МЕТОДЫ ОПТИМИЗАЦИИ

НОВЫЙ МЕТОД РЕШЕНИЯ МНОГОПРОДУКТОВОЙ ТРАНСПОРТНОЙ ЗАДАЧИ*

© 1995 г. **Гольштейн Е.Г., Соколов Н.А.**

(*Москва*)

Описан новый алгоритм решения многопродуктовой транспортной задачи, основанный на двойственной декомпозиции. Приведенные результаты численной апробации алгоритма подтверждают его вычислительную эффективность.

1. ВВЕДЕНИЕ

Статья посвящена новому методу решения многопродуктовой транспортной задачи на сети, который основан на декомпозиционном подходе, описанном в [1, 2].

Начнем с постановки задачи. Пусть имеется произвольная транспортная сеть (P, Γ), т.е. граф, который определен множеством P пунктов и множеством Γ направленных дуг, соединяющих некоторые пары различных пунктов из P. Рассмотрим общую многопродуктовую транспортную задачу на сети (P, Γ). Для этого введем обозначения: c_{ijr} – стоимость перевозки единицы продукта r из пункта i в пункт j, $r = 1,\ldots, s$, $(i, j) \in \Gamma$; a_{ir} – требуемое количество продукта r в пункте i (для пунктов отправления $a_{ir} > 0$, для пунктов назначения $a_{ir} < 0$, для промежуточных $a_{ir} = 0$, $r = 1,\ldots, s$, $i \in P$); d_{ijr} – пропускная способность дуги (i, j) для груза r, $r = 1,\ldots, s$, $(i, j) \in \Gamma$; d_{ij} – пропускная способность дуги (i, j) для приведенной суммы всех s грузов, $(i, j) \in \Gamma_1 \subset \Gamma$; $\alpha_{ijr} > 0$ – коэффициент приведения груза r на дуге (i, j), $r = 1,\ldots, s$, $(i, j) \in \Gamma_1$.

Математическая формулировка задачи имеет вид

$$c(x) = \sum_{r=1}^{s} \sum_{(i,j)\in\Gamma} c_{ijr} x_{ijr} \to \min, \tag{1}$$

$$\sum_{j:(i,j)\in\Gamma} x_{ijr} - \sum_{j:(i,j)\in\Gamma} x_{jir} = a_{ir}, \quad r = 1,\ldots,s, \quad i \in P, \tag{2}$$

$$0 \leqslant x_{ijr} \leqslant d_{ijr}, \quad r = 1,\ldots,s, \quad (i,j) \in \Gamma, \tag{3}$$

$$\sum_{r=1}^{s} \alpha_{ijr} x_{ijr} \leqslant d_{ij}, \quad (i,j) \in \Gamma_1. \tag{4}$$

Искомым планом в задаче (1)–(4) является набор x перевозок x_{ijr}.

Если ограничения (4) отсутствуют, то задача распадается на s обычных транспортных задач на сети. Предлагаемый метод сводится к поиску разрешающих множителей, отвечающих ограничениям (4), и на каждом шаге требует решения s транспорт-

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 93-012-499).

ных задач. Будем считать выполненными необходимые условия разрешимости задачи (1)–(4): $\sum_{i \in P} a_{ir} = 0,\ r = 1,\ldots,s$, которые, конечно, не являются достаточными. Предположим, что ограничения (2), (3) совместны, не исключая возможности несовместности всей системы (2)–(4).

2. ОПИСАНИЕ МЕТОДА

Рассмотрим семейство задач

$$T(R):\ c(x,R) = c(x) + R \sum_{(i,j)\in\Gamma_1} \left| \sum_{r=1}^{s} d_{ijr}x_{ijr} - d_{ij} \right|_+ \to \min,\ x \in G, \tag{5}$$

зависящее от положительного параметра R, где x – набор перевозок задачи (1)–(4); G – множество, определяемое ограничениями (2)–(3); $|z|_+ = \begin{cases} z, & z \geqslant 0, \\ 0, & z < 0 \end{cases}$. Задача $T(R)$ образуется из (1)–(4) с помощью замены жестких ограничений (4) линейным штрафованием за их невыполнение с коэффициентом штрафа R. Описываемый здесь метод позволяет найти приближенное решение задачи $T(R)$ при фиксированном R; вопрос о выборе R, при котором найденное решение является приближенным для задачи (1)–(4), обсуждается ниже.

