

# ОЦЕНКА УПОРЯДОЧЕНИЙ ДЛЯ ПЕРЕПОСТРОЕНИЯ БАЗИСОВ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ.

Г. И. Забиняко

*Институт вычислительной математики и математической геофизики СО РАН, 630090, Новосибирск*

УДК 519.852.61

Рассматриваются версии упорядочений для перепостроения базисов, реализованные в пакете программ линейного программирования. На основе решения тестовых задач линейного программирования и решения систем линейных алгебраических уравнений проводится сравнительный анализ алгоритмов упорядочения. Рассматриваются упорядочения ориентированные на обеспечение численной устойчивости и разреженности результирующих  $LU$ -разложений базисов.

**Ключевые слова:** линейное программирование, модифицированный симплекс-метод, перепостроение базисных матриц,  $LU$ -мультипликативное представление базисных матриц, упорядочения базисных матриц.

## Введение

Рассматривается применение модифицированного симплекс-метода [1] для решения задач линейного программирования (ЛП):  $\min(c, x)$  при условиях  $Ax = b, \alpha \leq x \leq \beta$ , где  $A$  — матрица размера  $m \times n, n \geq m$ , а векторы  $x, \alpha, \beta \in R^n, b \in R^m$ . Предполагается, что матрица  $A$  — разреженная. На итерациях модифицированного симплекс-метода при замещении базисного столбца небазисным осуществляется рекуррентный пересчет  $LU$  представления текущей базисной матрицы, либо рекуррентный пересчет матрицы, обратной к текущей базисной  $U^{-1}L^{-1}$ . В результате выполнения итерации накапливаются ошибки округления, увеличивается объем памяти, занятой мультипликативным представлением базисных матриц. Поэтому наступает момент когда нужно заново построить  $LU$  либо  $U^{-1}L^{-1}$  — разложения согласно списка базисных переменных на данный момент. На этапе перепостроения базиса к мультипликативному представлению  $LU$  можно предъявить некоторые требования по устойчивости и разреженности.

Пакет программ ЛП-ВЦ [2] кроме процедур численного решения задач содержит программное обеспечение по логическому контролю и преобразованию входного MPS-формата [1] во внутренний — столбцовый разреженный формат. Таким образом пользователям пакета представляется некоторый технологический цикл решения задач ЛП.

В пакете ЛП-ВЦ обновление на итерациях  $LU$  мультипликативного представления осуществляется по алгоритму Форреста-Томлина [3].

Первоначально на этапе перепостроения базисов у нас использовалась процедура  $P^3$  (Preassigned Pivot Procedure). Процедура  $P^3$ , предложенная в [4], представляет собой эвристический метод приведения с помощью перестановок строк и столбцов базисную матрицу  $B$  к виду, по возможности наиболее близкому нижней треугольной матрице. При этом делается попытка минимизировать как число столбцов, имеющих ненулевые элементы над главной диагональю, так и количество нулей в этих столбцах над главной диагональю. Столбцы, которые не имеют ненулевых элементов над главной диагональю, называются нормальными, а которые имеют — столбцами-выступами.

После упорядочения строк, столбцов ведущих элементов выбираются на главной диагонали в порядке сверху вниз. При этом мультипликаторы нормальных столбцов полностью определяются с помощью ссылки на номер столбца и ведущей строки исходной матрицы  $A$ . Мультипликаторы столбцов-выступов получаются после их умножения справа на матрицу  $\bar{L} = L_k^{-1}, \dots, L_1^{-1}$ , где  $L_i^{-1}$  — мультипликаторы, построенные на предыдущих шагах ( $i = 1, \dots, k$ ).

Обеспечить численную устойчивость разложения в процедуре  $P^3$  достаточно трудно. После получения списка ведущих элементов перед формированием мультипликаторов можно воспользоваться стандартным

контролем ведущих элементов на устойчивость и отложить использование нормального столбца или столбца-выступа, если ведущий элемент  $\alpha_i$  не удовлетворяет неравенству  $|\alpha_i| \geq u\alpha_{\max}$ ,  $0 < u < 1$ ,  $\alpha_{\max} = \max_{k \in K} |\alpha_k|$ ,  $K$  — множество индексов, которые не использовались для назначения ведущих элементов. В конце процесса разложения отложенные столбцы следует обработать, используя частичный выбор в столбце. В результате возрастает заполненность мультипликативного представления, и растут временные затраты на решение задач.

