УДК 519.852.61

Алгоритм симплекс-метода с использованием двойного базиса

Г.И. Забиняко

Институт вычислительной математики и математической геофизики Сибирского отделения Российской академии наук, просп. Акад. М.А. Лаврентьева, 6, Новосибирск, 630090
E-mail: zabin@rav.sscc.ru

Забиняко Г.И. Алгоритм симплекс-метода с использованием двойного базиса // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2015. — Т. 18, № 4. — С. 349–359.

Рассматривается алгоритм симплекс-метода, в котором на итерациях не требуется в явном виде обновление LU-разложений. Решения, полученные с фиксированными факторами LU, корректируются с помощью небольших вспомогательных матриц. Приводятся результаты численных экспериментов.

DOI: 10.15372/SJNM20150401

Ключевые слова: *LU-разложения, обновление разложений, разреженные матрицы, симплекс-метод, линейное программирование.*

Zabinyako G.I. An algorithm of the simplex method using a dual basis // Siberian J. Num. Math. / Sib. Branch of Russ. Acad. of Sci. — Novosibirsk, 2015. — Vol. 18, № 4. — P. 349–359.

An algorithm of the simplex method not requiring an explicit updating of the LU decomposition in iterations is considered. Solutions obtained with fixed LU factors are corrected using small auxiliary matrices. The results of numerical experiments are presented.

Keywords: *LU-decomposition, decomposition updating, sparse matrices, simplex method, linear programming.*

1. Введение

Рассмотрим применение модифицированного симплекс-метода [1] для решения задачи линейного программирования (ЛП):

$$\begin{aligned} \text{минимизировать} \quad & c^\top x \\ \text{при условиях} \quad & Ax = b, \\ & \alpha \le x \le \beta, \end{aligned}$$

где A — матрица размера $m \times n$, векторы c, x, α, $\beta \in R^n$, $b \in R^m$, $n \ge m$. В оптимизирующей последовательности метода переменные подразделяются на m базисных и $n - m$ небазисных. Небазисные переменные всегда принимают одно из своих граничных значений. Базисным переменным соответствует базисная матрица, состоящая из m независимых столбцов матрицы A.

На итерациях метода происходят изменения базисной матрицы $B_{k+1} = B_k + (a_j - a_i)e_p^\top$, где e_p — p-й единичный орт из R^m, а столбец a_i, который занимал в базисе p-е место, заменяется столбцом a_j. Матрица B_k используется для определения направления y изменения исходных переменных и двойственных переменных π из линейных

систем $B_k y = a_j$ и $B_k^\top \pi = c_k$, где c_k — целевые коэффициенты, отвечающие базисным переменным.

Эффективность и устойчивость метода существенно зависит от способа обновления LU-разложения на итерациях. В модифицированном семплекс-методе выделяется этап так называемого перепостроения обратных матриц, когда заново производится LU-разложение текущей базисной матрицы.

Бартелс и Голуб предложили численно устойчивый алгоритм (BG) обновления LU на итерациях [2]. Предположим, что произведено разложение $B_0 = LU$, и на следующей итерации вместо a_i на p-е место вводится небазисный столбец a_j и $B_1 = B_0 + (a_j - a_i)e_p^\top$. В результате в матрице U на p-м месте окажется столбец $\gamma = L^{-1}a_j$, который нарушает верхний треугольный вид матрицы. Переставив столбец γ на m-е место и передвинув влево столбцы, начиная с $p+1$, получим верхнюю матрицу Хессенберга. Далее с помощью серии элементарных преобразований $M_k, k = p+1, \ldots, m$, обнуляются элементы $u_{p+1\ p+1}, \ldots, u_{mm}$, так что $M_m \cdots M_{p+1} L^{-1} B_1 = \bar{U}$, $\bar{U}$ — верхняя треугольная матрица. Множитель M_k получается из единичной матрицы размера $m \times m$, у которой в столбцах $k-1$ и k расположена матрица

$$\tilde{M}_k = \begin{bmatrix} 1 & 0 \\ -u_{kk}/\bar{u}_{k-1}, & 1 \end{bmatrix}$$

размера 2×2, если не производится перестановка;

$$\tilde{M}_k = \begin{bmatrix} 0 & 1 \\ 1 & -\bar{u}_{k-1,k}/u_{kk} \end{bmatrix},$$

если производится перестановка ($\bar{u}_{k-1,k}$ получен из исходного $u_{k-1,k}$ умножением на некоторые M_ν для $\nu < k$). В обновленной матрице U могут возникать новые ненулевые элементы, кроме элементов, отвечающих вектору γ. По этой причине в реализации алгоритма BG используются сложно организованные структуры данных, возникает необходимость в очистке мусора.

