

ТРАНСПОРТНАЯ ЗАДАЧА И ОРТОГОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ НА ЛИНЕЙНЫЕ МНОГООБРАЗИЯ¹

П.И. СТЕЦЮК, Е.А. НУРМИНСКИЙ, Д.И. СОЛОМОН,
Институт кибернетики им. В.М.Глушкова НАН Украины,
Киев, Украина,
Институт автоматике и процессов управления ДВО РАН,
Владивосток, Россия,
Академия транспорта, информатики и коммуникаций,
Кишинэу, Молдова,
stetsyukp@gmail.com, nurmi@dvo.ru, atic@mtc.md

Рассматривается задача нахождения наименее удаленного от заданной точки решения системы линейных уравнений с двусторонними границами на переменные. Изложен двойственный алгоритм решения этой задачи и описана его программная реализация на основе r -алгоритма Шора. Приведены результаты вычислительных экспериментов для матричной транспортной задачи с ограничениями на объемы поставок.

Ключевые слова: транспортная задача, система линейных уравнений, задача квадратичного программирования, r -алгоритм, octave-программа.

Введение. Балансовые соотношения в транспортных задачах и задачах транспортно-производственного вида [1, 2] описываются линейными системами с вытянутыми матрицами, в которых намного больше столбцов-переменных, чем строк-ограничений. Так, например, для классической транспортной задачи с m поставщиками и n потребителями матрица полных связей включает $M = n + m$ строк и $N = mn$ столбцов. Уже при двух сотнях поставщиков и потребителей количество переменных будет в сотни раз больше, чем количество ограничений, и их отношение будет характеризовать показатель

$$q = \frac{M}{N} = \frac{1}{m} + \frac{1}{n} < 0.01 \ll 1.$$

¹ Работа частично поддержана грантом РФФИ 13-07-12010 "Облачные и грид-технологии для транспортного моделирования".

Существенно меньшим будет такой показатель для балансовых соотношений в логистических многоиндексных транспортных задачах. Так, например, если пересылка потоков между поставщиками и потребителями реализуется через k промежуточных складов, то матрицу балансовых соотношений уже при $m = n \geq 30$ и $k = 10$ можно охарактеризовать показателем

$$q = \frac{M}{N} = \frac{m + n + k}{mnk} = \frac{1}{mn} + \frac{1}{nk} + \frac{1}{mk} < 0.01 \ll 1.$$

Следовательно, уже при нескольких десятках поставщиков, потребителей и складов количество переменных в трехиндексной задаче будет в сотни раз больше, чем количество ограничений. По мере увеличения поставщиков, потребителей и складов такое соотношение будет стремительно расти (по квадратичной зависимости).

Такого рода задачи послужили мотивацией для исследования алгоритмов нахождения допустимых точек систем линейных уравнений с вытянутыми матрицами (т.е. таких, в которых количество столбцов значительно превосходит количество строк) и двусторонними ограничениями на переменные. Второй аспект в этой мотивировке объясняется тем, что даже транспортная задача в матричной постановке является достаточно сложной в том случае, когда на объемы поставок выставлены границы снизу и сверху. Метод потенциалов (венгерский метод) для учета границ требует значительных ухищрений, что очень снижает его эффективность в сравнении с той, которую он демонстрирует в отсутствие таких границ. Симплекс-метод в задачах линейного программирования с двусторонними границами на переменные работает с отраженными (или перевернутыми) границами, что ограничивает длину шага в направлении базисных граничных переменных. Поэтому, если в оптимальном решении транспортной задачи имеется много активных границ, то для решения соответствующей задачи линейного программирования может потребоваться очень много симплексных итераций.

Указанные выше проблемы позволяет преодолеть изложенный ниже субградиентный метод для нахождения взвешенной проекции точки на линейное многообразие $Ax = b$ с вытянутой матрицей A и двусторонними ограничениями на компоненты неизвестного вектора x .