Далее в пакете ЛП-ВЦ на этапе перестроения базисов был реализован подход принятый, в настоящее время, для решения СЛАУ с разреженными матрицами. Предварительное назначение ведущих элементов получается решением задачи назначения [5], которое обеспечивает определение максимального произведения модулей элементов  $\max_{i,j \in T} |b_{ij}|$ , где индексы  $i, j$  принадлежат искомой трансверсали  $T$ , а  $b_{i,j}$  — элементы текущей базисной матрицы.

Обновление мультипликативного  $LU$  представления на итерациях симплекс-метода по методу Форреста-Томлина в вопросе численной устойчивости уступает методу Бартелса-Голуба [6]. Метод Бартелса-Голуба в программной реализации требует использования сложных динамических структур данных.

Повысить численную устойчивость расчетов на итерациях симплекс-метода можно с помощью алгоритма симплекс-метода с использованием двойного базиса. В этом алгоритме на итерациях не требуется в явном виде обновлений  $LU$ -разложений. Решения, полученные с фиксированными факторами  $LU$  (полученными на этапе перестроения базисов), корректируются с помощью небольших вспомогательных матриц. Этот алгоритм впервые был предложен [7] и там же рассмотрена 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 (здесь  $B_0^{-1}$  получена на этапе перестроения, а  $B_k^{-1}$  — в результате  $k$  обновлений на итерациях симплекс-метода). В [8] рассмотрена реализация в которой за счет использования дополнительных массивов оперативной памяти не требуется два дополнительных обращения к  $B_0^{-1}$ . Нами также был реализован [9] экспериментальный вариант программы алгоритма с использованием двойного базиса. Алгоритм симплекс-метода с использованием двойного базиса в программной реализации не требует сложных структур данных,  $LU$ -факторы представляются в стандартном столбцовом разреженном формате. Алгоритм обладает хорошим быстродействием. Единственным ограничивающим фактором для применения в практике являются повышенные требования к объемам оперативной памяти.

Дальше рассмотрим алгоритмы упорядочения, отвечающие за численную устойчивость и разреженность  $LU$ -разложений на этапе перестроения базисов, реализованные в пакете программ ЛП-ВЦ. На основе численных экспериментов проанализируем эффективность алгоритмов. Все численные эксперименты, рассмотренные в этой заметке, выполнялись в Сибирском суперкомпьютерном центре на сервере SL390sG7 с тактовой частотой процессорных элементов 2.93 ГГц. Все вычисления проведены в однопроцессорном режиме.

## 1 Предварительное определение ведущих элементов

Поиск номеров ведущих элементов осуществляется на основе построения трансверсали из условия максимизации произведения модулей ее элементов [8]. 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  $\bar{A} = P^T A$  имеет главную диагональ, составленную из элементов  $a_{ij}$  для  $i, j \in T$ .

Задача построения трансверсали  $T$ , отвечающей максимуму  $\prod_{i,j \in T} |a_{ij}|$ , 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:

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

Для выбора алгоритма решения задач назначения мы использовали обзор [10]. Для тестирования выбрали прямо-двойственные алгоритмы назначения, теоретическая оценка трудоемкости у которых для плотных матриц одинаковая и составляет  $O(n^3)$ . В этом классе алгоритмов выделяется фраза предварительной обработки для определения допустимого двойственного решения и начального (частного) решения по исходным переменным.

Если в приведенной выше задаче булева программирования ослабить требования  $x_{ij} = 0$  или 1 для  $i, j = 1, \dots, n$ , заменив их на  $x_{ij} \geq 0$ , то придем к следующей двойственной задаче:

$$\begin{aligned} & \text{максимизировать} && \sum_{i=1}^n u_i + \sum_{j=1}^n v_j, \\ & \text{при ограничениях} && c_{ij} - u_i - v_j \geq 0 \end{aligned}$$

для  $i, j = 1, \dots, n$ . Таким образом, двойственная задача состоит в максимизации сумм двойственных переменных при условиях неотрицательности приведенных затрат. Приведенные затраты на отдельных этапах алгоритма выступают в качестве новых цен, т.е.  $c_{ij}$  заменяются на  $\bar{c}_{ij} = c_{ij} - u_i - v_j$ .

Первым прямо-двойственным алгоритмом с полиномиальной оценкой трудоемкости является венгерский алгоритм [11]. В венгерском алгоритме вначале определяются оценки  $u_i = \min_{j=1, \dots, n} c_{ij}$  для  $i = 1, \dots, m$ ; затем  $v_j = \min_{i=1, \dots, n} (c_{ij} - u_i)$  для  $j = 1, \dots, n$ . Начальное решение по исходным переменным:  $x_{ij} := 1$ , если  $c_{ij} - u_i - v_j = 0$ . Пусть таким образом удалось задать  $k$  переменных с  $x_{ij} = 1$ , если  $k = n$ , то задача решена. В случаях, когда  $k < n$  для решения задачи необходимо  $L = n - k$  раз обратится к процедуре определения кратчайшего увеличивающегося пути [10].