Форест и Томлин предложили [3] более удобный в реализации алгоритм (FT) обновления LU. Алгоритм FT можно получить из выше рассмотренного алгоритма BG, если на каждом шаге делать перестановки [1]. В результате в матрице U обнуляется p-я строка и строится только один строчный мультипликатор R из условия $U^\top R = u_{pp} e_p^\top$.

В [4] производится сопоставление алгоритмов обновления BG и FT. Алгоритм FT уступает алгоритму BG в численной устойчивости, при его использовании приходится чаще обращаться к перепостроению LU. В то же время алгоритм FT проще в реализации и на современных вычислительных системах позволяет провести векторизацию вычислительного процесса [5].

В работе [6] дается обоснование симплекс-метода, в котором нет необходимости производить явным образом обновление факторов на итерациях. Зафиксируем базисную матрицу B_0, которая использовалась на этапе перепостроения обратных матриц.

Матрицу B_k после замены k столбцов в B_0 на новые столбцы, не принадлежащие B_0, можно представить в следующем виде: $B_k = B_0 + (V_k - B_0 U_k^\top) U_k$, где матрица V_k размера $m \times k$ состоит из столбцов $a_{j1}, \ldots, a_{jk}$, а матрица U_k размера $k \times m$ составлена из строк $e_{p1}^\top, \ldots, e_{pk}^\top$. На основании теоремы Крона в [6] устанавливается, что решение системы $B_k x = b$ можно получить из соотношений $y = Q_k^{-1} U_k B_0^{-1} b$, $x = B_0^{-1}(b - V_k y)$; аналогично решение системы $\pi B_k = c$ можно получить из соотношений $y = c B_0^{-1} V_k Q_k^{-1}$, $\pi = (c - y U_k^\top) B_0^{-1}$, матрица $Q_k = U_k B_0^{-1} V_k$.

В работе [6] рассмотрена также программная реализация, в которой на каждой итерации требуется четыре обращения к B_0^{-1} вместо двух обращений к B_k^{-1} в стандартном модифицированном симплекс-методе. Кроме того, на итерациях рекуррентным образом вычислялась Q_k^{-1}, что, конечно, не позволяет говорить о численной устойчивости.

В [7] рассматривается другая реализация симплекс-метода с использованием двойного базиса. Здесь рассмотрены два варианта обращения матрицы Q_k: в явном виде и в виде разложения матрицы Q_k на факторы LU. На каждой итерации также требуется четыре обращения к B_0^{-1}.

В работе [8] предложена эффективная программная реализация метода с использованием подхода [6]. Разложим B_k на множители следующим образом:

$$B_k = \begin{pmatrix} B_0 & V_k \\ U_k & 0 \end{pmatrix} = \begin{pmatrix} B_0 & 0 \\ U_k & Q_k \end{pmatrix} \begin{pmatrix} I & Y_k \\ 0 & I \end{pmatrix},$$

где $B_0 Y_k = V_k$, $Q_k = U_k Y_k$.

Таким образом, предлагается запоминать разложения по B_0 для вновь поступающих столбцов из V_k. Это обеспечивает сокращение в два раза числа обращений к B_0^{-1} по сравнению с ранее упомянутыми вариантами. В [8] утверждается, что метод с двойным базисом обеспечивает такую же устойчивость, как и использование обновления BG, если B_0 хорошо обусловлен. Авторы работы [8] на практике показали, что алгоритм хорошо векторизуется при исполнении на современных вычислительных системах, что затруднительно получить для обновления BG (см. строение множителей M_k). В [8] корректировку решений с помощью матрицы Q_k предлагается обозначать как SC-обновление (Schur complement update). Для перепостроения LU-разложения обычно используется некоторая модификация критерия Марковица [9].

В работе [10] дается подробный анализ алгоритмов обновления, рассматривается применение обновления SC к задачам квадратичного программирования.

Далее в п. 2 рассматривается алгоритм решения задач ЛП с разреженными матрицами. Вначале рассматривается применение SC-обновления на итерациях, которое в основном следует схеме, предложенной в [8]. Перепостроение LU основано на идеях [11], при этом критерий численной устойчивости имеет приоритет перед требованием разреженности. Такое сочетание позволяет получить надежный и экономичный по затратам машинного времени алгоритм решения задач ЛП. В п. 3 приводятся результаты численных испытаний.