1. ЗАДАЧА ПРОЕКЦИИ И ДВОЙСТВЕННЫЙ АЛГОРИТМ

Пусть A – $n \times m$ -матрица с вещественными элементами. Будем считать, что матрица A плотно заполнена и $n \gg m$. Транспортно-производственные матрицы являются сильно разреженными и вписываются в указанные предположения с некоторой натяжкой. Однако, в МАТЛАБ-подобных языках реализация матрично-векторных операций в ряде случаев является более качественной для плотно-заполненных матриц, чем для разреженных матриц. Поэтому программная реализация алгоритма для общего случая, т.е. для плотно заполненной матрицы, может оказаться по временным характеристикам даже лучше, чем программная реализация этого же алгоритма для специального случая, т.е. для разреженной матрицы.

Пусть x^0 – заданный вектор из R^n (точка в евклидовом пространстве R^n). Взвешенной евклидовой проекции точки x_0 на линейное многообразие $Ax = b$ поставим в соответствие задачу квадратичного программирования:

найти

$$f^* = \min_{x \in R^n} (x - x^0)^T W (x - x^0) \quad (1)$$

при ограничениях

$$Ax = b, \quad (2)$$

$$x^{low} \leq x \leq x^{up}. \quad (3)$$

Здесь W – диагональная $n \times n$ -матрица, у которой все диагональные элементы w_1, \dots, w_n положительны; b – m -мерный вещественный вектор; x^{low} и x^{up} – n -мерные вещественные векторы (их конечнозначные компоненты задают значения нижних и верхних границ на переменные, соответственно).

Если ограничения (2)–(3) совместны, то задача (1)–(3) имеет единственное решение x^* , для которого

$$f^* = \sum_{i=1}^n w_i (x_i^* - x_i^0)^2, Ax^* = b, x_i^{low} \leq x_i^* \leq x_i^{up}, i = 1, \dots, n. \quad (4)$$

Из (4) видим, что в точке x^* достигается минимальное взвешенное отклонение (в евклидовой норме) от множества, заданного ограничениями (2)–(3), до фиксированной точки x^0 , с весами

равными $\sqrt{w_1}, \dots, \sqrt{w_n}$. В частном случае, если $w_1 = \dots = w_n = 1$ (все веса равны единице), то решением задачи (1)–(3) будет вектор x^* , наименее удаленный от точки x^0 , а если еще и $x^0 = 0$, то наименее удаленный от начала координат.

Если система ограничений (2)–(3) несовместна, то использование многих методов для решения задачи (1)–(3) затрудняется тем, что применять их приходится в два этапа. На первом этапе выясняется совместность системы (2)–(3), а на втором этапе решается задача квадратичного программирования (1)–(3). Избежать первого этапа позволяет двойственный алгоритм [3].

Двойственный алгоритм решает частично двойственную к (1)–(3) задачу, которая построена с помощью функции Лагранжа только для ограничений (2) в форме равенств. Такая двойственная задача связана с максимизацией вогнутой дифференцируемой функции

$$\psi(u) = \min_{x^{low} \leq x \leq x^{up}} \{(x - x^0)^T W (x - x^0) + u^T Ax - u^T b\}, \quad (5)$$

где $u \in R^m$ – множители Лагранжа, отвечающие ограничениям (2). Градиент и значение функции $\psi(u)$ в точке u вычисляются по формулам:

$$g_\psi(u) = Ax(u) - b \quad \text{и} \quad \psi(u) = (x(u) - x^0)^T D(x(u) - x^0) + u^T g_\psi(u),$$

где $x(u) = \min(\max(x^{low}, x^0 - 0.5W^{-1}A^T u), x^{up})$. Здесь $x(u)$ – аналитическое решение задачи минимизации по x в (5) и получено оно благодаря сепарабельности целевой функции в задаче (1)–(3).