В прямо-двойственных алгоритмах [11] вводится понятие допустимых преобразований матрицы  $C$ . Преобразование  $S$  матрицы  $C = \{c_{ij}\}$  в матрицу  $\tilde{C} = \{\bar{c}_{ij}\}$  называется допустимой, если для любых подстановок  $\phi(i)$  из  $\{1, 2, \dots, n\}$  имеет место равенство  $\sum_{i=1}^n c_{i\phi(i)} = \sum_{i=1}^n \bar{c}_{i\phi(i)} + z(s)$ , где  $z(s)$  — постоянная зависящая только от преобразования  $S$ .

Для использования в ЛП-ВЦ был выбран алгоритм [12]. В этом алгоритме на начальном этапе делается попытка с помощью допустимых преобразований максимизировать количество назначений. Пусть уже получено некоторое допустимое двойственное решение и некоторое частное решение по исходным переменным. Для каждой назначенной строки  $i$  такой, что для этой строки предписан столбец  $j_1$  приводятся следующие преобразования: определяется  $\mu = \min_{j \neq j_1} c_{ij} - v_j$  и переопределяются двойственные переменные  $v_{ij}$  и  $u_i$  следующим образом:  $v_{j_1} = v_{j_1} - (\mu - u_i)$ ,  $u_i = \mu_i$ . Аналогичные преобразования проводятся для неназначенных строк, в результате которых переопределяются двойственные переменные и, возможно, пополняется количество назначенных строк. На сайте [13] помещена для свободного доступа процедура на Фортране, в которой реализован алгоритм [12] для целочисленных матриц. Эта процедура была нами адаптирована для вещественных матриц.

Для этого алгоритма масштабирование, которое получается в результате построения матрицы  $C$  из базисной матрицы  $B$ , позволяет назначить большинство  $x_i$  на предварительном этапе. В таблице на примере тестовых матриц из [14] показывается различие в количестве назначенных  $x_i$  на начальном этапе для исходных матриц  $A$  и полученных из них матриц  $C$ .

В табл. 1 иллюстрируется применение прямо-двойственного метода решения задач назначения.

Таблица 1: Зависимость алгоритма назначения от масштабирования матриц.

Исходная матрица A						Матрица C		
название	$n$	$nz$	$\max  a_{ij} $	$L$	$t_a$	$\max  c_{ij} $	$L$	$t_c$
e40r5000	17281	553956	$0.41 \cdot 10^2$	6819	14.29	46.1	1338	5.48
fidap035	19716	218308	$0.10 \cdot 10^9$	1211	13.38	36.9	50	0.83
fidapm11	22294	623554	$0.53 \cdot 10^0$	3787	45.93	10.4	1185	6.98
rim	22560	1014951	$0.27 \cdot 10^5$	3582	75.35	55.5	56	0.82
sme3Db	29067	2081063	$0.32 \cdot 10^5$	1262	36.73	51.2	0	0.02
mixtank_new	29957	1995041	$0.10 \cdot 10^2$	7925	35.83	10.6	126	1.79
invextr1	30412	1793881	$0.30 \cdot 10^9$	4489	462.21	60.6	136	10.93
twotone	120750	1.224224	$0.64 \cdot 10^4$	3448	43.58	51.8	566	0.53
FEM_3D_therm	147900	3489300	$0.29 \cdot 10^0$	13315	449.84	9.0	0	0.04
freescal1	3428755	18920347	$0.62 \cdot 10^2$	428456	32651.18	35.6	0	0.53
circuit5m	5558328	59524291	$0.10 \cdot 10^4$	5161001	8470.88	42.0	13	5.23

В последней строке табл. 1 приведены результаты для матрицы с  $n = 5558328$ , где решение задачи назначения с коэффициентами исходной матрицы  $A$  требует выполнить  $L = 5161001$  обращений к процедуре определения кратчайшего увеличивающего пути, а для построенной из  $A$  матрицы  $C$  достаточно было  $L = 13$ . В некоторых примерах получено  $L = 0$ , т.е. решение задачи получено на начальном этапе. При решении задачи ЛП средней размерности  $n \times m = 3000 \times 5000$  максимальное значение для величины  $L$  колеблется в интервале 30–60 притом, что в некоторых задачах ЛП требуется выполнить сотни обращений к задачам назначения.