С каждым из ограничений (4) свяжем множитель Лагранжа y_{ij}; совокупность этих множителей составляет вектор y с числом координат, равным $|\Gamma_1|$. Введем вогнутую функцию

$$f(y) = \min_{x\in G}(c(y),x) - (d,y), \tag{6}$$

где $c(y)$ – вектор с координатами

$$c_{ijr}(y) = \begin{cases} c_{ijr} + \alpha_{ijr}y_{ij},\ (i,j) \in \Gamma_1,\ r = 1,\ldots,s, \\ c_{ijr},\ (i,j) \in \Gamma \setminus \Gamma_1,\ r = 1,\ldots,s; \end{cases}$$

d – вектор с координатами d_{ij}, $(i,j) \in \Gamma_1$.

Рассмотрим задачу

$$\tilde{T}(R):\ f(y) \to \max,\ y \in Q(R), \tag{7}$$

где $Q(R) = \{y: 0 \leqslant y_{ij} \leqslant R, (i,j) \in \Gamma_1\}$, которая является двойственной по отношению к $T(R)$ (см. [2]). Общее экстремальное значение задач $T(R)$ и $\tilde{T}(R)$ обозначим через $V(R)$; под V условимся понимать экстремальное значение задачи (1)–(4).

Предлагаемый алгоритм основан на применении метода уровней [3] к задаче (7). Он состоит из однотипных операций. Опишем одну из них.

После $(k-1)$-й итерации, $k = 1,\ldots$, имеем векторы $y^t \in Q(R)$, $t = 1,\ldots,k$, и планы перевозок $x^t \in G$, $t = 1,\ldots,k-1$ (вектор y^1 выбирается произвольно из множества $Q(R)$, например $y^1 = 0$). Очередной, k-й шаг метода начинается с пополнения множества $\{x^t\}$ планом x^k, в качестве которого выбирается один из оптимальных планов задачи

$$(c(y^k), x) \to \min,\ x \in G. \tag{8}$$

Подчеркнем еще раз, что (8) распадается на s однопродуктовых задач.

Обозначим через $G(k)$ выпуклую оболочку точек $x^1, \ldots, x^k$ и рассмотрим кусочно-линейную вогнутую функцию

$$f_k(y) = \min_{x\in G(k)} (c(y),x) - (d,y),$$

являющуюся верхней аппроксимацией функции $f(y)$, определяемой соотношением (6).

Нетрудно видеть, что задачи

$$T(R, k)\colon c(x, R) \to \min, \quad x \in G(k), \tag{9}$$

$$\tilde{T}(R,k)\colon \ f_k(y) \to \max, \quad y \in Q(R) \tag{10}$$

взаимно двойственные задачи линейного программирования. Их общее экстремальное значение обозначим через $V(R, k)$.

Продолжая k-ю итерацию, находим решение задачи (10), легко приводимой к виду

$$\upsilon \to \max, \quad (c(y), x^t) - (d, y) \geqslant \upsilon, \quad t = 1,\ldots, k, \quad y \in Q(R). \tag{11}$$

Одновременно с этим определяем также решение $\bar{x}^k$ задачи (9), причем легко проверить, что

$$\bar{x}^k = \sum_{t=1}^{k} \gamma_t^k x^t, \tag{12}$$

где γ_t^k – первые k компонент решения задачи, двойственной к (11). Поскольку задача (11) решается с помощью двойственного симплекс-метода, в сумме из правой части (12) содержится не более чем $|\Gamma_1| + 1$ слагаемых.

Имеет место соотношение

$$c(\bar{x}^k, R) = V(R,k) \geqslant V(R) = \max_{y \in Q(R)} f(y).$$

Поэтому

$$c(\bar{x}^k, R) - V(R) \leqslant \Delta_k, \tag{13}$$

где $\Delta_k = c(\bar{x}^k, R) - \max\limits_{1 \leqslant i \leqslant k} f(y^i)$ – управляющий параметр вычислительного процесса.