2. Алгоритмы

Вначале рассматривается применение обновления SC на итерациях метода, затем алгоритмы перепостроения LU. Определение ведущих элементов на этапе перепостроения LU производится из решения задачи назначения для матрицы, специальным образом построенной из элементов текущей базисной матрицы. Для приближенной минимизации заполненности мультипликаторов определяются симметричные перестановки с помощью процедур пакета METIS. Согласно полученной очередности ведущих элементов строится мультипликативное представление LU-разложения.

2.1. Выполнение итераций с обновлением SC

В конце k-й итерации после перепостроения LU имеем матрицы B_0, Q_k, вектор базисных переменных x_0^k и вектор двойственных оценок π^k. Пусть уже определен вектор a_j, который требуется ввести в базис.

Определим направления y изменения базисных переменных из следующих соотношений:

$$B_0 w = a_j,$$
$$Q_k y_1 = U_k w,$$
$$y = w - Y_k y_1.$$

Используя вектор y, определим вектор a_i, который подлежит исключению из базиса, и вектор базисных переменных x_0^{k+1}. Информация о векторах a_j и a_i позволяет обновить матрицу Q_k. После обновления Q_{k+1} вычислим двойственные оценки π^{k+1} из уравнений:

$$Q_{k+1}^\top z = Y_k^\top c^{k+1},$$
$$B_0^\top \pi^{k+1} = c^{k+1} - U_{k+1}^\top z,$$

где c^{k+1} — целевые коэффициенты, отвечающие новому набору базисных переменных (вектор a_i заменим на a_j). Двойственные оценки позволяют определить новый вектор a_j для ввода в базис, в случае если такого вектора не существует, то процесс решения завершается.

Рассмотрим изменения матрицы $Q_k = U_k B_0^{-1} V_k$ при замене базисного столбца a_i на небазисный столбец a_j. Для удобства изложения введем матрицу V_k^2, которая сформирована из столбцов, исключаемых из базиса. Всего возможно четыре различных случая обновления Q_{k+1} при изменении базиса $B_{k+1} = B_k + (a_j - a_i)e^\top p$, где e_p — p-й орт из R^m, если a_i занимал в базисе p-е место.

1. Столбец a_j не принадлежит B_0, а столбец a_i принадлежит множеству столбцов B_0 и занимает в базисе p-е место. В матрице Q_k добавляются новые строка и столбец:

   $$U_{k+1} = \begin{pmatrix} U_k \\ e_p^\top \end{pmatrix}, \quad Y_{k+1} = (Y_k, w),$$
   $$Q_{k+1} = U_{k+1}\ B_0^{-1} V_{k+1} = \begin{pmatrix} Q_k & U_k w \\ e_p^\top Y_k & \delta \end{pmatrix},$$

   где $B_0 w = a_j$, $\delta = e_p^\top w$.

2. Столбец a_j не принадлежит B_0 и столбец a_i не принадлежит B_0. В V_k столбец a_i заменяется на a_j. В матрице Q_k соответствующий столбец заменяется на $U_k w$, а в Y_k столбец заменяется на w. Размерность Q_k не меняется.

3. Столбец a_j принадлежит B_0 (на предыдущих итерациях он был исключен из текущего базиса) и столбец a_i принадлежит B_0. Предположим, что a_j занимает p-й столбец V_k^2. Тогда строка в Q_k заменяется p-й строкой Y_k; Y_k и V_k не изменяются.

4. Столбец a_j принадлежит B_0, а столбец a_i не принадлежит B_0 и занимает в V_k p-е место. Предположим, что a_j — q-й столбец V_k^2. Тогда из Q_k удаляются p-я строка и p-й столбец. В матрице U_k удаляется p-я строка, а q-я строка заменяется p-й строкой. Дополнительно в матрице Q_k делаются перестановки: на место удаленных строки и столбца переставляются последние строка и столбец. Из матрицы Y_k удаляется столбец, отвечающий вектору a_i. Размеры матрицы Q_{k+1} уменьшаются на единицу.

Для решения уравнений с матрицей Q_k используется программа LUmod [12]. С помощью LUmod поддерживается разложение $L_k Q_k = U_k$, где L_k и Q_k — плотные квадратные

матрицы, L_k — произведение матриц размера 2×2 стабилизированных исключений, U_k — верхняя треугольная матрица.