Нахождению оптимальных множителей Лагранжа $u^* \in R^m$ отвечает задача оптимизации

$$\psi^* = \psi(u^*) = \max_{u \in R^m} \psi(u), \quad (6)$$

для решения которой могут быть применены градиентные методы. Если матрица A полноранговая, т.е. $rank(A) = m$, то оптимальный вектор u^* единственный, а если $rank(A) < m$, то вектор u^* определяется неоднозначно. Второй случай для градиентных методов даже лучше, так он позволяет методам сходиться к любому из оптимальных векторов u^* .

В результате решения задачи (6) возможны два исхода. Если система ограничений (2)–(3) совместна, то $\psi^* = f^*$ и вектор

$$x^* = x(u^*) = \min(\max(x^{low}, x^0 - 0.5W^{-1}A^T u^*), x^{up}) \quad (7)$$

является оптимальным решением задачи (1)–(3). Если система ограничений (2)–(3) несовместна, то $\psi^* = \infty$. Это может быть учтено за счет остана метода, если выполнено условие

$$\psi(\lambda) > f_{up} = x_{up}^T D x_{up}, \quad (8)$$

где $x_{up} = \max(\text{abs}(x^{up} - x^0), \text{abs}(x^0 - x^{low}))$. В условии (8) используется f_{up} – максимально возможное значение целевой функции (1), которое построено для конечнозначных компонент нижних и верхних границ на переменные. Именно для этого при постановке задачи (1)–(3) мы избегали на переменные границ, равных $+\infty$.

Формально для решения задачи (7) можно выбрать любой из градиентных алгоритмов. Мы выбрали r -алгоритм Шора [4], имеющий ускоренную сходимость при максимизации вогнутых (не обязательно гладких) функций с овражной структурой поверхностей уровня. Аргументом в его пользу послужило то, что некоторые из его модификаций хорошо проявили себя при решении негладких двойственных задач с количеством переменных порядка нескольких сотен. Одна из этих модификаций и положена в основу приведенной ниже программной реализации двойственного алгоритма.

2. ОСТАВЕ–ПРОГРАММА ДВОЙСТВЕННОГО АЛГОРИТМА

Двойственный алгоритм реализован как функция `dualr` на МАТЛАБ-подобном некоммерческом языке GNU Octave [5]. Программа либо находит достаточно хорошее приближение к решению $x^* = x(u^*)$ задачи (1)–(3), либо сообщает о несовместности системы ограничений (2)–(3). Ядром программы является octave-функция `galgb5`, которая реализует вариант r -алгоритма с постоянной величиной коэффициента растяжения пространства и адаптивной регулировкой шага в направлении нормированного антисубградиента. Эта регулировка направлена на увеличение точности поиска минимума функции по направлению в процессе счета и при этом гарантирует, что среднее (по итерациям) число шагов по направлению будет небольшим.

```

% Входные параметры:
% w,x0,A,rhs,xlow,xup - данные в задаче (1)-(3), где rhs=b
% alpha - коэффициент растяжения пространства переменных
% h0, q1 - параметры адаптивной регулировки величины шага
% epsu, epsg, maxitn - параметры останова r-алгоритма
% Выходные параметры:
% xr - найденное решение задачи (1)-(3)
% fr - значение функции в точке xr(1:n)
% ist - код останова ( определяет статус решения:
% 2,3 - решение оптимальное; 4,5 - решение не найдено;
% 6 - система ограничений (2)-(3) несовместна. )

```

К управляющим параметрам программы `dualr` отнесены только те параметры r -алгоритма, которые существенно влияют на его скорость сходимости при оптимизации гладких функций. Это коэффициент растяжения пространства α (`alpha`, рекомендуемые значения $2 \div 4$) и два параметра адаптивной регулировки шага: h_0 – величина начального шага (`h0`, рекомендуется установить равной 1.0, а затем уточнить ее значение по количеству одномерных спусков на первой итерации) и q_1 – коэффициент уменьшения шага (`q1`, рекомендуемые значения $0.8 \div 0.95$). Такой выбор значений для q_1 связан с тем, что дополнительное по отношению к растяжению пространства измельчение шага способствует увеличению точности поиска минимума функции по направлению, что при минимизации гладких функций обеспечивает более быструю скорость сходимости. Параметры адаптивной регулировки n_h и q_2 менее важны для гладких функций, их значения установлены такими, как рекомендует практика (`nh=3`, `q2=1.1`, смотри код программы ниже).