Если $\Delta_k \leqslant \varepsilon$, где $\varepsilon > 0$ – заранее выбранная точность, то вычислительный процесс завершается. В противном случае происходит пополнение множества $\{y^i\}$ новой точкой y^{k+1}. В качестве этой точки принимается проекция y^k на многогранник $\{y \in Q(R)$: $f_k(y) \geqslant \rho_s\}$, где $\rho_s = \dfrac{1}{2}(V(R,s) + f(x^s))$. Таким образом, отыскание точки y^{k+1} сводится к решению задачи квадратичного программирования

$$|y - y^k|^2 \to \min, \quad (c(y), x^t) - (d, y) \geqslant \rho_k, \quad t = 1,\ldots, k, \quad y \in Q(R).$$

Этим завершается k-я итерация метода.

Несколько слов о связи между задачами (1)–(4) и $T(R)$ и о выборе штрафного параметра R.

Зафиксируем точность $\varepsilon > 0$.

План перевозок x, удовлетворяющий всем условиям разрешимой задачи (1)–(4), кроме, быть может, требований (4), назовем ε-решением, если

$$(c, x) \leqslant V + \varepsilon, \ \rho(x) = \max_{(i,j) \in \Gamma_1} \left| \sum_{r=1}^{s} \alpha_{ijr} x_{ijr} - d_{ij} \right|_+ \leqslant \varepsilon.$$

Положим

$$L = \max_{x \in G}(c, x), \quad l = \min_{x \in G}(c, x)$$

и определим $R = 1 + (L - l)/\varepsilon$. Проведем такое число $k = k(\varepsilon)$ итераций метода, что $\Delta_k \leqslant \varepsilon$. Могут представиться две возможности: $\rho(\bar{x}^k) \leqslant \varepsilon$ и $\rho(\bar{x}^k) > \varepsilon$. При второй

условия (2)–(4) исходной задачи противоречивы. Пусть имеет место первая возможность. Для разрешимой задачи (1)–(4) план $\bar{x}^k$ является ее ε-решением, в противном случае он оказывается ε-решением возмущенной задачи (1)–(4), в которой правая часть d ограничений (4) заменена на d', причем $\max\limits_{(i,j)\in\Gamma_1} |d_{ij} - d'_{ij}| \leqslant \varepsilon$. Сформулированные утверждения – следствие теормы 2 из [1].

3. ВЫЧИСЛИТЕЛЬНЫЙ ОПЫТ

Описанный алгоритм был опробован на задаче снабжения с промежуточными базами. Сформулируем ее.

Имеется m пунктов отправления и n – назначения. В каждом пункте отправления содержатся определенные количества s различных продуктов, потребности в которых фиксированы в каждом пункте назначения. Снабжение осуществляется через p баз, пропускные возможности которых ограничены. Требуется составить оптимальный план снабжения. Математическая формулировка этой задачи имеет вид

$$\sum_{i=1}^{m}\sum_{l=1}^{p}\sum_{r=1}^{s} c'_{ilr}x'_{ilr} + \sum_{l=1}^{p}\sum_{j=1}^{n}\sum_{r=1}^{s} c''_{ljr}x''_{ljr} + \sum_{i=1}^{m}\sum_{l=1}^{p}\sum_{r=1}^{s} c'''_{lr}x'_{ilr} \to \min \tag{14}$$

при условиях

$$\sum_{l=1}^{p} x'_{ilr} = a_{ir}, \quad i = 1,\ldots,m, \quad r = 1,\ldots,s, \tag{15}$$

$$\sum_{l=1}^{p} x''_{ljr} = b_{jr}, \quad j = 1,\ldots,n, \quad r = 1,\ldots,s, \tag{16}$$

$$\sum_{r=1}^{s}\sum_{i=1}^{m} x'_{ilr} = \sum_{r=1}^{s}\sum_{j=1}^{n} x''_{ljr} \leqslant d_l, \quad l = 1,\ldots,p, \tag{17}$$

$$x'_{ilr} \geqslant 0, \quad x''_{ljr} \geqslant 0, \tag{18}$$

где c'_{ilr} – стоимость перевозки единицы продукта r из пункта отправления i на базу l; c''_{ljr} – стоимость перевозки единицы продукта r из базы l в пункт назначения j; c'''_{lr} – стоимость складирования единицы продукта r на базе l; a_{ir} – количество продукта r в пункте отправления i; b_{jr} – величина потребности продукта r в пункте назначения j; d_l – пропускные возможности базы l; x'_{ilr} и x''_{ljr} – объемы соответствующих перевозок.

