УДК 519.854

Приближенное решение одной задачи управления поставками[1]

А.В. Еремеев, А.А. Романова, В.В. Сервах, С.С. Чаухан

Рассматривается задача оптимизации доставки продукции от поставщиков потребителям. Размер каждой открытой поставки ограничен снизу и сверху, размер потребления для каждого потребителя ограничен снизу, функции стоимости поставки линейны при ненулевых объемах поставки. Предложена вполне полиномиальная аппроксимационная схема для решения этой задачи в случае одного потребителя и исследована сложность задачи в общем случае.

Введение

В статье исследуется возможность нахождения решения с гарантированной оценкой точности для следующей задачи управления поставками. Поставщики доставляют некоторый продукт потребителям, причем количество продукции, которое может быть доставлено по открытой поставке, заключено в границах между минимальным и максимальным значениями, а стоимость, предложенная каждым поставщиком, есть линейная функция от объема поставки. Формально задача может быть записана в следующем виде:

найти минимум функции

$$\sum_{i=1}^{n}\sum_{j=1}^{m} k_{ij}(x_{ij}), \tag{1}$$

при условиях

$$\sum_{i=1}^{n} x_{ij} \geq A_j, \quad 1 \leq j \leq m, \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} \leq M_i, \quad 1 \leq i \leq n, \tag{3}$$

$$x_{ij} \in \{0\} \cup [m_{ij}, M_i], \quad 1 \leq i \leq n, \quad 1 \leq j \leq m, \tag{4}$$

где n – число поставщиков; m – число потребителей; A_j – минимальное количество продукта, требуемое потребителю j; m_{ij} – минимальное количество продукта, которое поставщик i готов доставить потребителю j; M_i – максимальное количество продукта, которое поставщик i может доставить. Переменная x_{ij} обозначает размер поставки от поставщика i потребителю j.

[1] Исследование поддержано грантами INTAS 00-217, 03-51-5501 и РГНФ 04-02-00238а.

Стоимость доставки

$$k_{ij}(x_{ij}) = \begin{cases} 0, & \text{если } x_{ij} = 0; \\ a_{ij} + b_{ij}x_{ij}, & \text{если } x_{ij} > 0. \end{cases}$$

Параметры $a_{ij}, b_{ij}, m_{ij}, M_i, A_j$ предполагаются целочисленными и неотрицательными при всех i, j.

Стоимость решения $x = (x_{ij})$, заданную выражением (1), будем обозначать через $f(x)$.

Эта задача является ослабленной версией задачи управления поставками с вогнутой целевой функцией, рассмотренной в [3, 5]: в них в условии (2) требуется равенство, а функции стоимости неотрицательные и вогнутые. Нахождение любого допустимого решения задачи в постановке из [5] является NP-трудной задачей даже при одном потребителе, хотя существует псевдополиномиальный точный алгоритм ее решения [3]. Как будет показано ниже, ослабленная версия задачи представляет больше возможностей для приближения и в связи с этим более интересна.

Задача (1)–(4) является NP-трудной, так как она содержит задачу о рюкзаке как частный случай. Кроме того, сведение задачи разбиения показывает, что задача нахождения допустимого решения, удовлетворяющего ограничениям (2)–(4), является NP-трудной даже в случае $m = 2$. Действительно, рассмотрим задачу разбиения как задачу распознавания: можно ли разбить множество натуральных чисел $\{p_1, p_2, \ldots, p_N\}$ на два подмножества с равными суммами? Если сумма $\sum\limits_{i=1}^{N} p_i$ нечетна, то ответ, очевидно, отрицательный. В противном случае, положим $m = 2$, $n = N$, $m_{i1} = m_{i2} = M_i = p_i$, $1 \le i \le N$ и $A_1 = A_2 = \frac{1}{2}\sum\limits_{i=1}^{N} p_i$. При этом система (2)–(4) разрешима тогда и только тогда, когда требуемое разбиение существует.

