Статистически эффективная схема распараллеливания алгоритмов дуализации*

***Е. В. Дюкова*[1], *А. Г. Никифоров*[2], *П. А. Прокофьев*[1]**
edjukova@mail.ru, ankifor@gmail.com, p_prok@mail.ru
[1]ВЦ РАН, [2]ВМиК МГУ

Одной из центральных задач дискретной математики является дуализация, которая может быть сформулирована как построение всех неприводимых покрытий булевой матрицы. Это задача является вычислительно сложной. В данной работе предлагается эффективная в типичном случае схема распараллеливания асимптотически оптимальных алгоритмов дуализации. Предлагаемая схема основана на статистическом анализе множества неприводимых покрытий булевой матрицы.

Ключевые слова: перечислительная задача; дуализация; неприводимое покрытие булевой матрицы; трансверсаль гиперграфа; эффективность в среднем; асимптотически оптимальный алгоритм; параллельные вычисления; балансировка нагрузки; сильная масштабируемость.

Statistically Efficient Parallel Scheme for Dualization Algorithms*

***E. Djukova*[1], *A. Nikiforov*[2], *P. Prokofyev*[1]**
Moscow, [1]Dorodnicyn Computing Centre of Russian Academy of Sciences, [2]Moscow State University

Background: Dualization is a fundamental problem in discrete mathematics. It is equivalent to irredicible coverings enumeration of a given boolean matrix. This problem is of high computational complexity. Therefore, using of parallel computing is practical. However, existing approaches to dualization algorithms parallelizing are trivial and not much efficient.
Methods: An efficient on average parallel scheme ($\mathfrak{B}$-scheme) for asymptotically optimal dualization algorithms is suggested. The proposed scheme is based on statistical analysis of irreducible coverings set.
Results: The experiments show that $\mathfrak{B}$-scheme provides a balanced load of processors on the average and that gained speedup of parallel dualization algorithm is almost the highest possible.
Concluding Remarks: $\mathfrak{B}$-scheme outperforms other considered approaches to dualization algorithms parallelization. However, the number of processors used by $\mathfrak{B}$-scheme should be rather small; otherwise, no performance improvement is gained. Getting rid of processors number limitation and expanding the range of aproach applicability will be a topic of future works.

Keywords: enumeration; dualization; irreducible covering of a boolean matrix; hypergraph transversal; on the average efficiency; asymptotically optimal algorithm; parallel computations; load balancing; strong scaling.

Введение

Пусть L — булева матрица размера m на n. Набор столбцов H матрицы L называется покрытием, если в подматрице L^H матрицы L, образованной столбцами из H, нет строки,

Работа частично поддержана грантами РФФИ №13-01-00787-а и №14-07-00819-а и грантом президента РФ НШ-4908.2014.1.

состоящей из одних нулевых элементов. Покрытие называется неприводимым, если любое его собственное подмножество не является покрытием. Обозначим через $P(L)$ множество всех неприводимых покрытий матрицы L.

Задача построения $P(L)$ называется дуализацией и является одной из центральных перечислительных задач дискретной математики. Дуализация возникает, в частности, в задачах логического анализа данных, например при разработке логических распознающих алгоритмов (тестовых алгоритмов, алгоритмов голосования по представительным наборам, по антипредставительным наборам, по покрытиям класса).

Существуют другие формулировки рассматриваемой задачи, в частности использующие понятия теории графов и теории булевых функций. Приведем эти формулировки.

1. Пусть дан гиперграф G. Требуется построить множество всех минимальных трансверсалей этого гиперграфа.
2. Пусть монотонная булева функция f задана конъюнктивной нормальной формой, не содержащей отрицания переменных. Требуется построить сокращенную дизъюнктивную нормальную форму функции f.

Эффективность алгоритмов перечисления принято оценивать сложностью шага, т. е. сложностью построения одного решения [1]. Говорят, что алгоритм работает с полиномиальной задержкой, если каждый его шаг выполняется за полиномиальное время. Алгоритм с полиномиальной задержкой для задачи дуализации до сих пор не построен. В [2] предложен подход к построению асимптотически оптимальных алгоритмов дуализации, который получил развитие в работах [3, 4]. Асимптотически оптимальные алгоритмы делают полиномиальные шаги, но некоторые из этих шагов являются «лишними». Шаг считается «лишним», если построен набор столбцов, не являющийся неприводимым покрытием, или построен набор столбцов, найденный ранее. При этом доля «лишних» шагов стремится к нулю для почти всех матриц данного размера при увеличении размера матрицы.

Время работы алгоритмов дуализации очень быстро растет с ростом размера входных данных, поэтому актуальным является использование параллельных вычислений. Существуют простые и очевидные схемы распараллеливания алгоритмов, решающих указанную задачу. В данной работе показано, что главный недостаток таких схем — неравномерная загрузка процессоров, что является причиной недостаточного ускорения времени работы параллельного алгоритма по сравнению с его однопоточной версией. Поэтому важной является задача создания более эффективных схем распараллеливания.