Нетрудно видеть, что эта задача сводится к (1)–(4) с числом пунктов $N = m + n + 2p$ (m пунктов отправления, n – назначения и $2p$ – промежуточных), с числом $U = (m + n + 1)p$ направленных дуг, из которых на p дугах имеются ограничения типа (4). В дальнейшем будем предполагать задачу (14)–(18) совместной, т.е.

$$\sum_{i=1}^{m} a_{ir} = \sum_{j=1}^{r} b_{jr} = e_r, \quad r = 1,\ldots,s, \quad \sum_{r=1}^{s} d_r = \beta\sum_{r=1}^{s} e_r, \quad \beta \geqslant 1. \tag{19}$$

Одной из целей вычислительного эксперимента являлась проверка на рассматриваемом классе задач геометрической скорости сходимости алгоритма. В частности, установлена следующая оценка

$$V(R) - f(y^k) \leqslant \left(V(R) - \min_{y\in Q(R)} f(y)\right)\exp\left(-\frac{k}{\mu p}\right), \quad k = 1,\ldots, \tag{20}$$

где константа μ не превышает 1. Кроме того, получено аналитическое выражение для общей трудоемкости решения.

В вычислительном эксперименте был использован широкий набор задач типа (4)–(18). Расчеты проводились на ПК IBM 386 и 486 (с сопроцессором, тактовая частота 33 Мгц, объем оперативной памяти 4 Мб) с помощью программы DMT, написанной на FORTRAN'е (транслятор NDP FORTRAN, версия 2,0). Вся информация хранилась в оперативной памяти. Как и в [2], в качестве решателя задач линейного и квадратичного программирования была использована программа [4]. Для решения транспортных задач в сетевой постановке была применена более современная версия программы CNEXTR из [5]*, позволяющая использовать накопленную информацию при решении близких задач и дающая значительный выигрыш времени. Размеры решаемых задач: $5 \leqslant p \leqslant 50$, p кратно 5, $10 \leqslant m \leqslant n \leqslant 100$, m, n кратны 10, $s \in \{2, 3, 5, 10, 15, 20\}$. Вследствие ограниченности оперативной памяти рассматривался случай, когда стоимости перевозок вдоль дуг сети не зависели от вида продукта r (т.е. $c'_{ilr} = c'_{il}$, $i = 1,\ldots,m$; $c''_{ljr} = c''_{lj}$, $j = 1,\ldots,n$, $c'''_{lr} = c'''_{l}$ $l = 1,\ldots,p$, $r = 1,\ldots,s$).

Для каждого набора (m, p, n, s) решалась серия из 15 тестовых задач, после чего параметры усреднялись. Задача считалась решенной, если удавалось найти приближенное решение с относительной точностью θ

$$\left|\frac{\Delta_k}{f_k(y_k)}\right| \leqslant \theta, \tag{21}$$

где θ, а также параметры R из (5), β из (19) задавались заранее.

Целые константы a_{ir}, b_{jr}, d_l и вещественные константы $c'_{il}, c''_{lj}, c'''_{l}$ генерировались с помощью датчика случайных чисел на интервалах $0 \leqslant a_{ir}, b_{jr}, d_l \leqslant 1000$, $0 < c'_{il}, c''_{lj}, c'''_{l} \leqslant 10$ соответственно, при этом пропускные способности d_l задавались по возможности равными, а c'''_{l} дополнительно преобразовывались так, чтобы сделать эти цены хранения существенно различными.

Как и в [2], для сокращения времени счета при тестировании алгоритма был использован вычислительный прием "обновление".

Проведенные вычислительные эксперименты показали высокую эффективность и надежность предложенного алгоритма при решении всех тестовых задач.