Под *ρ-приближенным алгоритмом* будем понимать алгоритм, находящий приближенное решение, стоимость которого не более чем в ρ раз превышает стоимость оптимального решения (если задача разрешима). Соответствующее решение будем называть *ρ-приближенным решением.*

Под *вполне полиномиальной аппроксимационной схемой* (FPTAS) будем понимать семейство $(1+\varepsilon)$-приближенных алгоритмов при всевозможных $\varepsilon > 0$ с временной сложностью, полиномиально зависящей от длины входа задачи и от $1/\varepsilon$.

Основными результатами статьи являются следующие (теоремы 1 и 2).

Теорема 1 *В случае $m = 1$ существует вполне полиномиальная аппроксимационная схема решения задачи* (1)–(4) *с временной сложностью $O(n^3/\varepsilon + n^2)$.*

Теорема 2 *Задача* (1)–(4) *и задача нахождения $p(n, m)$-приближенного решения при любом полиноме $p(n, m)$ с положительными коэффициентами являются NP-трудными в сильном смысле.*

Здесь и далее в оценках трудоемкости предложенных алгоритмов подразумевается вычислительная сложность машины RAM с произвольным доступом к памяти, когда стандартные арифметические операции имеют константную длительность (см., например, [1]).

Все арифметические операции выполняются над числами с длиной записи, ограниченной полиномом от длины входа в модели машины Тьюринга. Поэтому предложенные алгоритмы имеют полиномиальную временную сложность и в этой модели вычислений.

Для упрощения обозначений задачу (1)–(4) при $m = 1$ запишем в более компактном виде:

найти минимум функции

$$\sum_{i=1}^{n} k_i(x_i), \tag{5}$$

при условиях

$$\sum_{i=1}^{n} x_i \geq A, \tag{6}$$

$$x_i \in \{0\} \cup [m_i, M_i], \quad 1 \leq i \leq n. \tag{7}$$

Стоимость доставки

$$k_i(x_i) = \begin{cases} 0, & \text{если } x_i = 0; \\ a_i + b_i x_i, & \text{если } x_i > 0, \end{cases} \tag{8}$$

где $a_i \geq 0$ и $b_i \geq 0$, $1 \leq i \leq n$.

Чтобы получить FPTAS для решения задачи (5)–(8), сначала построим эффективный жадный 2-приближенный алгоритм, а затем псевдополиномиальный алгоритм, основанный на динамическом программировании. Этот подход аналогичен подходу, примененному к задаче о рюкзаке на минимум [2]. Последний параграф содержит доказательство теоремы 2.

В [4] для решения данной задачи была также построена вполне полиномиальная аппроксимационная схема, которая годится для более общего класса целевых функций и основана на подходе, подробно изложенном в [7]. В применении к задаче (5)–(8) эта схема имеет временную сложность $O((n/\varepsilon)^2 \log_2(nk_{\max}) \log_2(k_{\max}))$, где $k_{\max} = \max\limits_{i=1,\dots,n} k_i(M_i)$. Единственное дополнительное условие состоит в том, чтобы $a_i + b_i,\ i = 1, 2, \dots, n$ были отделены от нуля положительной константой, общей для всех индивидуальных задач. Как видно, по величине $1/\varepsilon$ схема [4] имеет бóльшую трудоемкость по сравнению со схемой, рассмотренной в теореме 1, и время ее работы зависит от числового входного параметра $k_{\max}$ даже в модели вычислений RAM.

1 2-приближенный жадный алгоритм

В данном параграфе предлагается жадный алгоритм решения более общей задачи вида (5)–(7), в которой стоимости доставки $k_i(x_i),\ 1 \leq i \leq n$ заданы функциями

$$k_i(x_i) = \begin{cases} 0, & \text{если } x_i = 0; \\ a_i + g_i(x_i), & \text{если } x_i > 0, \end{cases} \tag{9}$$

где $a_i \geq 0$ при любом i, а $g_i(x_i)$ – вогнутая неотрицательная неубывающая функция при $x_i \in [0, M_i]$. Обозначим через $y = (y_j)$ приближенное решение, найденное жадным алгоритмом; L – текущее количество продукта, которое еще необходимо доставить потребителю.

Жадный алгоритм.

Шаг 1. Положить $y_k := 0,\ k = 1, 2, \ldots, n$.