Ниже предлагается эффективная схема распараллеливания асимптотически оптимальных алгоритмов ($\mathfrak{B}$-схема), которая основана на статистическом анализе множества неприводимых покрытий булевой матрицы.

Положим $J_n = \{1, \ldots, n\}$. Через $P_j(L)$, $j \in J_n$, обозначим множество неприводимых покрытий булевой матрицы L, для которых j является наименьшим номером столбца. Очевидно, что $\{P_j(L) | \, j \in J_n\}$ является разбиением множества $P(L)$ на непересекающиеся подмножества. Предлагаемый подход к построению параллельного алгоритма дуализации базируется на использовании статистических свойств множеств $P_j(L)$, $j \in J_n$, для случайных матриц.

В $\mathfrak{B}$-схеме каждое из множеств $P_1(L), P_2(L), \ldots, P_n(L)$ строится одним из p процессоров, при этом каждый процессор может построить несколько множеств. Установлено, что схема является эффективной при сравнительно небольшом числе процессоров относительно числа столбцов матрицы L.

Работа этой схемы проиллюстрирована на примере асимптотически оптимального алгоритма АО2 [3], который на каждом шаге строит неприводимое покрытие. Однако каждое покрытие из $P(L)$ может быть построено неоднократно. Проверка, было ли построено неприводимое покрытие на предыдущих шагах, осуществляется за полиномиальное время. В [3, 4, 5] показана практическая применимость асимптотически оптимальных алгоритмов: при определенных условиях эти алгоритмы работают быстрее других существующих алгоритмов дуализации.

Проводится сравнение $\mathfrak{B}$-схемы с ранее известной $\mathfrak{U}$-схемой, в которой каждый из p процессоров обрабатывает одинаковое число столбцов, равное примерно n/p. Эксперименты показывают, что $\mathfrak{B}$-схема осуществляет более равномерную загрузку процессоров по сравнению с $\mathfrak{U}$-схемой и обеспечивает лучшее ускорение. Это ускорение близко к максимально возможному.

Алгоритм дуализации АО2

Обозначим через M_{mn} множество булевых матриц размера m на n. Пусть $L = (a_{ij}) \in M_{mn}$. Очевидно, что покрытие H длины r является неприводимым тогда и только тогда, когда в L^H найдется перестановочная подматрица порядка r (матрица $r \times r$, в каждой строке и в каждом столбце которой в точности один единичный элемент). Для перестановочной подматрицы Q с единичными элементами $a_{i_1j_1}, \ldots, a_{i_rj_r}$ введем следующее обозначение: $Q = \{(i_k, j_k) \mid k \in J_r\}$. Перестановочную матрицу Q назовем максимальной, если она не содержится ни в какой другой перестановочной подматрице матрицы L. Максимальная перестановочная подматрица $Q = \{(i_1, j_1), \ldots, (i_r, j_r)\}$ называется правильной, если набор столбцов $H(Q)$ с номерами $\{j_1, \ldots, j_r\}$ принадлежит $P(L)$.

Основной особенностью алгоритма АО2 является то, что на каждом шаге этого алгоритма строится правильная подматрица Q и осуществляется проверка, не было ли покрытие $H(Q)$ построено на предыдущих шагах.

Будем говорить, что $j \in J_n$ покрывает $i \in J_m$ ($i \in J_m$ покрывает $j \in J_n$), если $a_{ij} = 1$. Обозначим $R \subseteq J_m$ и $C \subseteq J_n$. Паре (R, C) поставим в соответствие подматрицу $L(R, C)$ матрицы L, образованную строками с номерами из R и столбцами с номерами из C. Будем говорить, что $i_2 \in J_m$ охватывает $i_1 \in J_m$ в C, если $a_{i_1j} \leqslant a_{i_2j}, \forall j \in C$ (обозначение $i_1 \preccurlyeq_C i_2$).

Алгоритм осуществляет обход дерева решений в глубину. Узлами этого дерева являются тройки (Q, R, C), где Q — перестановочная подматрица (для висячих вершин подматрица Q является правильной). Корню дерева соответствуют $Q = \varnothing, R = J_m, C = J_n$. На каждом шаге алгоритма строится ветвь дерева решений. Зафиксируем некоторую ветвь длины r. Пусть узлу t-го уровня, $t \in J_r$, соответствуют (Q_t, R_t, C_t). Назовем правильную подматрицу $Q_r = \{(i_1, j_1), \ldots, (i_r, j_r)\}$ верхней, если $\exists t \in J_r$ и $\exists i \in R_t, i > i_r$ такие, что подматрица $Q = Q_r \setminus \{(i_t, j_t)\} \cup \{(i, j_t)\}$ является правильной. Определим предикат $\mathrm{upper}(Q)$ равный единице тогда и только тогда, когда подматрица Q является верхней. Очевидно, если подматрица Q является верхней, то неприводимое покрытие $H(Q)$ не было построено на предыдущих шагах алгоритма.

Ниже приведено описание алгоритма АО2, которое для удобства разбито на четыре процедуры.

Процедура 1 запускает алгоритм АО2 путем вызова процедуры 2 с параметрами, соответствующими корню дерева решений.