Наиболее полное тестирование проведено для $s = 5$, $R = 10^5$, $\beta = 1{,}1$, $\theta = 10^{-5}$. Для числа баз $5 \leqslant p \leqslant 35$ вычисления производились на ПК IBM 386, для $40 \leqslant p \leqslant 50$ – на ПК IBM 486. (Сравнение трудоемкости решения ряда задач одновременно на обеих ПК выявило, что скорость счета на 486 примерно в 2,8–3,2 раза выше, чем на 386. При сопоставлении времени работы алгоритма использован коэффициент 3,0.) Всего решено 8250 задач. Поскольку при различных p, $5 \leqslant p \leqslant 50$, зависимость результатов от количества пунктов m и n оказалась примерно одинаковой, приведем их для $p = 20$ (табл. 1).

Смысл столбцов m и n этой таблицы очевиден. В остальных ее столбцах: k – среднее число больших итераций (метода уровней) для получения относительной точности θ; k_l/k – среднее число итераций линейного решателя на одну итерацию метода; k_q/k – среднее число итераций квадратичного решателя на одну итерацию метода; π – среднее число ограничений при решении вспомогательных линейных задач (без обновлений оно было бы равно k); $\tilde{\mu}$ – приближенное значение коэффициента μ из (20), определяемое соотношением

$$\tilde{\mu} = \frac{k}{p \ln \dfrac{L_k - l_k}{\Delta_k}}, \quad L_k = \max_{1 \leqslant t \leqslant k} f(y^t), \quad l_k = \min_{1 \leqslant t \leqslant k} f(y^t), \tag{22}$$

Time – среднее время, с, на нахождение приближенного решения с точностью θ, T –

---

* Автор программы Б.В. Черкасский любезно согласился внести в нее некоторые изменения, требуемые декомпозиционным алгоритмом.