Шаг 2. Положить $L := A - \sum_{k=1}^{n} y_k$ и выбрать такое $j \in \{1, 2, \ldots, n\}$, что $y_j = 0$ и

$$\frac{k_j(\min\{\max\{L, m_j\}, M_j\})}{\min\{L, M_j\}} \leq \frac{k_s(\min\{\max\{L, m_s\}, M_s\})}{\min\{L, M_s\}} \tag{10}$$

для всех s таких, что $y_s = 0$.

Если $y_j > 0$ для всех j, то **Конец**

Шаг 3. Положить $y_j := \min\{\max\{L, m_j\}, M_j\}$.

Шаг 4. Если $L - y_j > 0$, перейти на шаг 2.

Конец.

Если после остановки алгоритма выполняется неравенство $\sum_{k=1}^{n} y_k < A$, то задача не имеет допустимых решений. Пусть T_g – верхняя граница временной сложности вычисления функции $g_i(x_i)$ при всех i и x_i. Тогда, как легко видеть, для выполнения алгоритма требуется $O(n^2 T_g)$ арифметических операций.

Теорема 3 *Для задачи* (5)–(7), (9) *жадный алгоритм является* 2*-приближенным алгоритмом.*

Доказательство. Заметим, что на шаге 3 переменная y_j принимает значение M_j на всех итерациях, кроме последней; на ней y_j равна L или m_j (здесь и ниже под L понимается значение пременной L, которое она принимает по окончании выполнения алгоритма). Не ограничивая общности, после завершения работы алгоритма перенумеруем координаты решения так, что при некотором i выполнено равенство $y_j = M_j$ при всех $j \leq i$; $y_{i+1} < M_{i+1}$ и $y_j = 0$ при всех $j > i + 1$.

Пусть f^* – стоимость оптимального решения задачи (5)–(7), (9). Положим

$$h(x) = \sum_{j=1}^{n} \frac{(a_j + g_j(M_j))x_j}{M_j}$$

и рассмотрим задачу минимизации функции $h(x)$ на множестве, задаваемом ограничениями (6)–(7). Рассматривая оптимум $\hat{x}$ линейной релаксации данной задачи, т. е. задачи, в которой (7) заменено на $0 \leq x_j \leq M_j,\ 1 \leq j \leq n$, заключаем, что

$$h(\hat{x}) \geq \sum_{j=1}^{i} a_j + g_j(M_j) = \sum_{j=1}^{i} k_j(M_j).$$

По определению $h(x)$ имеем $h(x) \leq f(x)$ для любого x, удовлетворяющего условиям (6)–(7). Таким образом,

$$f^* \geq \sum_{j=1}^{i} k_j(M_j).$$

Следовательно, если $y_{i+1} = 0$ (т. е. $L = M_i$ на последней итерации жадного алгоритма), то y – оптимальное решение.

Теперь убедимся в том, что если $y_{i+1} > 0$, то $f^* \geq k_{i+1}(y_{i+1})$. Действительно, предположим, что существует допустимое решение z такое, что

$$\sum_{j=1}^{n} k_j(z_j) < k_{i+1}(y_{i+1}). \tag{11}$$

Так как $L = A - \sum\limits_{k=1}^{i} M_k$, то

$$\sum_{s>i} z_s \geq A - \sum_{j \leq i} z_j \geq L. \tag{12}$$

Умножая обе части неравенства (11) на $L/k_{i+1}(y_{i+1})$ и убирая первые i слагаемых, получаем

$$\sum_{s>i} \frac{Lk_s(z_s)}{k_{i+1}(y_{i+1})} < L. \tag{13}$$

Жадное правило (10), примененное на последней итерации, при всех $s > i$ обеспечивает неравенство

$$\frac{k_{i+1}(y_{i+1})}{L} = \frac{k_{i+1}(y_{i+1})}{\min\{L, M_{i+1}\}} \leq \frac{k_s(\min\{\max\{L, m_s\}, M_s\})}{\min\{L, M_s\}}. \tag{14}$$

Таким образом, из (13) следует неравенство

$$\sum_{s>i} \frac{\min\{L, M_s\} k_s(z_s)}{k_s(\min\{\max\{L, m_s\}, M_s\})} < L. \tag{15}$$

Рассмотрим слагаемые этой суммы с $z_s > 0$. Если $L < m_s$, то из (14) имеем $k_s(m_s)/L \geq k_{i+1}(y_{i+1})/L$, что противоречит неравенству $k_s(z_s) < k_{i+1}(y_{i+1})$, полученному из (11). Таким образом, слагаемые с $L < m_s$ отсутствуют в (15).