Векторы $w = B_0^{-1}a_j$ (из векторов w формируются матрицы Y_k) приводятся к компактной форме. В поле вещественных чисел с двойной точностью, отведенном под запись w, вначале записываются целые числа. В первом восьмибайтовом слове запоминаются число ненулевых компонент и номер, которому соответствует w в множестве столбцов V_k. Затем перечисляются номера ненулевых компонент w и далее перечисляются ненулевые компоненты w.

2.2. Перепостроение LU-разложения

Использование SC-обновления на итерациях позволяет упростить задачу перепостроения обратных матриц, использовать более простые структуры данных. В нашем случае для L- и U-мультипликаторов используется стандартный разреженный столбцовый формат [13].

Поиск номеров ведущих элементов осуществляется на основе построения трансверсали из условия максимизации произведения модулей ее элементов [11]. Множество пар индексов из трансверсали обозначим через T. Если $|T| = m$, то, используя набор индексов T, можно построить матрицу перестановок P по правилу $p_{ij} = 1$ для $i, j \in T$ и $p_{ij} = 0$, если $i, j \notin T$. Тогда матрица $\bar{A} = P^\top A$ имеет главную диагональ, составленную из элементов a_{ij} для $i, j \in T$.

Задача построения трансверсали T^*, отвечающей максимуму $\Pi_{i,j\in T}|a_{ij}|$, сводится к решению задачи назначения. Рассмотрим случай выбора ведущих элементов в столбцах. Сформируем вектор $\bar{a}$ из компонент $\bar{a}_j = \max_{i=1,\dots,m}|a_{ij}|$ для $j = 1,\dots,m$. Построим вспомогательную матрицу C с элементами:

$$c_{ij} = \begin{cases} \log \bar{a}_j - \log|a_{ij}|, & \text{если } a_{ij} \neq 0, \\ +\infty & \text{иначе.} \end{cases}$$

Задача определения трансверсали сводится к решению следующей задачи назначения:

$$\begin{aligned} \text{найти} \qquad & \min \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} \\ \text{при ограничениях} \qquad & \sum_{j=1}^{m} x_{ij} = 1,\ i = 1,\dots,m; \\ & \sum_{i=1}^{m} x_{ij} = 1,\ j = 1,\dots,m; \\ & x_{ij} \in \{0,1\} \text{ для } i, j = 1,\dots,m. \end{aligned}$$

Для решения задачи назначения используется алгоритм [14]. На сайте [15] доступна соответствующая программа для целочисленных матриц. Эта программа нами модифицирована для случая вещественных матриц.

После определения номеров ведущих элементов из решения задачи назначения с помощью процедур пакета METIS [16] определяются симметричные перестановки для приближенной минимизации заполненности мультипликативного представления LU. Фактически симметричные перестановки задают очередность использования ведущих элементов $a_{ij} \in T^*$.

Построение L- и U-мультипликаторов сопровождается контролем уровня ведущих элементов $|a_{ij}^k| \geq u\ a_{\max}^k$, где $0 < u \leq 1$, $a_{\max}^k = \max_{i \in K} a_{ij}^k$, K — множество индексов, которые не использовались в назначении ведущих элементов. Если для некоторых столбцов j неравенство не выполняется, то построение мультипликаторов из этого столбца откладывается. В результате может получиться некоторый перечень отложенных столбцов. После построения мультипликаторов для всех столбцов, удовлетворяющих дополнительным условиям численной устойчивости, процесс исключения, отвечающий отложенным столбцам, осуществляется на основе частичного выбора в столбцах. При этом учитывается, что отложенный столбец a_j уже был разложен по некоторому подмножеству мультипликаторов. Вектор, отвечающей частичному разложению столбца a_j, приводится к компактной форме. При этом используется форма записи, аналогичная форме записи столбцов в матрицах Y_k. В первом восьмибайтовом слове записывается число ненулевых компонент и номер k, отвечающий мультипликатору L_k, когда было прервано разложение.

3. Тестирование

В табл. 1 приведены результаты решения задач линейного программирования с разреженными матрицами из набора NETLIB [17]. Расчеты проводились в Сибирском суперкомпьютерном центре на кластере НКЗ0Т с использованием одного процессорного элемента с тактовой частотой 3 ГГц. В таблице используются следующие обозначения: m — число строк; n — число столбцов; nz — количество ненулевых элементов в матрице A; it — число итераций; $\max V$ — максимальный размер оперативной памяти, необходимый на этапах перепостроения LU за все время решения задачи; $\max |Y_k|$ указывает максимальный размер памяти, занятый матрицами Y_k за все время решения задачи; t — время решения задачи в секундах.

