Управляемые системы, ИМ СО РАН, 1993 г., вып. 31

УДК 519.854

ДЕКОМПОЗИЦИОННЫЙ АЛГОРИТМ ДЛЯ ОГРАНИЧЕННОЙ ЗАДАЧИ РАЗМЕЩЕНИЯ ПРОИЗВОДСТВА

А.Ж.Жолдошев

Рассматривается достаточно общий вариант хорошо известной трудно-решаемой задачи размещения с ограничениями на объемы производства (ограниченная задача размещения производства). Для этой задачи предлагается алгоритм, основанный на декомпозиции и многократном решении пары более простых задач. Первая из них - "упрощенная" задача, получаемая посредством замены "усложняющих" ограничений исходной задачи одним "суррогатным" ограничением с параметром. Вторая - задача распознавания совместности "усложняющих" оганичений при значениях параметров, получаемых в результате решения первой задачи. Для "упрощенной" задачи строится алгоритм динамического программирования, а для решения второй задачи используется алгоритм нахождения максимального потока с выигрышами [1] в двудольной сети.

В целом, предлагаемый декомпозиционный алгоритм решения ограниченной задачи размещения производства является приближенным алгоритмом, заканчивающим работу через конечное число шагов. При первоначальном значении параметра в "суррогатном" ограничении "упрощенная" задача является релаксацией исходной задачи и, следовательно, дает оценку снизу для оптимального решения. Таким образом, получаемое приближенное решение имеет оценку для величины относительного отклонения от оптимального решения. Проведенные численные эксперименты с предлагаемым алгоритмом свидетельствуют о его работоспособности и хорошем качестве получаемого приближенного решения для некоторых подклассов исследуемой задачи.

1. Рассматривается ограниченная задача размещения производства вида:

$$\min \sum_{i\in I} c_i^0 x_i + \sum_{i\in I}\sum_{j\in J} c_i p_{ij} x_{ij},$$

$$\sum_{i\in I} x_{ij}=1, \quad \sum_{j\in J} p_{ij} x_{ij} \le V_i, \quad x_i \ge x_{ij},$$

$$x_i\in\{0,1\}, \quad x_{ij} \ge 0, \quad i\in I, \ j\in J,$$

где $I=\{1,..,m\}$ - множество возможных пунктов размещения производства продукта; $J=\{1,..,n\}$ - множество потребителей этого продукта; c_i^0- стоимость размещения производства в пункте i; c_i- стоимость производства единицы продукта в пункте i; V_i- максимальный возможный объем производства продукта в пункте i; p_{ij}- количество единиц продукта, которое требуется для удовлетворения спроса потребителя j, если продукт произведен в пункте i.

В дополнение к переменным $x_i\in\{0,1\}$, $i\in I$, где $x_i=1$, если производство продукта размещено в пункте i, и $x_i=0$ в противном случае, и переменным $x_{ij}\geq 0$, $i\in I$, $j\in J$, где x_{ij} - доля спроса потребителя j, которая покрывается производством в пункте i, введем переменные $v_i\geq 0$, $i\in I$, где v_i - обьем призводства продукта в пункте i. Используя дополнительные переменные, запишем ограниченную задачу размещения производства следующим образом:

$$\min\sum_{i\in I}\{\ c_i^0\cdot \mathrm{sign}(v_i)+c_i v_i\ \},\tag{1}$$

$$\sum_{i\in I}x_{ij}=1,\quad j\in J,\tag{2}$$

$$\sum_{j\in J}p_{ij}x_{ij}\leq v_i,\quad i\in I,\tag{3}$$

$$v_i\leq V_i,\quad i\in I,\tag{4}$$

$$v_i,x_{ij}\geq 0,\quad i\in I,j\in J.\tag{5}$$

Декомпозируем данную задачу, исключив из нее "усложняющие" ограничения (2), (3) и заменив их на одно ограничение вида $\sum\limits_{i\in I}v_i\geq p$.

В результате получаем две задачи. Первая - "упрощенная" задача, которая может быть представлена в виде:

$$\min\sum_{i\in I}f_i(v_i),\tag{6}$$

$$\sum_{i\in I}v_i\geq p,\tag{7}$$

$$v_i\geq 0,\quad i\in I,\tag{8}$$

$$f_i(v)=\begin{cases}0 & \text{, если } v=0\ ,\\ c_i^0+c_i v & \text{, если } 0<v\leq V_i,\\ \infty & \text{, если } v>V_i.\end{cases}$$

Вторая - задача распознавания совместности системы

$$\sum_{i\in I}x_{ij}=1,\quad j\in J,\tag{9}$$

$$\sum_{j\in J}p_{ij}x_{ij}\leq v_i,\quad i\in I,\tag{10}$$

$$x_{ij}\geq 0,\quad i\in I,\ j\in J,\tag{11}$$

при заданных величинах v_i, $i\in I$.