Если $m_s \leq L \leq M_s$, то соответствующие слагаемые в (15) принимают вид $Lk_s(z_s)/k_s(L)$. Возможны два случая:

(a) $z_s \geq L$. Так как $k_s(x)$ неубывающая функция, то $k_s(z_s) \geq k_s(L)$. Из (14) имеем $k_s(L) \geq k_{i+1}(y_{i+1})$. Следовательно, $k_s(z_s) \geq k_{i+1}(y_{i+1})$, что противоречит (11). Поэтому слагаемые с $z_s \geq L$ также отсутствуют в (15).

(b) $z_s < L$. Из вогнутости функции k_s следует, что $Lk_s(z_s)/k_s(L) \geq z_s$.

Наконец, если $L > M_s$, слагаемые в (15) превращаются в $M_s k_s(z_s)/k_s(M_s)$, и в силу вогнутости функции k_s имеем $M_s k_s(z_s)/k_s(M_s) \geq z_s$. Следовательно,

$$\sum_{s>i} z_s \leq \sum_{s>i} \frac{\min\{L, M_s\} k_s(z_s)}{k_s(\min\{\max\{L, m_s\}, M_s\})}, \tag{16}$$

и из (15) и (16) получаем неравенство $\sum\limits_{s>i} z_s < L$, что противоречит (12). Таким образом, $f^* \geq k_{i+1}(y_{i+1})$ и

$$2f^* \geq \sum_{j=1}^{i} k_j(M_j) + k_{i+1}(y_{i+1}).$$

Теорема 3 доказана.

2 Псевдополиномиальный точный алгоритм

Пусть $\mathbf{Z}$ – множество целых чисел.

Лемма 1 *Разрешимая задача* (5)–(8) *имеет оптимальное решение* $z \in \mathbf{Z}^n$*, где* $z_i \in \{0, m_i, M_i\}$ *при всех* $1 \le i \le n$ *кроме, может быть, одного.*

Доказательство. Сначала установим существование решения y со свойствами, указанными в формулировке леммы, кроме свойства целочисленности. Предположим, что x – некоторое оптимальное решение задачи (5)–(8), и существует такая пара $i, j,\ 1 \le i, j \le n$, что $m_i < x_i < M_i$, $m_j < x_j < M_j$ и $b_i \le b_j$. Положим $\delta = \min\{M_i - x_i, x_j - m_j\}$, $y_i = x_i + \delta$ и $y_j = x_j - \delta$; остальные же координаты решений y и x совпадают. Легко видеть, что y также оптимальное решение задачи (5)–(8), и $y_i = M_i$ либо $y_j = m_j$. Так как все параметры задачи целочисленные, то существует по крайней мере одна нецелочисленная координата в решении y. Пусть это будет y_i. Тогда, полагая $z = (y_1, \ldots, y_{i-1}, \lfloor y_i \rfloor, y_{i+1}, \ldots, y_n)$, снова получаем оптимальное решение, так как $A \in \mathbf{Z}$ и $k_i(x_i)$ – неубывающая функция. Лемма 1 доказана.

Рассмотрим задачу (5)–(8) с дополнительным условием:

$$x_i \in \mathbf{Z} \text{ при любом } i,\ 1 \le i \le n. \tag{17}$$

В силу леммы 1 условие (17) не исключает оптимального решения в (5)–(8). В следующем параграфе для построения FPTAS будет удобно использовать следующую переформулировку задачи (5)–(8), (17):

найти минимум функции

$$\sum_{i=1}^{n} (\alpha_i u_i + \beta_i v_i), \tag{18}$$

при условии

$$\sum_{i=1}^{n} (m_i u_i + (M_i - m_i) v_i) \ge A, \tag{19}$$

$$v_i \le u_i,\ i = 1, 2, \ldots, n, \tag{20}$$

$$u_i \in \{0, 1\}, v_i \in \left\{0, \frac{1}{M_i - m_i}, \frac{2}{M_i - m_i}, \ldots, 1\right\},\ 1 \le i \le n. \tag{21}$$