Процедура 2 является рекурсивной процедурой. Она строит поддерево решений с корнем в $(Q \cup \{(i_0, j_0)\}, R', C'), R' \subseteq R, C' \subseteq C$, где R' и C' формируются после вызова проце-

Процедура 1 AO2

Вход: L;
Выход: $\varnothing$;
1: BuildSubtree$(\varnothing, J_m, J_n, L, 0, 0)$

Процедура 2 BuildSubtree

Вход: Q, R, C, L, i_0, j_0;
Выход: $\varnothing$;
1: $(R, C) \leftarrow$ ReduceSubmatrix(R, C, L, i_0, j_0)
2: $C \leftarrow$ ProcessUnityColumns(Q, R, C, L)
3: **для всех** $j \in C$
4: **для всех** $i \in \{u \in R \,|\, a_{ij} = 1\}$
5: BuildSubtree$(Q \cup \{(i, j)\}, R, C, L, i, j)$

Процедура 3 ReduceSubmatrix

Вход: R, C, L, i_0, j_0;
Выход: R, C;
1: **если** $i_0 > 0$ и $j_0 > 0$ **то**
2: $C \leftarrow \{j \in C \,|\, j > j_0\}$
3: $C \leftarrow \{j \in C \,|\, a_{i_0 j} = 0\}$
4: $R \leftarrow \{i \in R \,|\, a_{ij_0} = 0\}$
5: $R \leftarrow \{i \in R \,|\, \nexists u \in R, u > i : a_{ij} = a_{uj}, \forall j \in C\}$
6: $R \leftarrow \{i \in R \,|\, \nexists u \in R, u \neq i : u \preccurlyeq_C i\}$
7: $C \leftarrow \{j \in C \,|\, \exists i \in R : a_{ij} = 1\}$

Процедура 4 ProcessUnityColumns

Вход: Q, R, C, L;
Выход: C;
1: **для всех** $j \in \{v \in C \,|\, a_{ij} = 1, \forall i \in R\}$
2: **если** upper$(Q \cup \{(i, j)\}) = 1$ **то**
3: **печать** $H(Q \cup \{(i, j)\})$

дур 3 и 4. На первой итерации алгоритма $i_0 = 0$ и $j_0 = 0$, что соответствует корню дерева решений.

Процедура 3 вызывается из процедуры 2 и формирует упомянутые выше R' и C' путем

1) удаления из C покрытых i_0 элементов;
2) удаления из R покрытых j_0 элементов;
3) удаления из $L(R, C)$ повторов строк;
4) удаления из R охватывающих в C элементов;
5) удаления из $L(R, C)$ столбцов, состоящих из одних нулей.

Процедура 4 вызывается из процедуры 2 и строит висячие вершины, которые являются дочерними для узла (Q, R, C). В частности, процедура находит в подматрице $L(R, C)$ столбцы, состоящие из одних единиц, и сопоставляет каждому из них правильную под-

матрицу. Если эта подматрица является верхней, то покрытие, ей соответствующее, распечатывается.

Статистические свойства множеств $P_j(L)$

Зафиксируем размер $m \times n$ булевых матриц. В качестве пространства элементарных событий Ω рассмотрим множество всевозможных пар (L, H), где $L \in M_{mn}$, $H \in P(L)$.

Введем на Ω случайную величину $\eta(L, H)$, равную наименьшему номеру столбца в H. Обозначим $f(j) = \mathsf{P}\{\eta(L, H) = j\}$. Требуется найти функцию $f(j)$.

Случайная величина ξ имеет бета-биномиальное распределение $\mathfrak{B}(\alpha, \beta, r)$, если она принимает значения q из множества $\{0, 1, \ldots, r\}$ с вероятностью

$$\varphi(q; \alpha, \beta, r) = \binom{r}{q} \frac{B(q + \alpha, r - q + \beta)}{B(\alpha, \beta)},$$

где B — бета-функция, а $\alpha > 0, \beta > 0$ и $r \in \mathbb{N}$ — параметры распределения. Обозначим

$$\psi_{\alpha\beta}(j) = \varphi(j - 1; \alpha, \beta, n - 1), \quad j \in J_n.$$

Проведем статистический эксперимент. Пусть $\boldsymbol{x} = (x_1, \ldots, x_N)$ — выборка из распределения $f(j)$. Проверим статистическую гипотезу $H_0 = \{f(j) = \psi_{\alpha\beta}(j)\}$ о виде распределения. В случае, когда гипотеза верна, случайная величина $\eta - 1$ имеет распределение $\mathfrak{B}(\alpha, \beta, n - 1)$.

Для проверки гипотезы H_0 используется видоизмененный критерий Хи-квадрат. Статистика $Z(\boldsymbol{x})$ этого критерия может быть определена как $(1/N) \sum\limits_{j=1}^{n} \left(f^*(j) - \psi_{\alpha\beta}(j)\right)^2 / \psi_{\alpha\beta}(j)$, где $f^*(j)$ — доля элементов выборки $\boldsymbol{x} = (x_1, \ldots, x_N)$, равных j. При вычислении $\psi_{\alpha\beta}(j)$ значения параметров α и β принимаются равными оценкам максимального правдоподобия.