Статус решения `ist` определяется параметрами останова в r -алгоритме: он останавливается в точке u_{k+1} , если выполнено условие $\|u_{k+1} - u_k\| \leq \varepsilon_u$ (`epsu` – останов по аргументу, `ist=3`) или условие $\|g_\psi(u_{k+1})\| \leq \varepsilon_g$ (`epsg` – останов по норме градиента, `ist=2`, используется для гладких функций). Аварийное завершение программы связано либо с тем, что превышена величина `maxitn` – максимальное количество итераций (`ist=4`), либо начальный шаг h_0 слишком мал и его требуется увеличить (`ist=5`). Если в какой-либо

точке выполнено условие (8), то это служит сигналом о несовместности система ограничений (2)–(3) (ist=6).

```
# octave-code for dualr (version 1.0, 16.08.2013)
function [xr,fr,ist] = dualr(w,x0,A,rhs,xlow,xup,
                           alpha,h0,q1,epsu,epsq,maxitn);
itn=0; hs=h0; m=length(rhs); B=eye(m); u=zeros(m,1);
xs=max(abs(xlow-x0),abs(xup-x0)); fup=1.1*sum(xs.*xs.*w);
nfg = 1; xr=min(max(xlow,x0),xup); g0 = A*xr-rhs;
fr=sum((xr-x0).*(xr-x0).*w)+g0'*u; dgr=dg0=norm(g0);
printf("itn%4d f%14.6e fr%14.6e dg0%11.3e",itn,fr,fr,dg0);
printf(" dgr%11.3e ls%2d nfg%4d\n",dgr,0,nfg);
if (norm(g0) < epsu) ist = 2; return; endif
for (itn = 1:maxitn)
    du = B * (g1 = B' * g0)/norm(g1);
    d = 1; ls = 0; ddu = 0;
    while (d > 0)
        u += hs * du; ddu += hs * norm(du); nfg ++;
        xs=min(max(xlow,x0-0.5*A'*u./w),xup); g1=A*xs-rhs;
        dg1=norm(g1); f=sum((xs-x0).*(xs-x0).*w)+g1'*u;
        if (dg1 < dgr) fr = f; xr=x; dgr=dg1; endif
        if (f > fup) ist=6; return; endif
        if (dg1 < epsq) ist = 2; return; endif
        ls ++; (mod(ls,3)==0) && (hs *= 1.1);
        if (ls > 500) ist = 5; return; endif
        d = du' * g1;
    endwhile
    (ls == 1) && (hs *= q1);
    printf("itn%4d f%14.6e fr%14.6e dg1%11.3e",itn,f,fr,dg1);
    printf(" dgr%11.3e ls%2d nfg%4d\n",dgr,ls,nfg);
    if (ddu < epsu) ist = 3; return; endif
    xi = (dg = B' * (g1 - g0) )/norm(dg);
    B += (1 / alpha - 1) * B * xi * xi';
    g0 = g1;
endfor
ist = 4;
endfunction
```

При минимизации гладкой выпуклой функции, в том числе из существенно овражной структурой поверхностей уровня,

параметры останова $\varepsilon_u, \varepsilon_g \sim 10^{-8} \div 10^{-6}$ обеспечивают нахождение точки, в которой значение функции является достаточно близким к оптимальному. Кроме того, правильным подбором управляющих параметров из указанных выше диапазонов программу можно настроить под более быстрое решение семейства задач проекции, чем это позволяет рекомендуемый стандартный выбор управляющих параметров $\alpha = 2$, $h_0 = 1$ и $q_1 = 0.9$.