Параметры задачи предполагаются целочисленными и неотрицательными. Заметим, что теперь значения всех переменных заключены между 0 и 1, и исходная задача (5)–(8), (17) принимает форму (18)–(21) при

$$\alpha_i = a_i + b_i m_i,\ \beta_i = b_i(M_i - m_i), x_i = u_i m_i + (M_i - m_i) v_i,\ 1 \le i \le n.$$

Ради краткости положим

$$\gamma(u, v) = \sum_{i=1}^{n} (\alpha_i u_i + \beta_i v_i)$$

и

$$w(u,v)=\sum_{i=1}^{n}(m_iu_i+(M_i-m_i)v_i).$$

Целесообразно использовать также вспомогательную задачу, отличающуюся от задачи (18)–(21) лишь тем, что условие (21) надо заменить на условие

$$u_i\in\{0,1\}, v_i\in\{0,1\},\ 1\le i\le n, \tag{22}$$

т. е. размер поставки может принимать только крайние значения. Пусть U – верхняя граница целевой функции задачи (18)–(20), (22). Для каждого $k=1,2,\ldots,n$ и каждого $c=0,1\ldots,U$ определим множество

$$S(k,c)=\operatorname*{argmax}_{u,v\in\{0,1\}^n}\left\{w(u,v)\mid\gamma(u,v)=c,\ u_i=v_i=0\ \text{ при любом } i>k\right\}.$$

Для любой пары (k,c) определим функцию $\varphi(k,c)$. Если $S(k,c)\neq\emptyset$, то положим $\varphi(k,c)=w(u,v)$, где $(u,v)\in S(k,c)$. Если же $S(k,c)=\emptyset$, то $\varphi(k,c)=-\infty$. Тогда начальный шаг динамического программирования определяется следующим образом:

$$\varphi(1,c)=\begin{cases}0, & \text{если } c=0;\\ m_1, & \text{если } c=\alpha_1;\\ M_1, & \text{если } c=\alpha_1+\beta_1;\\ -\infty & \text{в противном случае.}\end{cases}$$

Каждый шаг $k=2,3,\ldots,n$ выполняется с использованием рекуррентного соотношения:

$$\varphi(k,c)=\begin{cases}\varphi(k-1,c), & \text{если } c<\alpha_k;\\ \max\{\varphi(k-1,c),m_k+\varphi(k-1,c-\alpha_k)\}, & \\ & \text{если } c\ge\alpha_k,\ c<\alpha_k+\beta_k;\\ \max\{\varphi(k-1,c),m_k+\varphi(k-1,c-\alpha_k),M_k+\varphi(k-1,c-\alpha_k-\beta_k)\}, & \\ & \text{если } c\ge\alpha_k+\beta_k.\end{cases}$$

Следовательно, если γ' – оптимальное значение целевой функции задачи (18)–(20), (22) и $\gamma'\le U$, то $\gamma'=\min\{c|\varphi(n,c)\ge A\}$ и множество $S(n,\gamma')$ содержит оптимальное решение (или решения) этой задачи.

Как следует из леммы 1, для решения задачи (18)–(21) достаточно искать только одну переменную v_j в множестве $\{0,\frac{1}{M_j-m_j},\frac{2}{M_j-m_j},\ldots,1\}$ с $u_j=1$, в то время как остальные переменные оптимизируются с помощью описанной выше схемы динамического программирования. Такая задача решается при каждом $j=1,2,\ldots,n$, и из полученных решений выбирается лучшее.

Предположим, нам дана верхняя граница $UB\ge\gamma^*$, где γ^* – стоимость оптимального решения задачи (18)–(21). Для решения этой задачи мы предлагаем следующий алгоритм.

Алгоритм 1.

Пусть $\tilde{\gamma}=UB$.

Для $j=1$ до n:

1. Применить метод динамического программирования с $U = UB$ к задаче (18)–(20), (22), где $m_j = M_j = 0$.