Таблица 1

m	n	k	k_1/k	k_q/k	π	$\tilde{\mu}$	T	*TiLev*	*Time*
10	10	111,8	20,29	12,31	24,12	0,41	28,7	44,7	62,6
10	20	106,1	20,34	12,04	23,93	0,37	40,3	42,1	70,4
10	30	111,8	20,98	11,85	24,74	0,39	47,7	45,4	86,6
10	40	104,7	21,14	11,78	23,90	0,36	55,8	42,4	95,7
10	50	104,5	21,54	11,59	23,78	0,37	61,3	42,5	109,9
10	60	103,1	21,18	11,38	23,27	0,34	67,0	41,2	125,0
10	70	102,5	21,47	11,34	23,46	0,35	71,2	41,5	144,2
10	80	105,2	21,63	11,17	23,72	0,38	74,3	42,4	164,8
10	90	102,7	21,74	11,34	23,70	0,37	77,5	41,6	184,8
10	100	103,7	21,26	11,10	22,54	0,37	80,5	41,0	209,9
20	20	104,2	20,82	11,58	23,27	0,36	48,7	41,2	80,3
20	30	103,7	21,48	11,48	23,35	0,37	55,3	41,7	93,3
20	40	102,9	21,26	11,33	23,60	0,35	62,5	41,1	109,6
20	50	102,0	21,68	11,33	23,25	0,36	66,8	41,1	123,8
20	60	104,0	21,16	11,66	23,31	0,35	71,6	41,8	147,1
20	70	101,7	21,71	11,21	23,36	0,36	74,7	40,5	159,9
20	80	100,8	21,93	11,07	23,43	0,35	78,1	40,4	184,0
20	90	105,2	21,94	11,18	23,41	0,37	80,1	42,5	213,1
20	100	101,4	22,30	11,10	23,55	0,36	82,5	41,4	236,2
30	30	103,5	20,87	11,50	23,43	0,36	62,1	40,9	107,7
30	40	102,1	21,65	11,25	22,79	0,34	66,8	40,9	123,3
30	50	104,7	21,93	11,29	23,33	0,37	70,6	42,6	144,9
30	60	100,7	21,74	11,21	23,40	0,36	74,8	40,3	160,1
30	70	99,7	21,88	11,00	23,07	0,35	78,2	39,5	181,0
30	80	100,7	21,93	11,06	22,70	0,35	80,8	40,1	208,4
30	90	100,5	22,24	10,89	23,50	0,37	81,7	41,1	225,0
30	100	99,5	22,32	11,01	23,25	0,36	83,8	41,0	252,7
40	40	101,2	21,62	11,31	22,76	0,34	70,8	40,9	140,0
40	50	98,0	22,07	10,80	23,21	0,37	73,7	39,5	150,3
40	60	100,7	21,58	11,23	23,27	0,34	77,9	40,3	182,5
40	70	103,6	21,64	11,10	22,98	0,36	79,9	41,5	206,4
40	80	98,4	21,73	11,05	22,93	0,35	82,4	39,5	224,6
40	90	96,3	22,07	11,10	22,82	0,34	84,4	39,3	251,5
40	100	99,5	21,77	10,96	23,35	0,35	85,9	40,0	285,5
50	50	101,1	22,01	10,90	23,44	0,36	76,7	40,7	174,9
50	60	102,2	21,86	11,17	23,33	0,36	79,7	41,2	202,8
50	70	100,7	22,00	11,11	23,53	0,35	81,8	41,0	225,5
50	80	96,2	21,62	11,07	22,52	0,34	84,6	38,5	250,1
50	90	99,7	21,92	11,12	23,07	0,34	85,7	40,5	283,9
50	100	97,7	21,94	11,03	22,61	0,34	87,0	39,8	305,5
60	60	98,5	22,60	11,04	23,23	0,34	81,5	40,5	218,7
60	70	97,1	22,15	11,06	22,50	0,34	83,9	39,5	244,6
60	80	98,2	22,30	10,91	23,30	0,35	85,3	40,4	274,3
60	90	98,7	21,93	11,05	22,72	0,35	87,0	40,3	308,5
60	100	99,1	22,29	10,93	22,52	0,35	88,3	40,5	346,6
70	70	102,0	22,54	10,97	23,91	0,36	84,7	42,5	277,2
70	80	98,2	22,48	11,05	22,89	0,34	86,7	40,4	302,7
70	90	98,5	22,13	10,94	23,06	0,35	88,0	40,3	336,2
70	100	100,3	22,52	10,99	23,03	0,34	88,8	41,5	371,6
80	80	98,5	22,27	10,88	23,06	0,35	87,3	40,4	319,3
80	90	103,3	22,33	11,12	23,65	0,35	88,2	42,9	363,7
80	100	97,5	22,11	11,02	22,46	0,34	89,8	39,7	389,4
90	90	97,0	22,09	10,97	22,47	0,34	89,4	39,4	372,6
90	100	96,7	22,06	10,75	22,20	0,35	90,2	39,2	399,5
100	100	100,7	22,45	10,97	23,77	0,35	90,5	42,0	445,3

Таблица 2

p	k	k/p	k_l/k	k_l/kp	k_q/k	k_q/kp	π	π/p	$\tilde{\mu}$	*Time*
5	28,9	5,79	7,14	1,43	5,20	1,04	6,15	1,23	0,46	18,45
10	57,8	5,78	12,06	1,21	7,31	0,73	12,43	1,24	0,45	61,28
15	78,7	5,25	16,85	1,12	9,44	0,63	17,46	1,16	0,38	127,78
20	101,3	5,07	21,79	1,09	11,19	0,56	23,23	1,16	0,36	215,61
25	116,9	4,68	26,62	1,06	12,97	0,52	27,70	1,11	0,32	318,24
30	135,3	4,51	31,88	1,06	14,43	0,48	33,19	1,11	0,31	461,31
35	155,0	4,43	36,37	1,04	16,13	0,46	38,33	1,10	0,30	650,24
40	163,5	4,09	41,88	1,05	17,20	0,43	42,40	1,06	0,28	806,11
45	210,9	4,69	45,65	1,01	19,64	0,44	54,21	1,20	0,31	1192,64
50	190,7	3,81	51,37	1,03	19,69	0,39	50,94	1,02	0,26	1315,06