Пусть задан уровень значимости γ и пусть $\chi^2_{\delta,r}$ — квантиль уровня δ распределения Хи-квадрат с r степенями свободы. Гипотеза H_0 отвергается тогда и только тогда, когда $Z(\boldsymbol{x}) > \chi^2_{1-\gamma,n-3}$ (напомним, что по выборке $\boldsymbol{x}$ оцениваются два параметра функции $\psi_{\alpha\beta}(j)$). Следуя этому критерию, можно ошибочно отклонить гипотезу H_0, когда она верна, с вероятностью, приближенно равной γ.

Достигнутым уровнем значимости критерия Хи-квадрат называется величина $\gamma^*(\boldsymbol{x}) = 1 - \chi^2_{n-3}(Z(\boldsymbol{x}))$, где $\chi^2_{n-3}(x)$ — функция распределения Хи-квадрат с $(n - 3)$ степенями свободы. Близость значения $\gamma^*(\boldsymbol{x})$ к 0 говорит о том, что гипотезу H_0 вероятнее всего следует отклонить.

Возникает вопрос, как построить выборку $\boldsymbol{x} = (x_1, \ldots, x_N)$ из распределения $f(j)$. Предлагается следующий способ. Построим t случайных матриц из M_{mn}: $L_1, \ldots, L_t$. Из каждого множества $P(L_k), k \in J_t$, выберем по u неприводимых покрытий. Возьмем $N = tu$ значений случайной величины η на парах матрица–покрытие и сформируем выборку.

Проведены эксперименты с выборками, построенными по матрицам размера $m \times n$ при $m = 20, 35, 50, 65$ и 80 и $n = 20, 35, 50, 65$ и 80. Для получения выборки формировалось по 200 случайных матриц одинакового размера и из множества $P(L)$ каждой матрицы L случайно выбиралось по 10 неприводимых покрытий. В результате получалась выборка длиной 2000. В табл. 1 приведены полученные в ходе экспериментов значения достигнутого уровня значимости $\gamma^*(\boldsymbol{x})$ и оценок максимального правдоподобия α и β.

Таблица 1. Достигнутый уровень значимости критерия и значения оценок максимального правдоподобия параметров, построенных по выборкам для матриц различного размера $m \times n$

(a) Достигнутый уровень значимости $\gamma^*(\boldsymbol{x})$

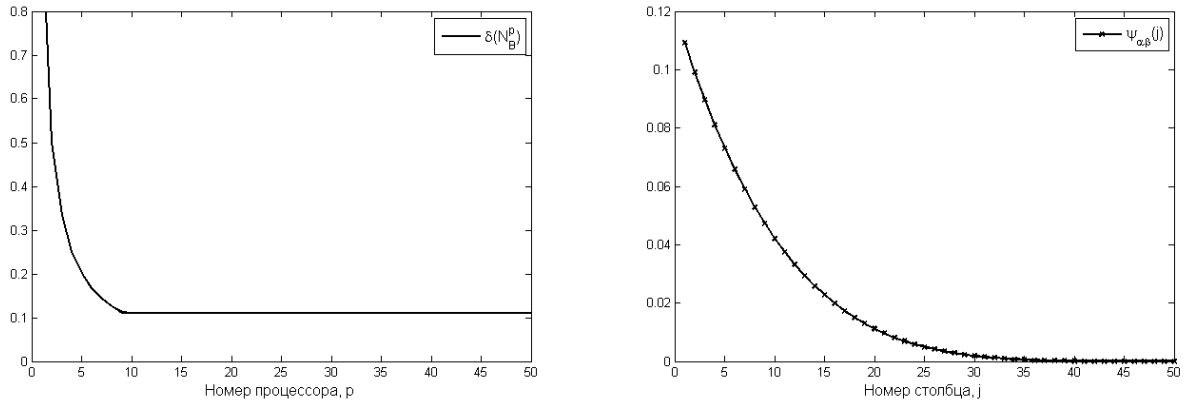
$m \setminus n$	20	35	50	65	80
20	0	0,106	0,326	0,502	0,751
35	0,001	0,517	0,992	0,842	0,974
50	0	0,919	0,515	0,647	0,983
65	0	0,638	0,719	0,327	0,556
80	0	0,303	0,969	0,79	0,252

(b) Оценка параметра α

$m \setminus n$	20	35	50	65	80
20	1,193	1,048	0,988	0,994	0,959
35	1,086	1,006	1,005	0,957	0,993
50	1,051	1,049	0,975	1,01	0,993
65	0,994	1,015	0,966	1,001	0,985
80	0,978	0,959	0,996	0,985	1,011

(c) Оценка параметра β

$m \setminus n$	20	35	50	65	80
20	6,062	5,118	4,762	4,941	4,75
35	6,694	5,901	5,649	5,427	5,51
50	7,283	6,51	6,12	6,26	6,14
65	7,013	6,887	6,432	6,633	6,491
80	7,727	6,887	6,938	6,962	6,69



