ТРАНСПОРТНАЯ ЗАДАЧА И ОРТОГОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ НА ЛИНЕЙНЫЕ МНОГООБРАЗИЯ[1]

П.И. СТЕЦЮК, Е.А. НУРМИНСКИЙ, Д.И. СОЛОМОН,
Институт кибернетики им. В.М.Глушкова НАН Украины, Киев, Украина,
Институт автоматики и процессов управления ДВО РАН, Владивосток, Россия,
Академия транспорта, информатики и коммуникаций, Кишинеу, 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 переменных будет в сотни раз больше, чем ограничений, и их отношение характеризует показатель

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

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

Существенно меньшим будет подобный 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 будет характеризовать показатель

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

уже при $m = n \geq 30$ и всего $k = 10$ складах. 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. когда количество столбцов значительно превосходит количество строк, и двусторонними границами на переменные. Второй аспект в этой мотивировке обьясняется тем, что даже транспортная задача в матричной постановке является достаточно сложной, если на объемы поставок выставлены границы снизу и сверху. Метод потенциалов (венгерский метод) для учета границ требует значительных ухищрений и его эффективность далека от той, которую он демонстрирует в отсутствии таких границ. Симплекс метод для задач линейного программирования с двусторонними границами на переменные работает с отраженными (или перевернутыми) границами. Это ограничивает шаги в направлении базисных граничных переменных. Поэтому, если в транспортной задаче много активных границ в оптимальном решении, 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 границами 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 $n >> m$. 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. плотно заполненной матрицы, может оказаться по временным характеристикам даже лучше, чем программная реализация этого же алгоритма для специального случая, т.е. 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 $Ax = b$ сопоставим задачу квадратичного программирования:

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

при ограничениях

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

$$x^{low} \le x \le x^{up}. \qquad (3)$$

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

Если система ограничений (2)–(3) совместна, то задача (1)–(3) имеет единственное решение. Им есть вектор x^* такой, что

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

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

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

Если система ограничений (2)–(3) может быть 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. Двойственная задача связана с максимизацией вогнутой дифференцируемой функции

$$\psi(u) = \min_{x^{low} \le x \le x^{up}} \left\{ (x - x^0)^T W (x - x^0) + u^T A x - u^T b \right\}, \qquad (5)$$

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

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

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

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

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

В результате решения задачи (6) возвожны 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 (2)–(3) несовместна, то $\psi^* = \infty$. Это может быть учтено остановом метода, если выполнено условие

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

где $x_{up} = \max(abs(x^{up} - x^0), abs(x^0 - x^{low}))$. Условие (8) использует 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 проекции 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 обкатаны на 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. OCTAVE–ПРОГРАММА ДВОЙСТВЕННОГО АЛГОРИТМА

Двойственный алгоритм реализован как функция dualr на МАТЛАБ-подобном некоммерческом языке GNU Octave [5]. Программа либо находит достаточно хорошое приближение к $x^* = x(u^*)$ – решению задачи (1)–(3), либо сообщает о несовместности системы ограничений (2)–(3). Ядром программы является octave-функция ralgb5, которая реализует вариант r-алгоритма с постоянным коэффициентом растяжения пространства (обозначается α) 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. )
```

В управляющие параметры 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÷4) и два параметра от адаптивной регулировки шага: h_0 – величина начального шага (h0, рекомендуется установить равным 1.0, а затем уточнить его значение по количеству одномерных спусков на первой итерации) и q_1 – коэффициент уменьшения шага (q1, рекомендуемые значения 0.8÷0.95). Такой выбор 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 умалчиваемые 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 $\| u_{k+1} - u_k \| \le \varepsilon_u$ (epsu – останов по аргументу, ist=3); или условие $\| g_\psi(u_{k+1}) \| \le \varepsilon_g$ (epsg – останов по норме градиента, ist=2, используется для гладких функций). Аварийное завершение программы связано либо с тем, что превышено maxitn – максимальное количество итераций (ist=4), либо начальный шаг h_0 слишком мал и его требуется увеличить (ist=5). Если в какой-то

точке выполнено условие (8), то оно 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,epsg,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=xs; dgr=dg1; endif
    if (f > fup) ist=6; return; endif
    if (dg1 < epsg) 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
```

Параметры останова $\varepsilon_u, \varepsilon_g \sim 10^{-8} \div 10^{-6}$ при минимизации гладкой выпуклой функции даже существенно овражной структуры обеспечивают нахождение точки со значением функции,

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 dualr может быть использована для эффективного решения задач (1)–(3) с мегабайтными матрицами, т.е. 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 для задач специального вида, где специализированные методы всегда можно сделать более быстрыми, чем программы общего назначения. В специализированных методах легко обойти такие неприятности, как плохая обусловленность задач. В задаче (1)–(3) она связана с линейной зависимостью строк в матрице A, т.е. $rank(A) < m$.