Используя указанную декомпозицию, построим приближенное решение исходной задачи (1-5). Значение p назовем допустимым, если для оптимального решения (v_i) задачи (6-8) существует решение (x_{ij}) системы (9-11). Таким образом, допустимому значению p соответствует допустимое решение $(v_i),(x_{ij})$ задачи (1-5). Отметим, что целевые функции у задач (1-5) и (6-8) совпадают, а значение целевой функции (6) не убывает с ростом значения p. Поэтому разумно в качестве искомого приближенного решения принять решение, соответствующее наименьшему допустимому значению p.

Описанию алгоритма (декомпозиционного алгоритма) поиска приближенного решения задачи (1-5), основанного на предложенной выше идее, предпошлем некоторые предложения.

Для $j \in J$ положим:

$$p_j = \min\{\, p_{ij} :\ i \in I \,\},\quad P = \sum_{j \in J} p_j\ ;$$

$$\bar{p}_j = \max\{\, p_{ij} :\ i \in I \,\},\quad \bar{P} = \sum_{j \in J} \bar{p}_j\ .$$

Лемма 1. При $p=P$ задача (6-8) есть релаксация задачи (1-5).

Доказательство. Покажем, что если (v_i), (x_{ij}) - оптимальное решение задачи (1-5), то (v_i) - решение задачи (6-8). Отсюда с учетом того, что значения целевых функций на рассматриваемых решениях равны, будет следовать требуемое. Поскольку для решения $(v_i),(x_{ij})$ выполняются ограничения (2), (3), можем написать

$$\sum_{i \in I} v_i \geq \sum_{i \in I} \sum_{j \in J} p_{ij} x_{ij} \geq \sum_{j \in J} p_j \sum_{i \in I} x_{ij} = \sum_{j \in J} p_j = P\ .$$

Из леммы следует, что оптимальное решение задачи (6-8) при $p=P$ дает оценку снизу для оптимального решения исходной задачи (1-5). Если имеем $p_{ij}=p_j$ для всех $i \in I$, $j \in J$, то оптимальное решение задачи (6-8) при $p=P$ порождает допустимое решение задачи (1-5), которое в силу леммы 1 будет оптимальным.

Лемма 2. Если (v_i) - оптимальное решение задачи (6-8) при $p=\bar{P}$, то система (9-11) совместна.

Доказательство. Пусть $x_{ij}=v_i/p$. Тогда можно написать:

$$\sum_{j \in J} p_{ij} x_{ij} = \sum_{j \in J} p_{ij} v_i/p = v_i \sum_{j \in J} p_{ij}/p \leq v_i\ .$$

Кроме того, поскольку $\sum_{i \in I} v_i = p$, имеем $\sum_{i \in I} x_{ij} = \sum_{i \in I} v_i/p = 1$.

Как следствие лемм 1 и 2 получаем, что наименьшее допустимое значение p лежит в промежутке от P до $\bar{P}$. Поиск наименьшего допустимого значения p будем вести делением отрезка пополам.

Декомпозиционный алгоритм.

Начальный шаг. Положим $p=P$, найдем оптимальное решение (v_i) задачи (6-8) и проверим совместность системы (9-11). Если система совместна и (x_{ij}) - ее решение, то алгоритм заканчивает работу и $(v_i),(x_{ij})$ - искомое приближенное решение. В противном случае положим $a_1=p$ и найдем оптимальное решение (v_i) задачи (6-8) при $p=\bar{P}$. Положим $x_{ij}=v_i/p$, $i\in I$, $j\in J$, $b_1=p$ и перейдем к шагу $l=1$.

Шаг l. Положим $p=\lfloor (a_l+b_l)/2\rfloor$. Если $a_l=p$, то алгоритм заканчивает работу и $(v_i),(x_{ij})$ - искомое приближенное решение. В противном случае найдем оптимальное решение (v_i) задачи (6-8) и проверим совместность системы (9-11). Если система совместна и (x_{ij}) - ее решение, положим $a_{l+1}=a_l$, $b_{l+1}=p$. В противном случае положим $a_{l+1}=p$, $b_{l+1}=b_l$ и перейдем к следующему шагу.