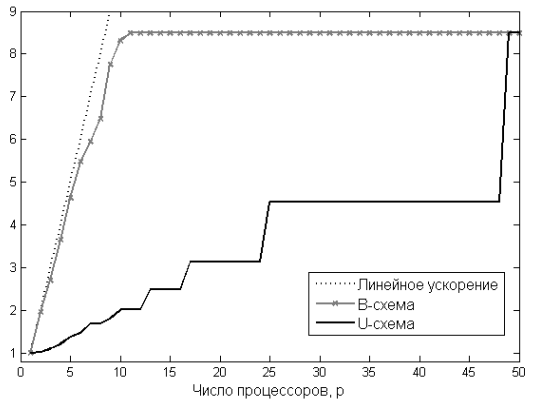
(а) (б)

Рис. 1. Пример зависимостей $\delta(\boldsymbol{N}_{\boldsymbol{B}}^{\boldsymbol{p}})$ от p и $\psi_{\alpha\beta}$ от j

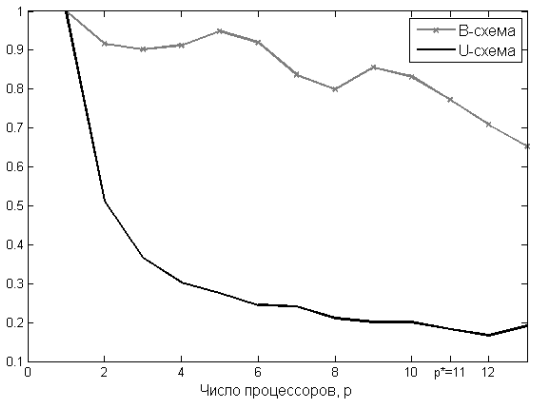
Проведенные эксперименты позволяют сделать следующие выводы. При $n = 20$ гипотезу H_0 следует отклонить (см. табл. 1, а). Возможной причиной этого является нерепрезентативность выборки, связанная с небольшим число неприводимых покрытий у матриц при $n = 20$. Матрицы размера 80×80, как правило, имеют большое число неприводимых покрытий (порядка 10^7), что также мешает получению репрезентативной выборки ограниченного объема используемым в работе способом. Достигнутый уровень значимости в этом случае достаточно мал. В остальных случаях гипотезу H_0 можно принять с большой уверенностью.

$\mathfrak{B}$-схема распараллеливания

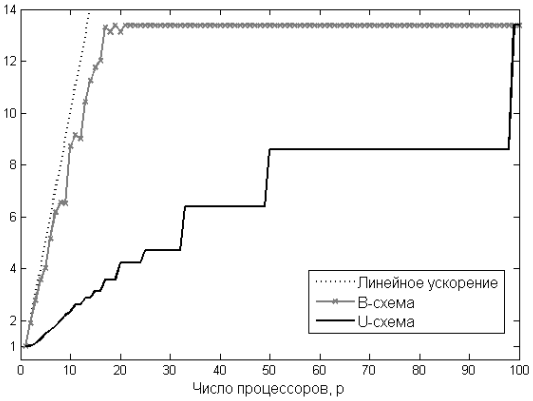
В этом разделе показано, как функцию вероятности $f(j)$ случайной величины $\eta(L, H)$ можно применять для эффективного распределения вычислительных ресурсов в начале



