} x_{ij}$.

Обозначим $L_1(p)$ ($L_2(q)$) значение целевой функции в оптимальном решении первой (второй) задачи.

В соответствии с основной теоремой [3], сумма

(39) $L_1(P) + L_2(Q) = L(P, Q)$

дает оценку снизу для исходной задачи коммивояжера. Таким образом, двойственная задача заключается в определении p и q, удовлетворяющих (36) и максимизирующих (39).

***Теорема* 4.** Двойственная задача является задачей выпуклого программирования.

РЕШЕНИЕ ОЦЕНОЧНЫХ ЗАДАЧ ДЛЯ СИММЕТРИЧНОЙ ЗАДАЧИ КОММИВОЯЖЕРА

Метод решения оценочных задач для симметричных матриц основан на понятии i-дерева [6].

Выбираем некоторую вершину i. Для подграфа без вершины i строим кратчайшее i-дерево T_i, т. е. дерево кратчайшей длины $l(T_i)$. Для построения кратчайшего i-дерева T_i построим кратчайшее дерево на вершинах графа без вершины i и добавим к нему два кратчайших ребра, инцидентных вершине i [6].

Оценка снизу для каждой задачи определяется i-деревом, для которого $l(T_i)$ максимальна.

***Пример* 3.** Рассмотрим граф, рис. 10.

Оценка снизу определяется любым i-деревом и равна $l(T_i) = 9$, $i = \overline{1,6}$. Разобьем граф на 2 (рис. 11). Имеем для первого графа $l(T_3) = 9$, а для второго $l(T_2) = 9$. Оценка снизу равна $l(T_3) + l(T_2) = 18$, т. е. в два раза лучше. Более того, она является достижимой. Оптимальный маршрут – (1, 2, 6, 5, 4, 3, 1).

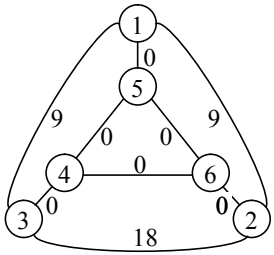

Рис. 10. Граф для примера 3

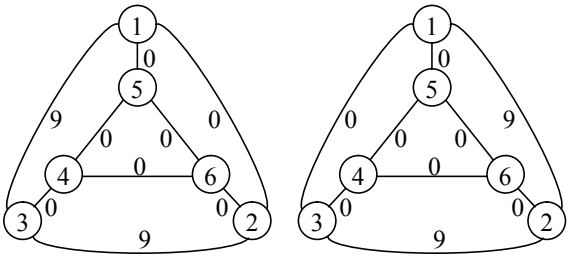

Рис. 11. Разбиение на два графа

7. Оптимизация программ по стоимости

Рассмотрим задачу формирования программы развития региона (либо предприятия, холдинга, корпорации), обеспечивающей требуемое значение комплексной оценки с минимальными затратами. Примем, что задана процедура формирования комплексной оценки программы. Программа оценивается по m критериям. Обозначим δ_{ij} минимальное (граничное) значение i-го критерия, которому соответствует оценка j ($j = 1, 2, 3, 4$). Таким образом, если значение критерия y_j лежит в полуинтервале

$$\delta_{ij} \le y_i < \delta_{ij+1},$$

то оценка по соответствующему направлению равна j.

Имеется n проектов – претендентов на участие в программе. Каждый проект характеризуется затратами c_k и показателями эффекта α_{ki}, которые определяют вклад k-го проекта в i-ый критерий. Обозначим $x_k = 1$, если k-ый проект включен в программу, $x_k = 0$ в противном случае. Предполагая, что эффекты суммируются, получаем, что увеличение i-го критерия в результате реализации программы составит

(40) $\Delta y_i = \sum_k \alpha_{ki} x_r$,

а соответствующая оценка по i-му направлению равна

(41) $j_i = \theta(y_i) = \theta(y_i^0 + \Delta y_i)$

где y_i^0 – начальное значение *i*-го критерия; θ – преобразование численного значения критерия в дискретную (качественную) шкалу. Суммарные затраты на реализацию программы составят

(42) $C(x)=\sum_i c_i x_i$.

Обозначим $K(J)$ – комплексную оценку программы при оценках направлений

$J=(j_1, j_2 ..., j_m)$

Задача. Определить множество проектов, обеспечивающих $K(J) = K_T$ при минимальных затратах (42). Задача относится к сложным задачам дискретной оптимизации.