Предложенный алгоритм корректен и заканчивает работу за конечное число шагов. Необходимо только уточнить алгоритмы построения оптимального решения задачи (6-8) и допустимого решения системы (9-11).

2. Для целых значений параметра p и при условии целочисленности величин v_i, $i\in I$, оптимальное решение задачи (6-8) может быть получено стандартным способом с использованием рекуррентных соотношений динамического программирования [2]. Эти соотношения применительно к случаю рассматриваемой задачи имеют вид:

$$F_m(x)=f_m(x),$$
$$F_i(x)=\min\{\ f_i(v)+F_{i+1}(x-v)\ :\ v=0,1,\dots,x\ \},$$
$$i=1,\dots,m-1,\quad x=0,1,\dots,p.$$

Основанный на данных соотношениях алгоритм включает два этапа. На этапе 1 вычисляются функции $F_i(x)$, $i=1,\dots,m$, и одновременно функции

$$v_m(x)=x,$$
$$v_i(x)=\operatorname{argmin}\{\ f_i(v)+F_{i+1}(x-v)\ :\ v=0,1,\dots,x\ \},$$
$$i=1,\dots,m-1,\quad x=0,1,\dots,p.$$

На этапе 2, используя функции $v_i(x)$, $i=1,\dots,m$, строим оптимальное решение (v_i).

Алгоритм.

Этап 1.

Шаг m. Для $x=p,p-1,\dots,0$ положим $F(x)=f_m(x)$, $v_m(x)=x$. Перейдем к шагу $i=m-1$.

Шаг i. Для $x=p,p-1,\dots,0$ положим

$$F(x):=\min\{\ f_i(v)+F(x-v)\ :\ v=0,\dots,x\ \}\ ,$$
$$v_i(x)=\operatorname{argmin}\{\ f_i(v)+F(x-v)\ :\ v=0,\dots,x\ \}\ .$$

Если $i=1$, то переходим к этапу 2, в противном случае - к шагу $i-1$.

Этап 2.

Шаг 1. Полагаем $v_1=v_1(p)$, $p=p-v_1$ и переходим к шагу $i=2$.

Шаг i. Положим $v_i=v_i(p)$, $p=p-v_i$. Если $i=m$, то алгоритм заканчивает работу и (v_i) - оптимальное решение, в противном случае перейдем к шагу $i+1$.

3. Построим алгоритм распознавания совместности системы (9-11) при любых заданных величинах V_i, $i\in I$. Для этого используем задачу отыскания максимального потока с выигрышами [1].

Рассмотрим двудольную сеть $N=(s,t,U,E)$ с множеством вершин $U=\{s\}\cup I\cup J\cup\{t\}$, где s – вершина-исток, t – сток, и множеством дуг $E=\{(s,i),(i,j),(j,t),\ i\in I,j\in J\}$. Пропускные способности b_e и коэффициенты выигрыша k_e для дуг $e\in E$ сети зададим следующим образом:

$$b_e=\begin{cases} V_i\ , & \text{если } e=(s,i),\\ p_{ij}, & \text{если } e=(i,j),\\ 1\ , & \text{если } e=(j,t);\end{cases}\qquad k_e=\begin{cases} 1/p_{ij}, & \text{если } e=(i,j),\\ 1, & \text{в противном случае.}\end{cases}$$

Задачу о максимальном потоке с выигрышами в случае рассматриваемой двудольной сети N запишем следующим образом:

$$\max\sum_{j\in J} y_{jt}\ , \tag{12}$$

$$y_{si}-\sum_{j\in J} y_{ij}=0,\quad i\in I,\qquad \sum_{i\in I} y_{ij}/p_{ij}-y_{jt}=0,\quad j\in J, \tag{13}$$

$$0\le y_{si}\le V_i\ ,\quad 0\le y_{ij}\le p_{ij}\ ,\quad 0\le y_{jt}\le 1,\quad i\in I,\ j\in J. \tag{14}$$

Допустимое решение (y_{si}), (y_{ij}), (y_{jt}) задачи (12-14) называется потоком, а величина $\sum_{j\in J} y_{jt}$ – мощностью потока. Поток наибольшей мощности называют максимальным. Поскольку поток полностью определяется величинами y_{ij}, $i\in I$, $j\in J$, то далее под потоком будем понимать только матрицу (y_{ij}).