Обращение к перепостроению LU при решении всех задач производилось после выполнения 100 итераций. Размеры требуемой памяти указаны с учетом памяти, занятой индексами. Небольшими усложнениями программы размеры памяти, необходимые для хранения матриц Y_k, могут быть не намного уменьшены. Рассмотрим на примере, когда столбцы a_j и a_i, участвующие в обмене, не принадлежат B_0. В матрице нужно заменить вектор $w_i = B_0^{-1} a_i$ на вектор $w_j = B_0^{-1} a_j$. Тогда, если вектор w_j содержит меньше ненулевых элементов по сравнению с w_i, w_j записывается на место w_i и не производится уплотнение записей.

Для решения задач линейного и целочисленного линейного программирования нами были разработаны программы [18] с использованием обновления FT. Контроль роста элементов в мультипликаторах позволяет получить достаточно надежное программное обеспечение и для этого случая. Вариант с обновлением SC требует меньшего числа обращений к перепостроению разложений и меньших затрат времени.

Рассмотрим некоторые экспериментальные данные по перепостроению LU-разложений. Определение номеров ведущих элементов основывается на решении задач назначения с целевой матрицей C. Матрица C обладает некоторыми особенностями. В каждом столбце матрицы C имеется по крайней мере один нулевой элемент, кроме того, матрица C особым образом отмасштабирована по сравнению с исходной матрицей A. Выбор алгоритма решения задач назначения проводился на основе тестирования некоторых прямо-двойственных алгоритмов. Алгоритмы относятся к одному и тому же классу алгоритмов кратчайших увеличивающих путей. В этом классе алгоритмов выделяется фаза предварительной обработки для определения допустимого двойственного и начально-

го (частного) решения по исходным переменным. В результате тестирования был выбран алгоритм [14]. В этом алгоритме особое внимание уделяется фазе инициализации. Пусть уже получено некоторое допустимое двойственное решение и некоторое частное по исходным переменным. Далее приводятся, как называют авторы [14], допустимые преобразования (эвристики по максимизации в двойственной задаче). Для алгоритма [14] для плотных матриц A установлена теоретическая оценка трудоемкости $O(Rn^3)$, где $R = \max_{i,j} |a_{ij}|$.

Таблица 1. Решения задач линейного программирования

Название	m	n	nz	it	$\max V$	$\max \lvert Y_k \rvert$	t
grow15	300	645	5620	672	4415	28645	0.07
stair	356	467	3856	509	11373	24865	0.04
scsd8	397	2750	8584	2714	1935	33548	0.38
pilot4	410	1000	5141	1077	10069	27766	0.16
grow22	440	946	8252	759	8800	20686	0.09
fffff800	524	854	6227	1378	5499	10042	0.41
perold	625	1376	6018	4500	24118	44005	0.96
scfxm2	660	914	5183	1073	4884	7334	0.32
pilot.we	722	2789	9126	5395	19077	53999	1.1
maros	846	1443	9614	2021	8668	36997	0.44
pilot.ja	940	1998	14698	5762	29250	64967	1.73
pilots	975	2172	13057	2261	17835	37075	0.74
scfxm3	990	1371	7777	1219	6591	10734	0.27
sctap2	1090	1880	6714	661	3067	7641	0.31
woodw	1098	8405	37474	1940	17578	31608	1.27
ships	1151	2763	8178	1444	1683	1179	0.28
shipl	1151	5427	16170	2669	1620	1303	0.51
ganges	1309	1681	6912	1236	4135	11760	0.60
sctap3	1480	2480	8874	766	4138	6889	0.51
greenba	2392	5405	30877	20879	16258	85751	17.81
maros-r7	3136	9408	144848	2873	315643	225766	5.40

В табл. 2 приведены результаты тестирования для больших разреженных матриц, доступных на сайте [19]. В численном эксперименте, кроме исходных матриц, использовались матрицы C. Матрицы C строились по исходным матрицам в точности так, как строились вспомогательные матрицы для определения трансверсали из максимизации произведения модулей ее элементов. В табл. 2 m указывает размеры квадратной матрицы; nz — количество ненулевых элементов; k показывает, какое число обращений нужно произвести к процедуре определения кратчайшего увеличивающегося пути после построения начального решения. Из данных таблицы видно, что решение задачи назначения для некоторых матриц C определяется на начальном этапе ($k = 0$).