Программа `dualg` может быть использована для эффективного решения задач (1)–(3) в том случае, когда для хранения матрицы A требуются мегабайты оперативной памяти. Так, например, для плотно заполненной матрицы с размерами $m = 50$, $n = 1000000$, хранение которой потребовало 400 Мегабайт оперативной памяти, время счета не превысило девяти минут на 64-разрядном компьютере Intel Core 2 Duo с тактовой частотой 2,66 ГГц и 2 ГБ оперативной памяти [6]. Вычисления проводились в операционной системе Windows XP, в среде GNU Octave версии 2.9.15.

Использовать стандартное программное обеспечение решения задач квадратичного программирования для задач (1)–(3) с мегабайтными матрицами неэффективно, так как многие из этих программ требуют значительного объема оперативной памяти для хранения данных в QMPS-формате. Кроме того, эти программы нелогично использовать при решении задач специального вида, для которых специализированные методы всегда можно сделать более быстрыми, чем программы общего назначения. В специализированных методах легко избежать такой неприятности, как плохая обусловленность задач. В задаче (1)–(3) она связана с линейной зависимостью строк в матрице A , т.е. $rank(A) < m$.

Вышесказанное подтверждается опытом тестирования программы `dualg`. При ее использовании наблюдается устойчивость g -алгоритма по отношению к большому количеству переменных, нечувствительность к вырожденности задачи квадратичного программирования. Программа успешно справляется с задачами для вырожденных матриц, полученных очень слабым возмущением m коэффициентов в одноранговой матрице A ($rank(A) = 1$) [6]. Вырожденность свойственна и матричной транспортной задаче. Ее матрица имеет $n + m$ строк, а ранг равен $n + m - 1$. Ниже покажем, как с помощью программы `dualg` можно решить матричную транспортную задачу с ограничениями на объемы поставок.

3. КАК dual РЕШАЕТ ТРАНСПОРТНУЮ ЗАДАЧУ

Пусть c_i , $i=1, \dots, n$ – произвольные вещественные числа. Положим $x_i^0 = -\frac{c_i}{2w_i}$, где $w_i > 0$, $i=1, \dots, n$. Тогда целевая функция в задаче (1)–(3) имеет вид

$$f(x) = \sum_{i=1}^n w_i (x_i - x_i^0)^2 = \sum_{i=1}^n w_i x_i^2 + \sum_{i=1}^n c_i x_i + \sum_{i=1}^n \frac{c_i^2}{4w_i}.$$

Если выбирать веса $w_i = \varepsilon_i$, где ε_i – достаточно малые положительные числа, $i=1, \dots, n$, то с помощью задачи (1)–(3) можно найти сколь-угодно точное приближение оптимального решения x^* в задаче линейного программирования: найти

$$f_1^* = \min_{x \in R^n} \sum_{i=1}^n c_i x_i \quad (9)$$

при ограничениях

$$Ax = b, \quad (10)$$

$$x^{low} \leq x \leq x^{up}. \quad (11)$$

Задачу линейного программирования (9)–(11) можно заменить задачей взвешенной (с весами $\varepsilon_1, \dots, \varepsilon_n$) проекции точки $x^0 = -(c_1/\varepsilon_1, \dots, c_n/\varepsilon_n)/2$ на линейное многообразие $Ax = b$ при двусторонних ограничениях на переменные. Оптимальные значения целевых функций в обеих задачах связаны соотношением

$$f_1^* \approx f^* - \sum_{i=1}^n \frac{c_i^2}{4\varepsilon_i}.$$
 Недостаток такой замены состоит в том, что

целевая функция в задаче взвешенной проекции будет стремиться к большим величинам при малых значениях весов ε_i . Однако, в двойственном алгоритме этот недостаток в значительной мере нивелируется тем, что само значение функции в расчетных формулах не используется, а малые значения ε_i не оказывают такого большого влияния на множители Лагранжа.