## 2 Симметрические перестановки обеспечения разреженности

Симметрические перестановки предназначены для определения очередности использования предварительно назначенных ведущих элементов с целью уменьшения заполненности  $LU$ -мультипликативного представления базисных матриц на этапе перестроения. В пакете ЛП-ВЦ для этих целей использовалась процедура из библиотеки пакета программ METIS [15].

Пакет программ METIS ориентирован на использование в UNIX-системах. Для применения в WINDOWS нужно провести дополнительную работу по адаптации среды. Кроме того, при эксплуатации пакета программ ЛП-ВЦ на компьютерах PC использование METIS требует дополнительно достаточно больших объемов оперативной памяти. В связи с этим была осуществлена попытка заменить пакет METIS на компактную процедуру минимальной степени [16]. Примеры применения алгоритмов в решении тестовых задач ЛП приведены в табл. 2.

Таблица 2: Сопоставление упорядочений в ЛП с помощью METIS и алгоритма минимальной степени.

N пп	$m$	$n$	%	metis		mmd	
				$K$	$V_0$	$K$	$V_0$
1	626	1376	0.70	54	21920	54	19901
2	820	1571	0.81	96	29574	90	29932
3	940	1988	0.80	51	29255	50	26120
4	1000	8806	0.32	120	14312	127	13858
5	1090	1880	0.33	6	5877	7	6077
6	1309	1681	0.31	11	7104	12	7257
7	1480	2480	0.24	9	7885	9	8057
8	2324	3489	0.17	110	22865	112	19881
9	3068	3678	0.12	53	25462	48	24488
10	3516	4067	0.10	71	26065	93	21071

В этой таблице:

- $K$  — указывает количество обращений к перепостроению базисов за все время решения ЛП;
- $V_0$  — максимальный объем данных занятый  $LU$ -мультипликативным представлением по всем перепостроениям базисов, выполненным для данной задачи.

Результаты табл. 2 показывают, что оба алгоритма упорядочения при решении задач ЛП небольшой и средней размерности обеспечивают примерно одинаковые объемы данных, занятые  $LU$ -мультипликативным представлением на этапе перепостроения базисов.

Можно предположить, что для разных видов структур заполненности алгоритмы будут давать разные результаты. В пакете METIS на основе эвристик определяется минимальный разделитель [17]. В алгоритме минимальной степени на основе исходного графа, используя понятие достижимых множеств, на очередном шаге выбирается вершина минимальной степени. Применение неразличных вершин позволяет экономить на количестве обращений к вычислению степеней вершин графа [16].

В табл. 3 на примерах решения СЛАУ иллюстрируются различия в применении алгоритмов из пакета METIS и минимальной степени.

Таблица 3: Сопоставление упорядочений в СЛАУ с помощью METIS и алгоритма минимальной степени.

$N$ пп	название	$m$	$nz$	%nz	Алгоритм минимальной степени			Подпрограмма METIS		
					$t_{sec}$	$K$	$ V $	$t_{sec}$	$K$	$ V $
1	psmigr_1	3140	543162	5.51	1.97	1	5385105	0.08	3	5665747
2	heart1	3557	1387773	10.97	0.01	4	3077775	0.02	14	3978359
3	vibrobox	12328	177578	0.12	0.05	0	650413	0.10	0	635260
4	shermAC	18510	145149	0.04	0.25	87	360122	0.07	57	328870
5	nmos3	18588	386594	0.11	0.01	4803	27195920	0.07	2674	9224594
6	tsopf	18696	4396289	1.26	8.55	1	9669146	1.32	0	13905773
7	fidap035	19716	218308	0.06	0.02	43	1084069	0.11	50	1072812
8	chipool	20082	281150	0.04	0.04	0	9233863	0.17	0	6616613
9	ns3Da	20414	1679599	0.40	0.09	16	10610358	0.23	11	16220422
10	rim	22560	1014951	0.20	0.10	4108	22978004	0.17	4181	21989901
11	af23560	23560	484256	0.09	0.09	10	10160550	0.28	11	8708760
12	zahao2	33861	166453	0.01	0.11	0	3791316	0.13	0	2983091
13	rajat15	37261	443573	0.03	0.18	104	1738815	0.31	117	1730298
14	epb3	84617	463625	0.03	0.02	0	4342086	0.30	0	3617287
15	rajat20	86916	605045	0.06	0.96	42	1004189	0.93	44	1141720
16	to2so2b	115976	1033473	0.18	0.03	0	4589278	0.47	0	5156241
17	ben1	245874	6698185	2.06	0.14	0	34810105	0.31	0	34292142

Обозначения табл. 3:

- $K$  — количество отложенных столбцов для их разложения с использованием частичного выбора в столбце;
- $V$  — объем памяти занятый мультипликаторами.