Таблица 3

s	k	k_l/k	k_l/kp	k_q/k	k_q/kp	π	π/p	$\tilde{\mu}$	*Time*
2	88,9	20,5	1,03	10,8	0,55	21,6	1,09	0,30	88,34
3	94,8	21,1	1,06	11,2	0,57	22,7	1,13	0,32	134,26
5	99,5	21,7	1,09	11,3	0,57	22,9	1,15	0,36	224,07
10	101,9	22,0	1,11	11,5	0,58	22,5	1,13	0,39	453,79
15	102,3	22,0	1,11	11,5	0,58	21,9	1,10	0,40	684,94
20	101,3	22,0	1,11	11,4	0,58	21,5	1,08	0,40	896,38

средний процент временны́х затрат *TiOr* на решение всех транспортных задач (для нахождения значений $f(y^k)$ и $f'(y^k)$) по отношению к общему времени; *TiLev* – время собственно алгоритма без решения транспортных задач, с.

В табл. 2 приведена зависимость тех же величин от количества баз p при $R = 10^5$, $\beta = 1{,}1$, $\theta = 10^{-5}$. Усреднение проводилось по 825 задачам (55 серий по 15 задач).

Из приведенных результатов можно сделать следующие выводы.

При фиксированном количестве p баз число k больших итераций алгоритма мало меняется в зависимости от параметров m и n; незначительна зависимость от m и n среднего числа итерций линейного и квадратичного решателя на одну большую итерацию алгоритма. Мало изменяются в зависимости от размеров внутренней транспортной задачи π и $\tilde{\mu}$.

Для средних значений величин k, k_l/k, k_q/k характерна почти линейная зависимость от параметра p (см. столбцы 3, 5 и 7 табл. 2, где k, k_l/k, k_q/k поделены на p); в отличие от [2] заметна тенденция уменьшения значений π/p и $\tilde{\mu}$ с ростом p.

Как и следовало ожидать, с ростом размеров транспортных задач увеличивается и доля временны́х затрат (*TiOr*) на их решение, а также общее время (*Time*) решения задачи.

Вычислительные эксперименты, описываемые ниже, проводились на ПК IBM 386 для 30 серий тестовых задач при $10 \leqslant m = n \leqslant 100$, $p \in \{15, 20, 25\}$. Усреднение осуществлялось по 450 задачам.

Зависимость основных параметров k, k_l, k_q, π $\tilde{\mu}$ от числа продуктов s иллюстрируется табл. 3 при $R = 10^5$, $\beta = 1{,}1$, $\theta = 10^{-5}$. Средние значения величин k, k_l/k, k_q/k, $\tilde{\mu}$ очень медленно растут с увеличением s. Величины k_l/kp, k_q/kp, π/p можно считать постоянными.

Было замечено, что время *TiLev* пропорционально числу k больших итераций. Для *TiOr* обнаружена более сложная зависимость. Приведем эмпирические оценки для величин *TiOr*, *TiLev* и *Time*:

$$Time = TiLev + TiOr,$$

$$TiLev \approx c_1 kp^2,\ c_1 \in (8{,}6 \cdot 10^{-4};\ 2{,}1 \cdot 10^{-3}),$$

$$TiOr \approx c_2 k^{3/4} sp(m+n)^{3/2},\quad c_2 \in (3{,}2 \cdot 10^{-5};\ 8{,}3 \cdot 10^{-5}).$$

Подставив формулу $k = p\tilde{\mu}\ln(1/\varepsilon)$, которая имеет место для каждой из решаемых задач при $\varepsilon = \Delta_k/(L_k - l_k)$ в силу (22), получим

$$TiLev \approx c_1' p^3 \ln(1/\varepsilon),\quad c_1' \in (2{,}7 \cdot 10^{-4};\ 9{,}61 \cdot 10^{-4}),$$

$$TiOr \approx c_2' sp^{7/4}(m+n)^{3/2}\{\ln(1/\varepsilon)\}^{3/4},\quad c_2' \in (1{,}1 \cdot 10^{-5};\ 3{,}6 \cdot 10^{-5}).$$

Заметим, что во всех расчетах $\tilde{\mu} < 1$. Максимальное $\tilde{\mu} = 0{,}68$. Диапазон изменения усредненных значений $\tilde{\mu} \in (0{,}18;\ 0{,}51)$.