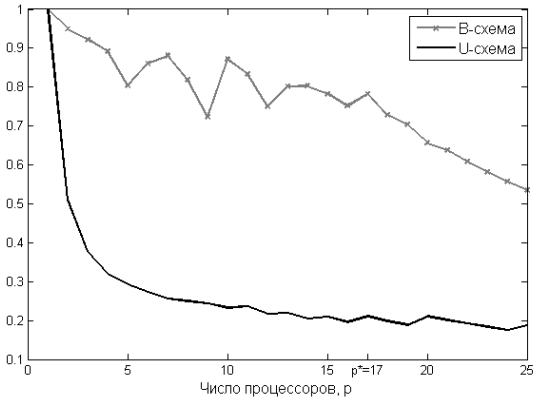
(а) $S(p)$ при $m = 50, n = 50$



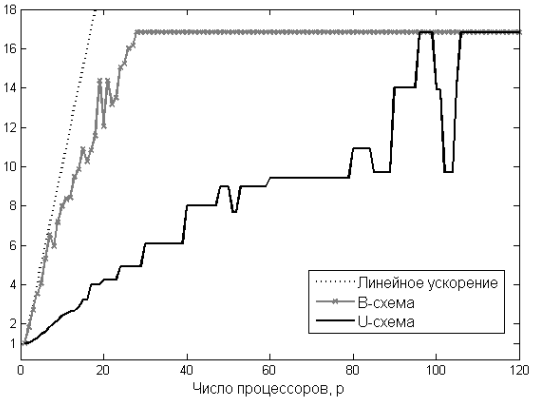
(б) $E(p)$ при $m = 50, n = 50$



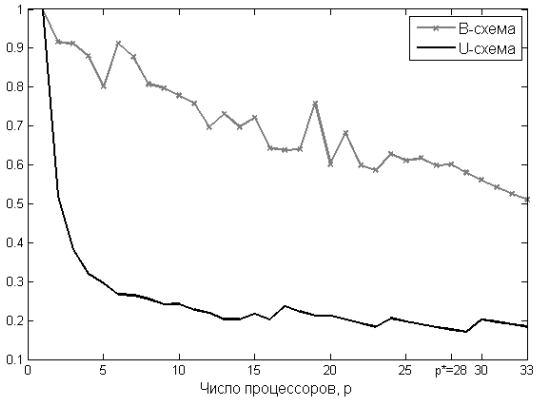
(в) $S(p)$ при $m = 50, n = 100$



(г) $E(p)$ при $m = 50, n = 100$



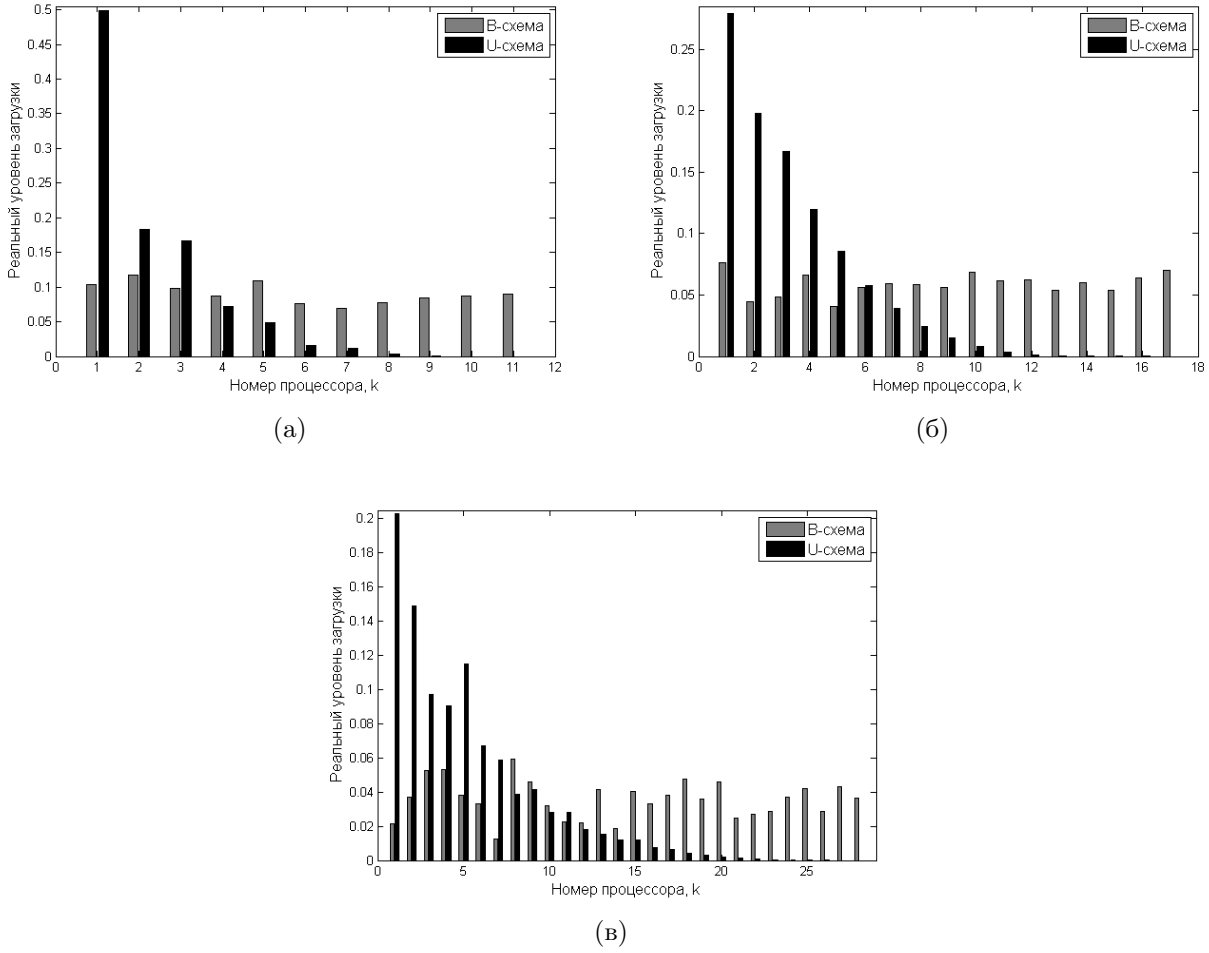
(д) $S(p)$ при $m = 30, n = 120$



(е) $E(p)$ при $m = 30, n = 120$

Рис. 2. Сильная масштабируемость

работы алгоритма дуализации. Пусть $L \in M_{mn}$ и пусть дано $p \leqslant n$ процессоров. Каждому из множеств $P_j(L), j \in J_n$, сопоставим процессор с номером N_j^p, который должен это множество построить. Вектор $\boldsymbol{N^p} = (N_1^p, \ldots, N_n^p)$ назовем расписанием. Уровнем загрузки



(а) (б)

(в)