Следует пояснить возникновение показателя  $K$ . Использование методологии из [8] позволяет строить современные пакеты программ решения СЛАУ без использования динамических массивов. Для этого после упорядочений на устойчивость и разреженность проводится символическое разложение которое дает точные значения необходимого объема оперативной памяти. При перестроении  $LU$ -разложении базисов такой универсальности достичь не удастся [18]. Если очередной предписанный упорядочениями столбец базисной матрицы не удовлетворяет стандартным условиям на устойчивость, то полученное разложение запоминается, а в конце процесса все отложенные столбцы обрабатываются с использованием частичного выбора в столбце. Отсюда следует, что при больших значениях величины  $K$ , трудно сравнить эффективность алгоритмов упорядочения.

Сравнение результатов из табл. 3 в строке 6 (матрица fsopf) и строке 9 (матрица ns3Da) наглядно показывают что для разных типов заполненности эффективность этих алгоритмов различна.

## Заключение

В этой заметке анализируется эффективность и удобство программной реализации упорядочений, направленных на обеспечение численной устойчивости и разреженности на этапе перестроения  $LU$ -разложений базисов ЛП.

## Список литературы

- [1] Муртаф Б. Современное линейное программирование. Теория и практика. М.: Мир, 1984.
- [2] Забиняко Г.И., Котельников Е.А., Кобкова Т.М., Рожин В.Е. Программы линейного программирования ЛП-ВЦ. — Новосибирск, 1995. — (Отчет / ВЦ СО РАН; № ГР.01.9.30001317; Инв. № 02.9.50003757).
- [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, **2**, p. 263–278.
- [4] Hellerman E., Rarick D. C. Reinversion with the preassigned pivot procedure // Math. Prog. 1971. N 1. P. 195–216.
- [5] Olschowka M., Neumaier A. A new pivoting strategy for Gaussian elimination // Linear Algebra Appl. 1996. V. 240. No 1–3. P. 131–151.
- [6] Bartels R.H., Golub G.H.. The simplex method of linear programming using the LU decomposition // Commun. ACM, 1969, 12, p. 266–268.
- [7] Bisschop J., Meeraus F. Matrix augmentation and partitioning in the updating of the basis inverse // Math. Program., 1977, **13**, p. 241–254.
- [8] Eldersveld S.K., Saunders M.A. A block-LU update for large-scale linear programming // SIAM J. Matrix Anal. Appl., 1992, V. 13, № 1, p. 191–201.
- [9] Забиняко Г. И. Алгоритм симплекс-метода с использованием двойного базиса // СибЖВМ. 2015. Т. 18, № 4. С. 349–359.
- [10] Dell’Amico M., Toth P. Algorithms and codes for dense assignment problems: the state of the art // Discrete Appl. Math. 2000. V. 100. P. 17–48.
- [11] Burkard R.E., Qela E. Linear Assignment Problems and Extensions // Handbook of combinatorial optimization D.-Z. Du and P.M. Pardalos (Eds.), 1999, pp. 75–149
- [12] Jonker R., Volgenant A. A shortest augmenting path algorithm for dense and sparse linear assignment problems // Computing. 1987. V. 38. P. 325–340.
- [13] Jonker R., Volgenant A. Linear Assignment Problem.  
URL: <http://www.assignmentproblems.com/LAPJV.htm>.
- [14] Davis T.A. University of Florida sparse matrix collection.  
URL: <http://www.cise.ufl.edu/~davis/sparse>.
- [15] 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).  
URL: <http://www.cs.umn.edu/~karypis>.
- [16] Liu J.W.H. Modification of the Minimum-Degree Algorithm by Multiple Elimination // ACM Transactions on Mathematical Software, Vol. 11, No. 2, June 1985.
- [17] А.Джордж, Дж.Лю Численное решение больших разреженных систем уравнений. М.: Мир, 1984.
- [18] Забиняко Г.И. Перестроение обратных матриц // Сиб.журн.индустр.матем. 2009. Т. 12, № 3. С. 41–51.

*Герард Идельфонович Забиняко — к.т.н., вед. науч.сотр. Института  
вычислительной математики и математической геофизики СО РАН;  
e-mail: zabin@rav.sscs.ru.*

*Дата поступления — 24 мая 2017 г.*