Пусть m_i и n_i – количество потребителей и поставщиков, A_i – запасы продукта у i -го поставщика, $i=1, \dots, n_i$, B_j –

потребности в продукте у j -го потребителя, $j = 1, \dots, m_1$. Предполагается, что $\sum_{i=1}^{n_1} A_i = \sum_{j=1}^{m_1} B_j$. Для каждого поставщика i и потребителя j известны: c_{ij} – тариф на поставку единицы продукта, x_{ij}^{low} и x_{ij}^{up} – нижняя и верхняя границы на объем поставляемого продукта.

Сбалансированной матричной транспортной задаче с ограничениями на объемы поставок соответствует задача вида (1)–(3) в такой постановке:

$$f_{\varepsilon}^* = \min_{x_{ij}} \sum_{i=1}^{n_1} \sum_{j=1}^{m_1} \varepsilon \left(x_{ij} + \frac{c_{ij}}{2\varepsilon} \right)^2 \quad (12)$$

при ограничениях

$$\sum_{j=1}^{m_1} x_{ij} = A_i, \quad i = 1, \dots, n_1, \quad (13)$$

$$\sum_{i=1}^{n_1} x_{ij} = B_j, \quad j = 1, \dots, m_1, \quad (14)$$

$$x_{ij}^{low} \leq x_{ij} \leq x_{ij}^{up}, \quad i = 1, \dots, n_1, j = 1, \dots, m_1. \quad (15)$$

Здесь, для удобства, все ε_{ij} , $i = 1, \dots, n_1$, $j = 1, \dots, m_1$, положены равными одному и тому же $\varepsilon > 0$. При малых значениях ε оптимальное решение в задаче (12)–(15) будет “близким” к оптимальному решению матричной транспортной задачи с ограничениями на объемы поставок. Если для объемов поставок выбрать границы $x_{ij}^{low} = 0$ и $x_{ij}^{up} = \min\{A_i, B_j\}$, $i = 1, \dots, n_1$, $j = 1, \dots, m_1$, то приходим к классической транспортной задаче в матричной постановке.

Рассмотрим пример транспортной задачи с тремя поставщиками и четырьмя потребителями², который взят из сайта <http://matica.org.ua> по решению контрольных работ по математике. В примере matica-3-4 матрица полных связей A имеет $7 = 3+4$ строк и $12 = 3*4$ столбцов. Целевая функция (определяется матрицей тарифов), вектор правых частей (равен запасам продукта у поставщиков и потребностям в продукте у потребителей) и оптимальное решение задачи являются такими:

² <http://matica.org.ua/primeri/reshenie-transportnoy-zadachi-metodom-potensialov>

$$\begin{aligned}
 c &= [7 \ 8 \ 1 \ 2 \ 4 \ 5 \ 9 \ 8 \ 9 \ 2 \ 3 \ 6]', \\
 b &= rhs = [200 \ 180 \ 190 \ 150 \ 130 \ 150 \ 140]', \\
 xopt &= [0 \ 0 \ 60 \ 140 \ 150 \ 30 \ 0 \ 0 \ 0 \ 100 \ 90 \ 0]'.
 \end{aligned}$$

Оптимальное решение единственно, и на нем достигаются минимальные затраты на транспортировку продукции

$$Z_{min} = 1 * 60 + 2 * 140 + 4 * 150 + 5 * 30 + 2 * 100 + 3 * 90 = 1560.$$

Таблица 1. Затраты на транспортировку продукции в примере matica-3-4 при разных ε и *temp1*

<i>temp1</i>	Значения ε в задаче (12)–(15)					
	1.00000	0.10000	0.01000	0.00100	0.00010	0.00001
150	2989.0	2650.0	1595.0	1560.0	1560.0	1560.0
130	2989.0	2650.0	1759.5	1730.0	1730.0	1730.0
110	2989.0	2650.0	1922.3	1910.0	1910.0	1910.0
90	2989.0	2650.0	2102.5	2090.0	2090.0	2090.0
70	2989.0	2650.0	2330.0	2330.0	2330.0	2330.0
50	3047.4	3046.3	3040.0	3040.0	3040.0	3040.0