Связь задачи (12-14) с интересующей нас задачей распознавания совместности системы (9-11) устанавливает

Л е м м а 3. Система (9-11) совместна тогда и только тогда, когда мощность максимального потока равна n.

Д о к а з а т е л ь с т в о. Пусть мощность максимального потока равна n. Тогда $y_{jt}=1$ для всякого $j\in J$. Положим $x_{ij}=y_{ij}/p_{ij}$, тогда в силу (13) и (14), получаем, что (x_{ij}) – допустимое решение системы (9-11).

Пусть (x_{ij}) – допустимое решение системы (9-11). Положим $y_{ij}=p_{ij}x_{ij}$, $i\in I$, $j\in J$. Тогда (y_{ij}) – поток, для которого $y_{jt}=\sum_{i\in I} y_{ij}/p_{ij}=\sum_{i\in I} x_{ij}=1$ при любом $j\in J$. Следовательно, мощность потока (y_{ij}) равна n.

Таким образом, задача распознавания совместности системы (9-11) сводится к задаче (12-14). Алгоритм решения задачи о максимальном потоке с выигрышами рассматривается в [1]. Конкретизируем элементы этого алгоритма применительно к случаю двудольной сети N.

При фиксированном потоке (y_{ij}) рассмотрим так называемую сеть приращения потока $N'=(s,t,U,E')$, где $E'=E\cup\{(j,i),\ j\in J,i\in I\}$. Коэффициент выигрыша $k'(e)$ и пропускная способность $b'(e)$ для дуги $e\in E'$ сети N'

определяются следующим образом. Когда $e \in E$, то $k'(e)=1/p_{ij}$ и

$$b'(e)=\begin{cases} v_i-y_{si}, & \text{если } e=(s,I),\\ p_{ij}-y_{ij}, & \text{если } e=(I,J),\\ 1-y_{jt}, & \text{если } e=(J,t).\end{cases}$$

Когда $e=(J,I)$, $J \in J$, $I \in I$, то $k'(e)=p_{ij}$, $b'(e)=y_{ij}/p_{ij}$.

Рассмотрим путь $\mu=\{(s,I_1),(I_1,J_1),\ldots,(I_K,J_K),(J_K,t)\}$ из s в t в сети N' и части этого пути $\mu_1=(s,I_1)$, $\mu_k=\{(s,I_1),(I_1,J_1),(J_1,I_2),\ldots,(J_{k-1},I_k)\}$, $k=2,\ldots,K$. Коэффициент выигрыша $k'(\mu)$ и пропускная способность $b'(\mu)$ для пути μ выражаются следующим образом:

$$k'(\mu)=\prod_{k=1}^{K-1} p_{i_{k+1}j_k} \Big/ \prod_{k=1}^{K} p_{i_k j_k},$$

$$b'(\mu)=\min\left\{ (v_{i_1}-y_{si_1});\ (1-y_{j_K t})/k'(\mu);\ \varepsilon \right\},$$

$$\varepsilon=\min\left\{ \min\{(p_{i_k j_k}-y_{i_k j_k});\ y_{i_{k+1} j_k}\}/k'(\mu_k)\ :\ k=\overline{1,K-1} \right\}.$$

Путь μ из s в t в сети N' называют проходимым, если $b'(\mu)>0$. Проходимый путь назовем увеличивающим, если среди проходимых путей, имеющих наибольший коэффициент выигрыша, этот путь имеет наименьшую длину (число дуг, составляющих путь).

Если (y_{ij}) поток в сети N, а μ - увеличивающий путь, то увеличение потока (y_{ij}) вдоль увеличивающего пути μ есть построение нового потока $(\bar{y}_{ij})$ в сети N такого, что

$$\bar{y}_{ij}=\begin{cases} y_{ij}+b'(\mu)k'(\mu_k), & \text{если } I=I_k,\ J=J_k,\\ y_{ij}-b'(\mu)k'(\mu_k), & \text{если } I=I_k,\ J=J_{k-1},\\ y_{ij}, & \text{в противном случае.}\end{cases}$$

Алгоритм Форда-Фалкерсона построения потока наибольшей мощности.

Начальный шаг. Положим $y_{ij}=0$, $I \in I$, $J \in J$.

Общий шаг. Находим увеличивающий путь. Если такого пути нет, то алгоритм заканчивает работу и (y_{ij}) - поток наибольшей мощности. В противном случае увеличиваем поток вдоль увеличивающего пути и повторяем шаг.