Рис. 3. Графики $\sigma_p(k)$ для $p = p^*$ для различных конфигураций матриц: (а) $m = 50, n = 50, p = 11$; (б) $m = 50, n = 100, p = 17$; (в) $m = 30, n = 120, p = 28$

k-го процессора назовем величину

$$\delta_k(L, \boldsymbol{N^p}) = \sum_{j \in J_n : N_j^p = k} |P_j(L)| / |P(L)|. \tag{1}$$

Введем функцию вероятности в пространстве Ω как $\mathsf{P}\{L, H\} = 2^{-mn}|P(L)|^{-1}$, $L \in M_{mn}$, $H \in P(L)$. Нетрудно показать, что математическое ожидание $\mathsf{E}\big(|P_j(L)|/|P(L)|\big)$ равно $\mathsf{P}\{\eta(L, H) = j\}$, так как имеет место равенство:

$$|P_j(L)| = \sum_{H \in P(L)} \mathrm{Ind}(\eta(L, H) = j), \tag{2}$$

где

$$\mathrm{Ind}(x) = \begin{cases} 1, & \text{если } x \text{ истинно;} \\ 0 & \text{иначе.} \end{cases}$$

Из формул (1) и (2) следует, что средний уровень загрузки k-го процессора $\mathsf{E}\delta_k(L, \boldsymbol{N^p})$ равен

$$\sum_{j \in J_n : N_j^p = k} \mathsf{P}\{\eta(L, H) = j\}. \tag{3}$$

Для построения эффективной в типичном случае схемы распараллеливания требуется решить задачу минимизации

$$\delta(\boldsymbol{N^p}) = \max_{k \in J_p} \mathsf{E}\delta_k(L, \boldsymbol{N^p}) \to \min_{\boldsymbol{N^p}}. \tag{4}$$

Ниже приведено описание процедуры 5, которая ищет приближенное решение задачи 4 при помощи жадного алгоритма. На вход процедура принимает число процессоров p, число столбцов n обрабатываемой матрицы и функцию вероятности $f(j) = \mathsf{P}\{\eta(L, H) = j\}$.

Процедура 5 AllocateResources

Вход: p, n, f;
Выход: $\boldsymbol{N^p}$;
1: **для** $k = 1, \ldots, p$
2: $\quad \delta_k \leftarrow 0$
3: **для** $j = 1, \ldots, n$
4: $\quad k_0 \leftarrow \arg\min\limits_{k \in J_p} \delta_k$
5: $\quad N_j^p \leftarrow k_0$
6: $\quad \delta_k \leftarrow \delta_k + f(j)$

В предыдущем разделе показано, что $f(j) \approx \psi_{\alpha\beta}(j)$, где параметры α и β вычисляются с использованием оценок максимального правдоподобия. Пусть

$$\boldsymbol{N_B^p} \leftarrow \text{AllocateResources}\,(p, n, \psi_{\alpha\beta})\,.$$

Тогда схему распараллеливания, использующую расписание $\boldsymbol{N_B^p}$, назовем $\mathfrak{B}$-схемой. Эта схема не будет эффективна при большом числе процессоров. На рис. 1, а изображен пример зависимости $\delta(\boldsymbol{N_B^p})$ от p. Ясно, что при p не меньшем некоторого p^* функция $\delta(\boldsymbol{N_B^p})$ является постоянной. В этом случае

$$\delta(\boldsymbol{N_B^p}) = \mathsf{E}\delta_1(L, \boldsymbol{N_B^p}) = \psi_{\alpha\beta}(1),$$

что следует из поведения функции $\psi_{\alpha\beta}$, график которой приведен на рисунке 1, б для случая $\alpha = 1, \beta = 6$ и $n = 50$. Другими словами, при $p \geqslant p^*$ недостаточно задач для равномерной загрузки процессора.

Эксперименты и сравнение схем распараллеливания

Все численные эксперименты выполнены на системе с Intel Core i3-2370M и 8192 MB RAM. В предложенных схемах распараллеливания межпроцессорный обмен осуществляется на начальном этапе, когда управляющий процессор раздает задания.

Опишем тривиальную схему распараллеливания асимптотически оптимальных алгоритмов дуализации ($\mathfrak{U}$-схему), с которой будет сравниваться $\mathfrak{B}$-схема. Пусть дано p процессоров и $L \in M_{mn}$. По аналогии с $\mathfrak{B}$-схемой работа $\mathfrak{U}$-схемы основана на использовании расписания $\boldsymbol{N^p} = (N_1^p, \ldots, N_n^p)$, где $N_j^p = \lceil pj/n \rceil, j \in J_n$, и $\lceil x \rceil$ равно ближайшему целому, не меньшему x. Другими словами, первые примерно n/p столбцов должны быть обработаны первым процессором, следующие n/p столбцов — вторым и т. д.

Пусть $L \in M_{mn}$. Фиксируем схему распараллеливания. Обозначим через $T_p(k)$ время работы k-го из p процессоров. Тогда время работы $T(p)$ параллельного алгоритма будет

равно максимальному из $T_p(k)$. Фактическим уровнем загрузки k-го процессора назовем величину $\sigma_p(k) = T_p(k) \big/ \sum_{k \in J_p} T_p(k)$.