В работе [11] рассмотрены два критерия определения номеров ведущих элементов: поиск трансверсали из условия максимизации произведения модулей ее элементов и из условия максимизации ее минимального по модулю элемента. Результаты численных экспериментов [20, 21] показывают преимущество первого критерия по отношению к обеспечению численной устойчивости.

Таблица 2. Эффективность алгоритма назначения в зависимости от масштабирования матриц

Название	Исходная матрица A					Матрица C		
	m	nz	$\max\|\alpha_{ij}\|$	k	t	$\max\|c_{ij}\|$	k	t
fidap036	3079	53851	$0.18_{10}4$	636	0.05	44.70	21	0.00
e40r5000	17281	553956	41.06	7116	4.31	45.87	1762	2.61
tsopf	18696	4396289	$0.10_{10}7$	294	1.41	42.37	1	0.03
mixtnk	29957	1995041	10.56	9885	9.37	43.21	182	0.37
invextr_new	30412	1793881	$0.30_{10}9$	6015	272.45	52.63	276	3.17
av41092	41092	1683902	3.07	9355	9.52	43.74	4803	18.56
barrier2-1	113076	3805068	$0.71_{10}12$	53198	> 7200	70.67	0	0.05
cage12	130228	2032536	0.92	37496	93.90	4.30	0	0.02
ben1	245874	6698185	$0.84_{10}4$	4032	5749.81	24.28	0	0.06
atmosmod	1489752	10319760	$0.31_{10}8$	7524	19.76	4.39	0	0.10

Работа [11] способствовала разработке пакетов программ по решению систем линейных алгебраических уравнений с разреженными матрицами, в которых по-другому решалась проблема распределения памяти по сравнению с использованием модификации критерия Марковица. Реализация стратегии Марковица требует поддержки динамических массивов потому, что на шагах гауссовских исключений изменяется заполнение строк и столбцов ненулевыми элементами. В результате применения подхода [11] созданы пакеты программ, в которых производится планирование памяти на этапе символического разложения на множители.

Статическое распределение памяти позволяет существенно упростить программирование. В качестве примера удачной реализации идеи [11] может служить пакет PARDISO [22]. Если на некоторых шагах исключения модуль ведущего элемента оказывается меньше некоторого ϵ, то значением разрешающего элемента принимается ϵ. По окончании процесса исключения в этом случае производятся итерационные уточнения. В тех случаях, когда итерационное уточнение не сходится, решение продолжается некоторым итерационным методом.

При применении методик [11] к перепостроению обратных матриц не удается добиться такого полного решения проблем распределения памяти, потому что заранее неизвестно, какие размеры памяти необходимо резервировать для отложенных столбцов.

В табл. 3 приведены примеры решения больших линейных систем с разреженными матрицами алгоритмом с использованием отложенных столбцов. В табл. 3 m обозначает размеры квадратной матрицы; nz — количество ненулевых элементов; % — процент заполнения ненулевыми элементами; k — количество отложенных столбцов; V_k — количество восьмибайтовых слов, занятых отложенными столбцами; k_{n} — количество “нормальных” столбцов; $|L| + |U|$ обозначает размеры памяти занятые под L- и U- мультипликаторами с учетом информации, занятой индексами.

Перечисление в таблице матрицы также доступны на сайте [19].

Количество отложенных столбцов зависит от числовых свойств матриц и заданного значения параметра u, который используется для контроля уровня ведущих элементов. В нашем случае полагалось $u = 0.1$.

Подразделение столбцов на столбцы-выступы и нормальные столбцы позволяет в сильно разреженных матрицах экономить память. Допустим, что в ходе разложения получена последовательность мультипликаторов $L_1, \ldots, L_k$. При построения L_{k+1} из неко-

торого столбца a_j оказалось, что последовательность $L_1, \ldots, L_k$ не изменяет исходный столбец a_j (в a_j компоненты, отвечающие ведущим строкам мультипликаторов, — нулевые). В этом случае в поле мультипликаторов L записывается признак того, что столбец нормальный и дается ссылка на столбец a_j.