Корректность алгоритма показана в [3], а его конечность в [1]. Для поиска увеличивающего пути используем мультипликативный вариант алгоритма Форда [4].

Алгоритм Форда включает два этапа. На этапе 1 вычисляется максимальный коэффициент выигрыша по всем проходимым путям из s в t. При этом на шаге r вычисляются величины f_i, $J(I)$, $I \in I$, когда r - нечетное, и величины h_j, $I(J)$, d_j, $J \in J$, когда r - четное. Величина f_i есть максимальный коэффициент выигрыша по всем проходимым путям из s в I, а номер

$J(I)$ определяет последнюю дугу $(J(I),I)$ в пути наименьшей длины из S в I с коэффициентом выигрыша f_i. Аналогично, h_j есть максимальный коэффициент выигрыша по всем проходимым путям из S в J, номер $I(J)$ задает последнюю дугу $(I(J),J)$ в пути наименьшей длины из S в J с коэффициентом выигрыша h_j. Величина d_j указывает на длину этого пути. На этапе 2 с использованием найденных на этапе 1 номеров $I(J)$, $J\in J$, $J(I)$, $I\in I$, восстанавливается путь наименьшей длины из S в t с максимальным коэффициентом выигрыша.

Мультипликативный алгоритм Форда.

Этап 1.

Начальный шаг. Положим $f_i=0$, $J(I)=1$, $h_j=0$, $I\in I$, $J\in J$.

Шаг 1. Для всякого $I\in I$, если $v_i>y_{si}$, положим $f_i=1$, $J(I)=0$.

Шаг r $(r>1)$. Когда r - четное, то для всякого $J\in J$ вычислим величины $h=max\{ f_i/p_{ij} : I\in I,\ y_{ij}<p_{ij} \}$, $k=argmax\{ f_i/p_{ij} : I\in I,\ y_{ij}<p_{ij} \}$. Если $h>h_j$, положим $h_j=h$, $I(J)=k$, $d_j=r$. Если $max\{ d_j,\ J\in J\}<r$, то переходим к этапу 2. Когда r - нечетное, то для всякого $I\in I$ вычислим $f=max\{ h_jp_{ij} : J\in J,\ y_{ij}>0 \}$, $l=argmax\{ h_jp_{ij} : J\in J,\ y_{ij}>0 \}$, и, если $f>f_i$, положим $f_i=f$, $J(I)=l$. Положим $r:=r+1$ и перейдем к следующему шагу.

Этап 2.

Шаг 1. Вычислим величину $h=max\{ h_j : J\in J,\ 0\le y_{jt}<1\}$ и положим $J_1=argmin\{ d_j : J\in J,\ h_j=h\}$, $I_1=I(J_1)$. Если $J(I_1)=0$, то алгоритм заканчивает работу и $\{(S,I_1),(I_1,J_1),(J_1,t)\}$ - искомый путь, в противном случае перейдем к шагу $r=2$.

Шаг r $(r>1)$. Положим $J_r=J(I_{r-1})$, $I_r=I(J_r)$. Если $J(I_r)=0$, то алгоритм заканчивает работу и $\{(S,I_r),(I_r,J_r),(J_r,I_{r-1}),\dots,(I_1,J_1),(J_1,t)\}$ - искомый путь, в противном случае переходим к шагу $r+1$.

Доказательство корректности и конечности алгоритма опирается на следующую лемму.

Л е м м а 4. Полученная после шага r величина f_i, когда r нечетно, и величина h_j когда r четно, есть максимальный коэффициент выигрыша по всем путям из S соответственно в I и в J с числом дуг не более r.

Д о к а з а т е л ь с т в о проведем индукцией по числу шагов. Если $r=1$, то утверждение верно. Пусть после шага r получена величина h_j (рассматриваем случай четного r, случай нечетного r рассматривается аналогично). В силу алгоритма имеем, во-первых, $h_j\ge f_i/p_{ij}$ для любого $I\in I$, $y_{ij}<p_{ij}$ и, во-вторых, $h_j=f_{i(j)}/p_{i(j)j}$. Величина f_i, $I\in I$, получена после шага $r-1$ и поэтому равняется максимальному коэффициенту выигрыша по всем путям из S в I с числом дуг не более $r-1$. Отсюда следует, что h_j - коэффициент выигрыша некоторого пути из S в J с числом дуг не

более r и что h_j - максимальный коэффициент по таким путям.