2. Запомнить значения $\varphi_j(n, c),\ 0 \le c \le UB$ и представителей $(u^{(0)}, v^{(0)}), \ldots, (u^{(UB)}, v^{(UB)})$ множеств решений $S(n, 0), \ldots, S(n, UB)$. Если для некоторого c множество $S(n, c)$ пусто, то $(u^{(c)}, v^{(c)})$ произвольно.

3. Для $c = 0, 1, \ldots, UB$ выполнить:

   Если $A \le \varphi_j(n, c)$ и $c \le \tilde{\gamma}$, то обновить $\tilde{\gamma} := c$, $\tilde{u} := u^{(c)}$, $\tilde{v} := v^{(c)}$.

   Если $\varphi_j(n, c) < A < m_j + \varphi_j(n, c)$ и $c + \alpha_j \le \tilde{\gamma}$, то обновить $\tilde{\gamma} := c + \alpha_j$; $\tilde{u}_j := 1,\ \tilde{v}_j := 0;\ \ \tilde{u}_i := u_i^{(c)},\ \tilde{v}_i := v_i^{(c)}, i \ne j.$

   Если $\varphi_j(n, c) + m_j \le A \le \varphi_j(n, c) + M_j$ и $c + \alpha_j + \beta_j \frac{A - m_j - \varphi_j(n,c)}{M_j - m_j} \le \tilde{\gamma}$, то обновить

$$\tilde{\gamma} := c + \alpha_j + \beta_j \frac{A - m_j - \varphi_j(n, c)}{M_j - m_j};\ \ \tilde{u}_j := 1,\ \ \tilde{v}_j := \frac{A - m_j - \varphi_j(n, c)}{M_j - m_j};$$

$$\tilde{u}_i := u_i^{(c)}, \tilde{v}_i := v_i^{(c)}, i \ne j.$$

Результат является оптимальным решением $(\tilde{u}, \tilde{v})$.

Теорема 4 *Алгоритм* 1 *решает задачу* (18)–(21) *за* $O(n^2 UB)$ *арифметических операций.*

Доказательство. Правильность алгоритма следует из леммы 1 и свойств схемы динамического программирования, обсужденной выше (заметим, что $U = UB$ во вспомогательной задаче не исключает частичного решения, соответствующего оптимальному). Оценка временной сложности очевидна. Теорема 4 доказана.

3 Вполне полиномиальная аппроксимационная схема

Применим подход, основанный на округлении входных данных (см., например, [2, 6]), чтобы сложность процедуры динамического программирования стала полиномиальной от длины входа.

Доказательство теоремы 1. Прежде всего исключим из рассмотрения случай $f^* = 0$, так как его легко обнаружить, и в этом случае найти оптимальное решение. Поэтому предположим, что $f^* > 0$.

Взяв стоимость жадного 2-приближенного решения за верхнюю границу UB^* оптимума задачи (5)–(8), мы также получаем его нижнюю границу $LB^* = UB^*/2 \le f^*$ (заметим, что $LB^* > 0$, так как $0 < f^* \le UB^*$).

Зафиксируем $\varepsilon,\ 0 < \varepsilon < 2$ (если $\varepsilon \ge 2$, то жадное решение обеспечивает достаточную точность) и положим $\delta = \max\left\{\frac{\varepsilon LB^*}{2n}, 1\right\}$. Применим алгоритм 1 к примеру задачи (18)–(21) со следующими значениями параметров:

$$\alpha_i = \left\lceil \frac{a_i + b_i m_i}{\delta} \right\rceil, \beta_i = \left\lceil \frac{M_i - m_i}{\delta / b_i} \right\rceil,\ UB = \frac{UB^*}{\delta} + 2n,\ 1 \le i \le n. \tag{23}$$

Верхняя граница UB справедлива, так как стоимость оптимума в (18)–(21) с α_i, β_i, определенными в (23), не превышает $f^*/\delta + 2n$. Далее заметим, что если $\frac{\varepsilon LB^*}{2n} > 1$, то $UB = 4n/\varepsilon + 2n$, в противном же случае все равно имеем $UB \leq 4n/\varepsilon + 2n$. Таким образом, в силу теоремы 4 вычисления в алгоритме 1 и в жадном алгоритме в сумме требуют $O(n^3 \frac{1}{\varepsilon} + n^2)$ операций.