В таблице 1 приведены затраты на транспортировку продукции, которые получены в результате решения задачи (12)–(15) при шести значениях ε и одинаковых по величине границах для объемов поставок. Нижняя граница выбиралась равной нулю, а верхние границы устанавливались равными одному и тому же числу *temp1*, которое изменялось от 150 (максимальное количество потока в оптимальном решении хорт) до 50 с интервалом в 20 единиц. Если верхние границы выбрать меньшими, чем 50 единиц, то транспортная задача для примера matica-3-4 не имеет решения.

Из таблицы видим, что начиная для значений $\varepsilon = 10^{-3}$ и меньших затраты на транспортировку продукции в оптимальных решениях задачи (12)–(15) совпадают. Это означает, что оптимальное решение задачи (12)–(15) равно оптимальным поставкам продукта в транспортной задаче. Это равенство выполняется с точностью ε_g – останов по норме градиента в программе dualr. В таблице 2 приведены оптимальные объемы транспортируемого продукта при $\varepsilon = 0.0001$ и всех значениях *temp1* = [150 130 110 90 70 50] из таблицы 1.

Таблица 2. Оптимальные объемы транспортируемого продукта в примере matic-a-3-4 при $\varepsilon = 10^{-4}$ и всех $temp1$ из таблицы 1

(i,j)	Значения $temp1$ в задаче (12)–(15)					
	150	130	110	90	70	50
(1,1)	0.00	2.50	17.50	32.50	60.00	50.00
(1,2)	0.00	0.00	0.00	0.00	0.00	50.00
(1,3)	60.00	67.50	72.50	77.50	70.00	50.00
(1,4)	140.00	130.00	110.00	90.00	70.00	50.00
(2,1)	150.00	130.00	110.00	90.00	70.00	50.00
(2,2)	30.00	40.00	40.00	40.00	60.00	30.00
(2,3)	0.00	0.00	0.00	0.00	10.00	50.00
(2,4)	0.00	10.00	30.00	50.00	40.00	50.00
(3,1)	0.00	17.50	22.50	27.50	20.00	50.00
(3,2)	100.00	90.00	90.00	90.00	70.00	50.00
(3,3)	90.00	82.50	77.50	72.50	70.00	50.00
(3,4)	0.00	0.00	0.00	0.00	30.00	40.00
Z_{\min}	1560.0	1730.0	1910.0	2090.0	2330.0	3040.0

Таблица 3. Полное время (в секундах) решения транспортных задач с десятками поставщиков и потребителей при разных ε

(m_1, n_1)	Значения ε в задаче (12)–(15)					
	0.01	0.001	0.0001	0.00001	0.000001	0.0000001
(10,10)	0.10315	0.11883	0.13832	0.15390	0.17533	0.16914
(20,20)	0.18037	0.28046	0.25872	0.28093	0.32727	0.33257
(30,30)	0.47860	0.58045	0.80859	0.97012	1.26851	1.31829
(40,40)	1.31219	1.97510	2.18888	2.52609	2.81282	3.54913

В таблице 3 указаны времена, затраченные на решения транспортных задач с несколькими десятками поставщиков и потребителей и ненулевыми ограничениями на переменные. Тарифы генерировались случайными целыми числами из диапазона [1,10], что обеспечивало неоднозначность отыскиваемого оптимального решения. Запасы и потребности в продукте вычислялись так, чтобы случайно сгенерированная точка с целочисленными компонентами из диапазона [1,10] была допустимой для ограничений (13)–(14). Нижние и верхние границы вычислялись умножением компонент допустимого решения на 0.5 и 1.5 соответственно. Вычисления проводились на Pentium E5200 и показали, что решение транспортных задач с несколькими

десятками поставщиков и потребителей требует нескольких секунд на современных ПЭВМ и слабо зависит от малости величины ε .