Величины k_l, k_q, *TiLev* можно было бы улучшить, если использовать решатели линейных и квадратичных задач, учитывающие близость задач, решаемых на соседних итерациях. К сожалению, программа [4] для этого не приспособлена.

Были проведены вычислительные эксперименты для трех значений параметра $\theta = 10^{-g}$, $g \in \{3, 4, 5\}$. С ростом требований к относительной точности возрастают среднее количество больших итераций k алгоритма и все временны́е характеристики. Эксперимент подтвердил логарифмическую зависимость k от θ, или, что то же самое, от ε. Очень медленно растут k_l/kp, π/p, $\tilde{\mu}$ и также k_q/kp очень медленно убывает.

Вычислительные эксперименты для пяти различных наборов R ($R = 10^h$, $h \in \{1, 2, 3, 4, 5\}$) дали следующие результаты. При $R = 10$ ни одна из 450 задач не была решена, как и 150 задач при $R = 100$ для $p = 25$, для остальных R все 30 серий задач были успешно решены. С ростом штрафного коэффициента растет среднее количество итераций k и среднее время *Time* решения задачи, однако на просмотренном диапазоне изменения R изменение этих параметров не превышает 1,5 раза. Величины k_l/kp, k_q/kp, π/p очень медленно растут с увеличением R, $\tilde{\mu}$ медленно убывает.

Как показали вычислительные эксперименты, проведенные для пяти вариантов $\beta \in \{1{,}0;\ 1{,}1;\ 1{,}2;\ 1{,}5;\ 2{,}0\}$, наиболее трудоемким оказался случай $\beta = 1{,}1$. При $\beta = 1{,}0$ среднее значение k больших итераций алгоритма равномерно (по всем задачам) меньше, чем для $\beta = 1{,}1$. Также с ростом β от 1,1 до 2,0 основные параметры улучшаются. Только k_q/kp на всем диапазоне β увеличивается с ростом β.

В заключение приведем сравнительный анализ решения двух многопродуктовых транспортных задач описанным выше алгоритмом *DMT* (при стандартных значениях параметров) и симплекс-алгоритмом (вычисления с помощью программы *MINOS* проведены У.Х. Малковым). Первая задача с $m = 20$, $n = 30$, $p = 15$, $s = 5$ приводится к стандартной задаче линейного программирования с числом строк $\overline{M} = 416$, столбцов $\overline{N} = 4242$, $\overline{L} = 12300$ строк записей в *MPS*-формате. Для второй задачи $m = 30$, $n = 50$, $p = 25$, $s = 10$, $\overline{M} = 1326$, $\overline{N} = 20250$, $\overline{L} = 63600$. Для первой задачи декомпозиционный алгоритм *DMT* нашел приближенное решение с относительной точностью $\theta = 10^{-5}$ за $k = 86$ больших итераций и *Time* = 52 с, *MINOS* получил решение за 3045 симплексных итераций и 645 с. Для второй задачи *DMT* понадобилось 119 итераций и 412 с, *MINOS* затратил 17509 итераций и 17 305 с. Таким образом, для первой задачи наблюдался выигрыш по времени более чем в 12 раз, для второй – более чем в 42 раза.

ЛИТЕРАТУРА

1. *Гольштейн Е.Г.* Об одном двойственном блочном методе линейного программирования // Автоматика и телемеханика. 1994. № 12.
2. *Гольштейн Е.Г., Немировский А.С., Соколов Н.А.* Новый алгоритм двойственной декомпозиции с приложением к задачам транспортного типа // Экономика и мат. методы. 1994. Т. 30. Вып. 2.
3. *Lemarechal C., Nemirovskii A., Nesterov Yu.* New Variants of Bundle Methods // Res. Report № 1508. Institut National de Recherche en Informatique et en Automatique (INRIA). Paris, 1991.
4. *Goldfarb D., Idnani A.* A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Problems // Math. Program. 1983. V. 27. № 1.
5. Программа CNEXTR решения сетевой транспортной задачи с ограничениями пропускной способности / Пакет анализа оптимизационных моделей ППП ПАОЭМ ЕС, задачи транспортного типа. Калинин: НПО ЦентрПрограммСистем, 1982.

Поступила в редакцию
14 XI 1994