Заметим, что полученное решение

$$\tilde{x}_i = m_i \tilde{u}_i + (M_i - m_i)\tilde{v}_i,\ 1 \leq i \leq n$$

есть $(1+\varepsilon)$-приближенное решение задачи (5)–(8).

В самом деле, если $\delta = 1$, то $\tilde{x}$ – оптимальное решение. Если $\delta > 1$, то положим $\gamma(x) = \gamma(u(x), v(x))$, где

$$u(x)_i = \text{ sign } x_i,\ v(x)_i = \frac{x_i - m_i u(x)_i}{M_i - m_i},\ 1 \leq i \leq n,$$

т. е., $\gamma(x)$ есть стоимость решения $u(x), v(x)$ в задаче (18)–(21). Пусть x^* – оптимальное решение задачи (5)–(8). Тогда

$$\frac{f(x^*)}{\delta} = \sum_{i=1}^{n} \frac{a_i + b_i m_i}{\delta} u(x^*)_i + \sum_{i=1}^{n} \frac{b_i(M_i - m_i)}{\delta} v(x^*)_i \geq \sum_{i=1}^{n} \left\lceil \frac{a_i + b_i m_i}{\delta} \right\rceil u(x^*)_i +$$

$$+ \sum_{i=1}^{n} \left\lceil \frac{b_i(M_i - m_i)}{\delta} \right\rceil v(x^*)_i - 2n = \gamma(x^*) - 2n.$$

Таким образом, так как $(\tilde{u}, \tilde{v})$ – оптимум для (18)–(21), то $\delta\gamma(\tilde{x}) \leq \delta\gamma(x^*) \leq f^* + 2n\delta$ и

$$\frac{|f(\tilde{x}) - f^*|}{f^*} \leq \frac{\delta\gamma(\tilde{x}) - f^*}{f^*} \leq \frac{2n\delta}{LB^*} = \varepsilon.$$

Теорема 1 доказана.

4 Сложность приближенного решения в случае нескольких потребителей

Покажем, что в общем случае задача (1)–(4) для приближения с любой фиксированной точностью NP-трудна в сильном смысле. Для этого будем использовать задачу распознавания наличия вершинного покрытия требуемой мощности в графе $G = (V, E)$ с множеством вершин $V = \{1, 2, \ldots, s\}$ и множеством ребер $E = \{e_1, e_2, \ldots, e_r\}$. Подмножество $C \subseteq V$ называется *вершинным покрытием* графа G, если каждая вершина из $V \backslash C$ смежна по крайней мере с одной вершиной из C. Задача состоит в том, чтобы для заданного $K,\ K < s$ ответить на вопрос: существует ли в графе G вершинное покрытие мощности не более K? Обозначим через $\deg j$ степень вершины j.

Опишем сведение задачи о вершинном покрытии графа к задаче (1)–(4). Каждый вход задачи о вершинном покрытии представляет собой некоторый граф $G = (V, E)$. Поставим в соответствие вершинам графа G пункты потребления, а ребрам – пункты производства и введем дополнительный пункт производства с номером $r + 1$. Таким образом, имеем $n = r+1$ и $m = s$. В данном сведении открытие поставок от пункта производства с номером $r + 1$ будет соответствовать включению вершин в покрытие. Спрос каждого потребителя j положим равным степени соответствующей вершины: $A_j = \deg j$.

Для каждого поставщика i, $1 \le i \le r$ назначим $M_i = m_{ij} = 1$, $b_{ij} = 0$ при всех j, и пусть $a_{ij} = 0$ для двух его "выделенных" потребителей, соответствующих вершинам инцидентным ребру e_i в графе G. Для остальных потребителей $j \notin e_i$ полагаем $a_{ij} = \Omega$, где $\Omega > s$ – достаточно большое число. Другими словами, каждый поставщик i, $1 \le i \le r$ может отправить единицу продукции по нулевой цене только одному из своих двух "выделенных" потребителей. Для дополнительного поставщика $r + 1$ положим $m_{r+1,j} = s$, $a_{r+1,j} = 1$, $b_{r+1,j} = 0$ при всех j; $M_{r+1} = Ks$.

Далее полученную с помощью описанной сводимости задачу о поставках будем обозначать через $P(G)$.