Как следствие леммы получаем, что на шаге r величина f_i, когда r нечетно (и величина h_j когда r четно), изменяется тогда и только тогда, если существует путь из S в i длины r, коэффициент выигрыша у которого больше по сравнению с путями из S в j длины не более $r-1$.

Покажем теперь, что мультипликативный алгоритм Форда за конечное число шагов действительно дает путь с наибольшим коэффициентом выигрыша и при этом наименьшей длины.

Конечность этапа 1 следует из того, что величины h_j, $j\in J$, и f_i, $i\in I$, не могут увеличиваться бесконечно. Действительно, в противном случае по лемме 4 для некоторого $j\in J$ длина пути из S в j с наибольшим коэффициентом выигрыша будет бесконечно увеличиваться. Но это означает, что, начиная с некоторого шага, путь из S в j будет содержать контур с коэффициентом выигрыша больше 1, чего не может быть в силу основной леммы из [1]. По этой же причине будет конечен и этап 2. Действительно, если последовательность $j_1,i_1,j_2,i_2,\ldots$ бесконечна, то при некотором r имеем либо $j_r=j(i_{r-1})=j_{r'}$, либо $i_r=i(j_r)=i_{r'}$, где $1\le r'\le r-1$. Пусть $j_r=j_{r'}$ (случай $i_r=i_{r'}$ рассматривается аналогично). Среди величин $h_{j_{r'}},f_{i_{r'}},\ldots,h_{j_{r-1}},f_{i_{r-1}},h_{i_r}$ существует такая, которая которая в ходе работы алгоритма изменялась (увеличивалась) последней. Если таковой является величина h_{j_k}, где $r'<k\le r$, то имеем $f_{i_{k-1}}<h_{i_k}p_{i_{k-1}j_k}$, а если величина f_{i_k}, где $r'\le k<r$, то $h_{j_k}<f_{i_k}/p_{i_kj_k}$. В обоих случаях вступаем в противоречие с тем, что этап 1 закончен.

Пусть путь$\{(S,i_r),(i_r,j_r)(j_r,i_{r-1}),\ldots,(i_1,j_1),(j_1,t)\}$, полученный в результате работы алгоритма, по построению имеет коэффициент выигрыша равный h_{j_1}. Длина этого пути равняется $2r$. Чтобы убедиться, что этот путь имеет наименьшую длину, заметим, что величина h_{j_1} получена на шаге $2r$ этапа 1. Действительно, пусть h_{j_1} получена на шаге $2r'$, $r'<r$. Тогда f_{i_1} получена на шаге $2r'-1$, h_{j_2} - на шаге $2r'-2$, а $f_{i_{2r-2r'+1}}$ - на шаге 1. Но тогда $j(i_{2r-2r'+1})=0$, что противоречит построению рассматриваемого пути.

4. Предложенный алгоритм не является одинаково продуктивным (в смысле точности приближенного решения) на всем классе задач. По построению он ориентирован на подкласс задач, для которых, во-первых, ограничения (4) существенны и, во-вторых, разброс величин p_{ij}, $i\in I$, при фиксированном $j\in J$ не очень большой. Возможность получения "хорошего" приближенного решения подтверждается результатами численных экспериментов с алгоритмом, проведенных на следующих подклассах задач:

c_i^0 - постоянная величина C ;

c_i - равномерно распределенная случайная величина со значениями 1,2,3;

p_{ij} - равномерно распределенная случайная величина со значениями 1,2,3,...,9;

V_i - равномерно распределенная случайная величина со значениями 10,20,30.

Расчеты проводились для серий задач (20 задач в каждой серии) разной размерности (m=30,...,40, n=30,...,40) и при различных значениях параметра C (C=30,...,60). Полученные в ходе эксперимента оценки для относительного отклонения приближенного решения от оптимального не превышают величины 0.1.

Поступила в ред.-изд. отдел
9 октября 1992 г.

Литература

1. Береснев В.Л., Жолдошев А.Ж. Алгоритм для задачи о максимальном потоке с выигрышами // Управляемые системы. - Новосибирск, 1990. - Вып. 30. - С. 17-24.

2. Береснев В.Л., Гимади Э.Х., Дементьев В.Т. Экстремальные задачи стандартизации. - Новосибирск: Наука, 1978. - 333 с.

3. Onaga K. Optimum flows in general communication networks // J. of the Franclin Institute. - 1967.- P. 308-327.

4. Форд Л., Фалкерсон Д. Потоки в сетях. - М:Мир, 1966. - 276 с.