Заключение. Транспортная задача может быть успешно заменена задачей взвешенной проекции на линейное многообразие при малых значениях параметра ε , а программа `dualg` может быть использована для решения транспортных задач при ограничениях на объемы поставок. С помощью этой программы можно усилить учебные курсы по математическим методам линейного программирования, позволяющим получать количественное обоснование для выбора оптимальных решений транспортных задач [7,8].

Для решения реальных транспортных задач двойственный алгоритм может быть существенно ускорен. Так, например, специальная структура матрицы ограничений позволяет сделать его зависящим от меньшего количества двойственных переменных, их количество равно $\min(m, n)$, а не $n + m$ как при использовании программы `dualg`. Кроме того, для транспортных задач не требуется хранить даже разреженную структуру матрицы, так как все коэффициенты в ней равны единице и расположены по определенному правилу. Учет этих особенностей матрицы ограничений в сочетании с применением быстро сходящихся вариантов субградиентных методов способен значительно ускорить двойственный алгоритм при решении транспортных задач.

Используемый при построении двойственного алгоритма подход можно перенести на задачи со строго выпуклыми сепарабельными нелинейными функциями. Если аналитическое решение одномерных подзадач получить не удастся, то его можно заменить дихотомией для каждой переменной x_i , что незначительно замедлит двойственный алгоритм, построенный на основе r -алгоритма. Это позволяет создать комплекс программ, который ориентирован на работу с моделями транспортно-производственного вида и может быть востребован логистическими компаниями или фирмами-разработчиками программного обеспечения для этих компаний. Краткость кодов таких программ позволяет легко встраивать их в любые оболочки, в том числе в часто используемую надстройку «Поиск решения» в Microsoft Excel.

Простые матрично-векторные операции r -алгоритма делают его перспективным при реализации методов решения

мегабайтных систем линейных уравнений с двусторонними границами на переменные в системах параллельных или распределенных вычислений. Существующие библиотеки стандартных программ для параллельных матрично-векторных операций позволяют в краткие сроки адаптировать такие алгоритмы для эффективной работы с использованием векторных процессоров на основе графических ускорителей (GPU) и других.

Литература.

1. Михалевич В.С., Трубин В.А., Шор Н.З. Оптимизационные задачи производственно-транспортного планирования. – М.: Наука, 1986. – 264 с.
2. Шор Н.З., Соломон Д.И. Декомпозиционные методы в дробно-линейном программировании. – Кишинев: Штиинца, 1989. – 204 с.
3. Стецюк П.И. О решении системы линейных уравнений с двусторонними ограничениями на переменные // Алгебра и линейная оптимизация. Тезисы международной конференции, посвященной 100-летию С.Н.Черникова. Екатеринбург, 14-19 мая 2012 года. – Изд-во "УМЦ-УПИ", Екатеринбург, 2012. – С. 155–157.
4. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. – Киев: Наукова думка, 1979. – 200 с.
5. Octave [Электронный ресурс]: <http://www.octave.org> – Режим доступа: свободный.
6. Стецюк П.И., Ивличев А.В. Тестовые эксперименты с g-алгоритмом для мегабайтных систем линейных уравнений с двухсторонними границами на переменные // VI-а международная школа-семинар "Теория принятия решений", Ужгород, Украина, 1–6 октября 2012 года. Тезисы докладов. – С. 186-187.
7. Гамецкий А.Ф., Соломон Д.И. Исследование операций. Том II. – Кишинев, Еврика, 2008. – 592 с.
8. Широков А.П. Математические модели и методы в управлении транспортными системами. Учебно-методическое пособие. В 2-х частях. Часть 2. Решение транспортных задач методами линейного программирования. – Хабаровск: ДВГУПС, 1999. – 51 с.