Лемма 2 a) *Если G имеет вершинное покрытие мощности не более K, то оптимальное значение целевой функции задачи о поставках $P(G)$ не превышает K.*
b) *Если задача о поставках $P(G)$ имеет допустимое решение, стоимость которого меньше Ω, то существует вершинное покрытие графа G мощности не более K.*

Доказательство. (a) Пусть дано вершинное покрытие C графа G. Положим для каждого j размер поставки от дополнительного поставщика равным s, если $j \in C$; в противном случае $x_{r+1,j} = 0$. Тогда для того, чтобы x было допустимым решением задачи (1)–(4) достаточно для всех $i \le r$ и всех j положить $x_{ij} = 1$ при $j \in e_i$, $j \notin C$; в противном случае $x_{i,j} = 0$. Так как все поставки от поставщиков $i = 1, 2, \ldots, r$ направлены только "выделенным" потребителям и поставщик $r + 1$ обслуживает не более K клиентов, то $f(r) \le K$.

(b) Пусть x – допустимое решение задачи (1)–(4) и $f(x) < \Omega$. Обозначим $C(x) = \{j | x_{r+1,j} > 0\}$. Здесь $|C(x)| \le K$, так как $x_{r+1,j} \ge s$ для всех $j \in C(x)$, и $x_{r+1,1} + \ldots + x_{r+1,n} \le M_{r+1} = Ks$. Первые r поставщиков снабжают только "выделенных" потребителей, поэтому $C(x)$ – вершинное покрытие, так как если $x_{r+1,u} = 0$ и $x_{r+1,v} = 0$ для какого-либо ребра $e_i = uv$, то

$$\sum_{i=1}^{n} x_{iu} + \sum_{i=1}^{n} x_{iv} \le \deg u + \deg v - 1 < A_u + A_v,$$

что противоречит (2). Лемма 2 доказана.

Доказательство теоремы 2. Пусть дан граф G и $\Omega = p(r, s)s$. При этом в соответствующей индивидуальной задаче о поставках $P(G)$ все численные параметры ограничены некоторым полиномом от длины записи задачи о вершинном покрытии графа G.

Пусть x^* – оптимальное решение задачи (1)–(4), а x' – ее $p(r, s)$-приближенное решение. С одной стороны, если существует вершинное покрытие мощности не более K, то по утверждению (a) леммы 2 имеем $f(x^*) \le K$, следовательно, $f(x') \le p(r, s)K < \Omega$. С другой стороны, если $f(x') \leq p(r,s)K$, то $f(x^*) < \Omega$ и покрытие мощности не более K существует по утверждению (b) леммы 2.

Таким образом, наличие $p(n,m)$-приближенного решения x' для задачи (1)–(4) позволяет ответить на вопрос о существовании требуемого вершинного покрытия положительно, если $f(x') < \Omega$, и отрицательно в противном случае. Теорема 2 доказана.

Авторы благодарят М. Ю. Ковалева, А. А. Колоколова и А. В. Кононова за полезные замечания.

Литература

[1] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

[2] Генс Г.В., Левнер Е.В. Эффективные приближенные алгоритмы для комбинаторных задач. Препринт. ЦЭМИ, М.: 1981.

[3] Chauhan S. S., Eremeev A. V., Kolokolov A. A., Servakh V. V. Concave cost supply management for single manufacturing unit // Supply chain optimisation. Product/process design, factory location and flow control. New York: Springer. Applied Optimization V. 94. 2005. P. 167–174.

[4] Chauhan S. S., Eremeev A. V., Romanova A. A., Servakh V. V., Woeginger G. J. Approximation of the supply scheduling problem // *Oper. Res. Lett.* 2005. V. 33, N 3. P. 249–254.

[5] Chauhan S. S., Proth J.–M. The concave cost supply problem // *European J. of Oper. Res.* 2003. V. 148, N 2. P. 374–383.

[6] Ibarra O. H., Kim C. E. Fast approximation algorithms for the knapsack and sum of subset problems //*J. ACM.* 1975. V. 22, N 4. P. 463–468.

[7] Woeginger G. J. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS J. on Computing.* 2000. V. 12, N 1. P. 57–74.