Таблица 3. Примеры распределения памяти для LU-разложений с отложенными столбцами

Название	m	nz	%	k	V_k	$k_{\text{п}}$	$\lvert L\rvert + \lvert U\rvert$
vibrobox	12328	177578	0.12	–	–	1831	635260
e40r5000	17281	553956	0.19	6195	3920930	1532	65 143611
fidap035	19716	218308	0.06	69	2531	3549	1 080448
chipcooll	20082	281150	0.07	10539	14339558	1759	184524295
mark3j060	27498	170695	0.02	1916	886347	11405	15465775
av41092	41092	1683902	0.10	3103	3071448	13579	26539242
epb3	84617	463625	0.01	–	–	20334	3617287
rajat20	86916	605405	0.01	88	16847	43593	1171775
torso2	115967	1033473	0.01	–	–	21711	5156241
ben1	245874	6698185	0.01	–	–	20579	54649714

4. Заключение

В статье рассмотрен алгоритм решения задач ЛП с разреженными матрицами, в котором на итерациях не требуется явное обновление LU-разложения, что позволяет использовать простые структуры хранения данных. На этапе перепостроения LU используется схема, в которой критерий численной устойчивости имеет приоритет перед требованием разреженности. Результаты численных экспериментов подтверждают достаточно высокую эффективность алгоритмов.

Литература

1. **Муртаф Б.** Современное линейное программирование. Теория и практика.— М.: Мир, 1984.
2. **Bartels R.H., Golub G.H.** The simplex method of linear programming using the LU decomposition // Commun. ACM.— 1969.— Vol. 12.— P. 266–268.
3. **Forrest J.J.H., Tomlin J.A.** Updating triangular factors of the basis to maintain sparsity in the product form simplex method // Math. Program.— 1972.— Vol. 2.— P. 263–278.
4. **Reid J.K.** A sparsity-exploiting variant of the Bartels–Golub decomposition for linear programming bases // Math. Program.— 1982.— Vol. 24.— P. 55–69.
5. **Forrest J.J.H., Tomlin J.A.** Vector processing in simplex and interior methods for linear programming // IBM Res. Report.— 1988.— № RJ 6390 (62372).
6. **Bisschop J., Meeraus F.** Matrix augmentation and partitioning in the updating of the basis inverse // Math. Program.— 1977.— Vol. 13.— P. 241–254.
7. **Proctor P.E.** Implementation of the double-basis simplex method for the general linear programming problem // SIAM J. Algebraic Discrete Methods.— 1985.— Vol. 6, № 4.— P. 567–575.
8. **Eldersveld S.K., Saunders M.A.** A block-LU update for large-scale linear programming // SIAM J. Matrix Anal. Appl.— 1992.— Vol. 13, № 1.— P. 191–201.

9. **Markowitz H.M.** The elimination form of the inverse and its application to linear programming // Management Science. — 1957. — Vol. 3. — P. 255–269.
10. **Saunders M.A.** Large-Scale Numerical Optimization. — 2014. — http://web.stanford.edu/class/msande318/notes/notes05-updates.pdf
11. **Olschowka M., Neumaier A.** A new pivoting strategy for Gaussian elimination // Linear Algebra Appl. — 1996. — Vol. 240, № 1–3. — P. 131–151.
12. http://www.opensource.org/licenses/cpl1.0.php
13. **Писсанецки С.** Технология разреженных матриц. — М.: Мир, 1988.
14. **Jonker R., Volgenant A.** A shortest augmenting path algorithm for dense and sparse linear assignment problems // Computing. — 1987. — Vol. 38. — P. 325–340.
15. **Jonker R., Vogenant A.** Linear Assignment Problem. — http://www.assignmentproblems.com/LAPJV.htm
16. **Karypis G., Kumar V.** Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0. — University of Minnesota, Department of Computer Science/Army HPC Research Center Minneapolis, MN 55455, 1998. — http://glaros.dtc.umn.edu/gkhome/views/metis
17. **Gay D.M.** Electronic mail distribution of linear programming test problems. // Mathematical Programming Society COAL Newsletter — 1985. — Vol. 13. — P. 10–12.
18. **Zabinyako G.I., Kotelnikov E.A.** Linear optimization programs // NCC Bulletin. Series Numerical Analysis. — Novosibirsk, 2002. — Iss. 11. — P. 103–112.
19. **Davis T.A.** University of Florida sparse matrix collection. — http://www.cise.ufl.edu/ d̃avis/sparse
20. **Duff I.S., Koster J.** The design and use of algorithms for permuting large entries to the diagonal of sparse matrices // SIAM J. Matrix Anal. Appl. — 1999. — Vol. 20, № 4. — P. 889–901.
21. **Забиняко Г.И.** Перепостроение обратных матриц // Сиб. журн. индустр. матем. — 2009. — Т. 12, № 3. — С. 41–51.
22. **Schenk O., Gartner K.** Solving unsymmetric sparse systems of linear equation with PARDISO // Future Generation Computer Systems. — 2004. — Vol. 20. — P. 475–487.