Назовем многоцелевыми проекты, которые дают эффект в несколько направлений. Обозначим Ф – множество многоцелевых проектов; n_i – число направлений, в которые дает эффект *i*-ый проект. Разделим каждый проект *i* на m_i подпроектов с затратами u_{ij} такими, что

(43) $\sum_j u_{ij} = c_i$

и, соответственно, эффектами a_{ij}. Получаем ситуацию, когда для каждого направления существует свое множество проектов и подпроектов. В этом случае решение задачи становится значительно проще.

Обозначим R_i – множество одноцелевых проектов для *i*-го направления, $Q_i \subset Q$ – множество многоцелевых проектов, дающих вклад в *i*-е направление.

1 *шаг*. Решаем *m* задач о ранце для каждого критерия: минимизировать

(44) $C_i(x,u)=\sum c_k x_k$

при ограничении

(45) $\sum_{k\in R_i} a_k x_k + \sum_{k\in Q_i} u_{ki} x_k \geq \delta_{i4} - y_i^0 = \Delta_{i4}$

Как известно, решение задачи о ранце при правой части ограничения Δ_{i4} дает оптимальные решения и для всех меньших значений правой части, т. е. для Δ_{i3} Δ_{i2} и Δ_{i1}. Обозначим S_{ij}-

минимальные затраты, требуемые для достижения оценки J по i-ому критерию.

2 *шаг.* Поскольку структура формирования комплексной оценки является деревом, то решаем задачу, последовательно решая для каждой матрицы процедуры комплексного оценивания задачу с двумя переменными.

Согласно теореме 1, полученная величина затрат $S(u)$ является нижней оценкой для исходной задачи.

Двойственная задача. Определить $u = \{u_{ij}\}$ так, чтобы максимизировать $S(u)$ при ограничениях (45).

***Теорема* 5.** Двойственная задача является задачей выпуклого программирования.

Полученную оценку можно применить в методе ветвей и границ.

8. Заключение

Анализируя рассмотренные задачи можно заметить универсальность схемы применения метода сетевого программирования, включающей четыре этапа:

1. Построение сетевых представлений для целевой функции и ограничений с одинаковой структурой.
2. Выбор начальных значений двойственных переменных.
3. Решение оценочных задач.
4. Целенаправленное изменение значений двойственных переменных.

В настоящее время на основе этой схемы предложены алгоритмы решения задач размещения объектов обслуживания, задача оптимизации сетей по стоимости, задача оптимального управления объектами недвижимости, задач слияния предприятий и ряд других.

На повестке дня стоит задача разработки программного комплекса для решения широкого класса задач дискретной оптимизации на основе метода сетевого программирования.

Литература

1. АЛФЕРОВ В.И., БАРКАЛОВ С.А., КУРОЧКА П.Н. *Управление проектами в дорожном строительстве.* – Воронеж: «Научная книга», 2009. – 340 с.
2. БАРКАЛОВ С.А. *Теория систем и системный анализ:* учебное пособие. – Воронеж: «Научная книга», 2009. – 626 с.
3. БУРКОВ В.Н., БУРКОВА И.В. *Задачи дихотомической оптимизации*. – М.: Радио и связь, 2003. – 156 с.
4. БУРКОВ В.Н., БУРКОВА И.В., ОВЧИННИКОВА Т.И., ПОПОК М.В. *Метод сетевого программирования* // Проблемы управления. – 2005. – №3. - С. 25–27.
5. БУРКОВА И.В. *Метод сетевого программирования в симметричной задаче коммивояжера*. // Проблемы управления. – 2008. – №4. - С. 7–10.
6. СИГАЛ И.Х., ИВАНОВА А.П. *Введение в прикладное дискретное программирование*. – М.: ФИЗМАТЛИТ, 2007. – 304 с.

NETWORK PROGRAMMING IN PROJECT MANAGEMENT

Vladimir N. Burkov, Institute of Control Sciences of RAS, Moscow, Doctor of Science, professor (vlab17@bk.ru).
Irina V. Burkova, Institute of Control Sciences of RAS, Moscow, Cand. Sci. (irbur27@mail.ru).

Abstract: *The method of network programming was developed to give exact or approximate solutions for multi-extremal* (*in particular, discrete*) *optimization problems. The idea of the method is based on reduction of the problem in hand to a superposition of simpler problems. The scheme of reduction is conveniently represented in the form of a network* (*the, so called, network representation*), *with nodes being the sub-problems. Simple optimization*

problems are solved at each node, while the solution at the terminal node of the network delivers the upper (*or lower*) *bound estimate for the initial problem. For the tree-shaped network representation the solution at the terminal node of the network delivers the exact solution of the initial optimization problem. This paper surveys applications of the network programming method to the several problems of project management.*

Keywords: network programming, project management, discrete optimization.

Статья представлена к публикации
членом редакционной коллегии М. В. Губко