Для исследования сильной масштабируемости предложенных схем распараллеливания (зависимости основных показателей работы алгоритма от количества процессоров при фиксированном размере входа задачи) рассмотрим такие показатели, как ускорение алгоритма $S(p) = T(1)/T(p)$ и средняя загрузка $E(p) = S(p)/p$. Значения $E(p)$, близкие к единице, свидетельствуют о равномерности загрузки процессоров.

На рис. 2 приведены результаты работы параллельных версий алгоритма АО2 (с $\mathfrak{B}$-схемой и $\mathfrak{U}$-схемой) для матриц размера $50{\times}50, 50{\times}100$ и $30{\times}120$. Согласно графикам функции $S(p)$, $\mathfrak{B}$-схема эффективна лишь при $p \leqslant p^*$. Например, для матриц $50{\times}50$ величина p^* равна 11. Видно, что при $p \leqslant p^*$ ускорение $S(p)$ для $\mathfrak{B}$-схемы близко к линейному, что является несомненным преимуществом перед $\mathfrak{U}$-схемой. На рис. 3 в виде столбчатых диаграмм приведены функции $\sigma_{p^*}(k), k \in J_{p^*}$, демонстрирующие уровень загруженности процессоров. Во всех рассмотренных случаях $\mathfrak{B}$-схема существенно превосходит $\mathfrak{U}$-схему по равномерности загрузки процессоров.

Заключение

В данной работе предложена эффективная схема распараллеливания ($\mathfrak{B}$-схема) асимптотически оптимальных алгоритмов дуализации.

Пусть на вход алгоритму подана матрица L с n столбцами. При наличии $p \leqslant n$ процессоров схема строит расписание $N^p = (N_1^p, \ldots, N_n^p)$, где N_j^p равно номеру процессора, который должен построить подмножество неприводимых покрытий $P_j(L), j \in J_n$, которые начинаются с j-го столбца.

На пространстве элементарных событий $\Omega = \{(L, H) \,|\, L \in M_{mn}, H \in P(L)\}$ введена случайная величина $\eta(L, H)$, равная наименьшему номеру столбца в H. Выдвинута статистическая гипотеза о том, что величина $\eta(L, H) - 1$ имеет бета-биномиальное распределение $\mathfrak{B}(\alpha, \beta, n-1)$, и эта гипотеза подтверждена экспериментально. $\mathfrak{B}$-схема использует бета-биномиальную аппроксимацию распределения случайной величины $\eta(L, H)$ для построения расписания N^p.

Рассмотрена более простая схема распараллеливания — $\mathfrak{U}$-схема, в которой все процессоры строят примерно одинаковое число множеств $P_j(L)$.

Эксперименты показывают, что одним из главных преимуществ $\mathfrak{B}$-схемы перед $\mathfrak{U}$-схемой является значительно более равномерная загрузка процессоров и почти линейное ускорение $S(p)$. Однако указанная схема является эффективной при числе процессоров, существенно меньшем n. Поэтому в рамках дальнейших исследований планируется избавиться от указанного ограничения, разработав схемы для произвольного числа процессоров.

Литература

[1] *Johnson D. S., Yannakis M., Papadimitriou C. H.* On generating all maximal independent sets // *Inform. Process. Lett.*, 1988. Vol. 27. P. 119–123.

[2] *Дюкова Е. В.* Об асимптотически оптимальном алгоритме построения тупиковых тестов // *ДАН СССР*, 1977. Т. 223. №4. С. 527–530.

[3] *Дюкова Е. В.* О сложности реализации дискретных (логических) процедур распознавания // *Журнал вычислительной математики и математической физики*, 2004. Т. 44. №3. С. 551–561.

[4] *Дюкова Е. В., Инякин А. С.* Асимптотически оптимальное построение тупиковых покрытий целочисленной матрицы // *Математические вопросы кибернетики*, 2008. Т. 17. С. 235–246.

[5] *Murakami K., Uno T.* Efficient algorithms for dualizing large-scale hypergraphs. Tokyo: Institute of Informatics, 2011.

References

[1] *Johnson D. S., Yannakis M., Papadimitriou C. H.* 1988. On generating all maximal independent sets. *Inform. Process. Lett.* 27:119–123.

[2] *Djukova E. V.* On an asympotically optimal algorithm for constructing irredundant tests. 1977. *Dokl. Akad. Nauk SSSR* 233(4):423–426.

[3] *Dyukova E. V.* On the implementation complexity of the realization of discrete (logical) recognition procedures. 2004. *Comp. Math. Math. Phys.* 44(3):532–541.

[4] *Djukova E. V., Inyakin A. S.* 2008. Asimptoticheski optimal'noe postroenie tupikovykh pokrytiy tselochislennoy matritsy [Asymptotically optimal construction of irredundant coverings of an integer matrix]. *Matematicheskie voprosy kibernetiki* [Mathematical Problems of Cybernetics] 17:235–246.

[5] *Murakami K., Uno T.* 2011. Efficient algorithms for dualizing large-scale hypergraphs. Tokyo: Institute of Informatics.