Поступила в редакцию 5 марта 2015 г.,
в окончательном варианте 13 апреля 2015 г.

Литература в транслитерации

1. **Murtaf B.** Sovremennoe linejnoe programmirovanie. Teoriya i praktika. — M.: Mir, 1984.
2. **Bartels R.H., Golub G.H.** The simplex method of linear programming using the LU decomposition // Commun. ACM. — 1969. — Vol. 12. — P. 266–268.
3. **Forrest J.J.H., Tomlin J.A.** Updating triangular factors of the basis to maintain sparsity in the product form simplex method // Math. Program. — 1972. — Vol. 2. — P. 263–278.
4. **Reid J.K.** A sparsity-exploiting variant of the Bartels–Golub decomposition for linear programming bases // Math. Program. — 1982. — Vol. 24. — P. 55–69.
5. **Forrest J.J.H., Tomlin J.A.** Vector processing in simplex and interior methods for linear programming // IBM Res. Report. — 1988. — № RJ 6390 (62372).
6. **Bisschop J., Meeraus F.** Matrix augmentation and partitioning in the updating of the basis inverse // Math. Program. — 1977. — Vol. 13. — P. 241–254.
7. **Proctor P.E.** Implementation of the double-basis simplex method for the general linear programming problem // SIAM J. Algebraic Discrete Methods. — 1985. — Vol. 6, № 4. — P. 567–575.

8. **Eldersveld S.K., Saunders M.A.** A block-LU update for large-scale linear programming // SIAM J. Matrix Anal. Appl. — 1992. — Vol. 13, № 1. — P. 191–201.
9. **Markowitz H.M.** The elimination form of the inverse and its application to linear programming // Management Science. — 1957. — Vol. 3. — P. 255–269.
10. **Saunders M.A.** Large-Scale Numerical Optimization. — 2014. — http://web.stanford.edu/class/msande318/notes/notes05-updates.pdf
11. **Olschowka M., Neumaier A.** A new pivoting strategy for Gaussian elimination // Linear Algebra Appl. — 1996. — Vol. 240, № 1–3. — P. 131–151.
12. http://www.opensource.org/licenses/cpl1.0.php
13. **Pissanetski S.** Tekhnologiya razrezhennykh matrits. — M.: Mir, 1988.
14. **Jonker R., Volgenant A.** A shortest augmenting path algorithm for dense and sparse linear assignment problems // Computing. — 1987. — Vol. 38. — P. 325–340.
15. **Jonker R., Vogenant A.** Linear Assignment Problem. — http://www.assignmentproblems.com/LAPJV.htm
16. **Karypis G., Kumar V.** Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0. — University of Minnesota, Department of Computer Science/Army HPC Research Center Minneapolis, MN 55455, 1998. — http://glaros.dtc.umn.edu/gkhome/views/metis
17. **Gay D.M.** Electronic mail distribution of linear programming test problems. // Mathematical Programming Society COAL Newsletter — 1985. — Vol. 13. — P. 10–12.
18. **Zabinyako G.I., Kotelnikov E.A.** Linear optimization programs // NCC Bulletin. Series Numerical Analysis. — Novosibirsk, 2002. — Iss. 11. — P. 103–112.
19. **Davis T.A.** University of Florida sparse matrix collection. — http://www.cise.ufl.edu/ d̃avis/sparse
20. **Duff I.S., Koster J.** The design and use of algorithms for permuting large entries to the diagonal of sparse matrices // SIAM J. Matrix Anal. Appl. — 1999. — Vol. 20, № 4. — P. 889–901.
21. **Zabinyako G.I.** Perepostroenie obratnykh matrits // Sib. zhurn. industr. matem. — 2009. — T. 12, № 3. — S. 41–51.
22. **Schenk O., Gartner K.** Solving unsymmetric sparse systems of linear equation with PARDISO // Future Generation Computer Systems. — 2004. — Vol. 20. — P. 475–487.