Сказанное подтверждает опыт тестирования программы dualr. На ней наблюдается устойчивость r-алгоритма по отношению к выбору большим количества 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 ее равен $n+m-1$. Ниже покажем как с помощью программы dualr можно решить матричную транспортную задачу с ограничениями на объемы поставок.

3. КАК dualr РЕШАЕТ ТРАНСПОРТНУЮ ЗАДАЧУ

Пусть c_i, $i=1,\dots,n$ – произвольные вещественные числа. Положим $x_i^0 = -\dfrac{c_i}{2w_i}$, $i=1,\dots,n$, 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 сколь-угодно точно приблизить x^* – оптимальное решение задачи линейного программирования:

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

при ограничениях

$$Ax = b, \tag{10}$$

$$x^{low} \le x \le x^{up}. \tag{11}$$

Значит, 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 границах на переменные. Оптимальные значения целевых функций в обеих задачах связаны соотношением $f_1^* \approx f^* - \sum_{i=1}^{n} \dfrac{c_i^2}{4\varepsilon_i}$ и недостатком такой замены есть то, что целевая функция в задаче взвешенной проекции будет стремиться к большим величинам при малых значениях весов ε_i. Однако, этот недостаток в двойственном алгоритме в значительной мере нивелируется тем, что само значение функции в расчетных формулах не участвует, а малые значения ε_i такого большого влияния на множители Лагранжа не оказывают.

Пусть m_1 и n_1 – количество потребителей и поставщиков, A_i – запасы продукта у поставщиков $i=1,\dots,n_1$, B_j –

потребности в продукте у потребителей, $j=1,\dots,m_1$. 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 отвечает задача вида (1)–(3) в такой постановке:

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

при ограничениях

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

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

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

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 близко к оптимальному решению матричной транспортной задачи с ограничениями на объемы поставок. Если на обьемы 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 тогда имеем решение классической транспортной задачи в матричной постановке.

Рассмотрим пример транспортной задачи с тремя поставщиками и четырьмя потребителями[2] 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

[2] http://matica.org.ua/primeri/reshenie-transportnoy-zadachi-metodom-potentsialov

продукте у потребителей) и оптимальное решение являются следующими

```
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]'.
```

Оптимальное решение единственное 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 траспортировку продукции в примере matica-3-4 при разных ε и *temp*1

*temp*1	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 траспортировку продукции, которые получены как решение задачи (12)–(15) при шести значениях ε и одинаковых по величине границах на обьемы поставок. Нижняя граница выбиралась равной нулю, а верхние границы устанавливались равными одному и тому же числу *temp*1, которое изменялось от 150 (максимальное количество потока в оптимальном решении xopt), до 50 с интервалом в 20 единиц. Если верхние границы выбрать ниже, чем 50 единиц, то транспортная задача для примера matica-3-4 не имеет решения. Из таблицы видим, что начиная с $\varepsilon = 10^{-3}$ и ниже затраты на транспортировку продукции в оптимальных точках задачи (12)–(15) совпадают. Это означает, что оптимальное решение задачи (12)–(15) равно оптимальным поставкам продукта в транспортной задаче с точностью ε_g – останов по норме градиента в программе dualr. В таблице 2 даны оптимальные объемы транспортируемого продукта при значении параметра $\varepsilon = 0.0001$ и всех значениях temp1 = [150 130 110 90 70 50] из таблицы 1.

Таблица 2. Оптимальные обьемы транспортируемого продукта в примере matica-3-4 при $\varepsilon = 10^{-4}$ и всех *temp*1 из таблицы 1

(I,j)	Значения *temp*1 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 приведены времена решения транспортных задач с несколькими десятками поставщиков и потребителей и ненулевыми границами на переменные. Тарифы генерировались случайными целыми числами из диапазона [1,10], что обеспечивало неоднозначность оптимального решения. Запасы и потребности в продукте вычислялись так, чтобы случайно сгенерованная 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 показывают, 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 dualr может быть использована в учебных целях для решения транспортных задач при ограничениях на объемы поставок. С ее помощью 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 $min(m,n)$ двойственных переменных, а не от $n+m$ переменных, как при использовании программы dualr. Кроме того, здесь не требуется хранить разреженную структуру матрицы, так как все коэффициенты в ней равны единице и расположены в матрице по определенному правилу. Эти особенности в сочетании с быстро сходящимися вариантами субградиентных методов способны значительно ускорить двойственный алгоритм при решении транспортных задач.

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

Простые матрично-векторные операции r-алгоритма делают его перспективным для реализации методов решения мегабайтных систем линейных уравнений с двусторонними границами на переменные в системах параллельных или

распределенных вычислений. Существующие библиотеки стандартных программ для параллелельных 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 r-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. – Кишинеу, 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.