Об эффективном распараллеливании алгоритмов для дискретных перечислительных задач*

***Е. В. Дюкова*[1], *А. Г. Никифоров*[2]**
edjukova@mail.ru, ankifor@gmail.com
[1]Вычислительный центр РАН им. А. А. Дородницына, Москва, Россия
[2]Московский государственный университет им. М. В. Ломоносова, Москва, Россия

Разработана новая статическая схема распараллеливания асимптотически оптимальных алгоритмов для задачи дуализации. Данная задача относится к числу труднорешаемых перечислительных задач. Предлагаемая схема основана на предварительной статистической обработке входных данных с целью установления вида распределения случайной величины, определяющей объемы подзадач. Статья является развитием ранней работы авторов, в которой при получении указанных оценок использовалась менее эффективная методика, учитывающая только размер задачи. Выявлены условия, при которых обеспечиваются достаточно равномерная загрузка процессоров и ускорение, близкое к максимальному.

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

DOI: 10.21469/22233792.1.13.07

On efficient parallelizing of the algorithms for discrete enumeration problems*

***E. V. Djukova*[1] *and A. G. Nikiforov*[2]**
[1]Dorodnicyn Computing Centre of the Russian Academy of Sciences, 40 Vavilova st., Moscow, Russia
[2]Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, Russia

Background: Approach to construction of efficient parallel algorithms for discrete enumeration problems is introduced in the previous works of the authors. This approach is based on statistical estimations for computational tasks size. The approach is demonstrated on dualization, which is an intractable problem and consists in enumeration of irreducible coverings of a given boolean matrix. The main disadvantage of formerly suggested parallel schemes for asympotically optimal dualization algorithms is time-costly tasks size estimation method which considers only the problem size.

Methods: A new parallel scheme has been developed for asymptotically optimal dualization algorithms, reducing time costs on statistical data collection. Statistical data are obtained via processing of a given matrix submatrices.

Results: Task distribution is performed according to schedule calculated in advance. For this purpose, a distribution of random variable, used for tasks size estimation, is fitted and the processor load level is optimized. A parallel scheme is applied to an asymptotically optimal algorithm RUNC-M.

Concluding Remarks: A new parallel scheme works not worse than the formerly suggested ones, demonstrates an almost maximal speedup and makes it possible to dualize matrices of big

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

size. However, this scheme is efficient only if the number of processors is significantly smaller than the number of matrix columns.

Keywords: *enumeration; dualization; irreducible covering of a boolean matrix; hypergraph transversal; asymptotically optimal algorithm; parallel computations; load balancing; strong scaling*

DOI: 10.21469/22233792.1.13.07

1 Введение

Одной из фундаментальных задач дискретной математики является дуализация. Ниже приведена ее матричная формулировка.

Дана булева матрица L размера $m \times n$. Набор H, состоящий из различных столбцов матрицы L, называется неприводимым покрытием, если он удовлетворяет двум условиям: (1) в подматрице L^H матрицы L, образованной столбцами набора H, не содержится строки вида $(0, 0, \ldots, 0)$; (2) подматрица L^H содержит каждую из строк вида $(1, 0, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, $\ldots$, $(0, 0, 0, \ldots, 1)$. Если набор столбцов H удовлетворяет условию (1), то он называется покрытием. Если набор столбцов H удовлетворяет условию (2), то он называется совместимым. Требуется построить множество $P(L)$ всех неприводимых покрытий матрицы L.

Дуализация имеет и другие эквивалентные формулировки. Приведем основные из них.

1. Дана конъюнктивная нормальная форма, реализующая монотонную булеву функцию F. Требуется построить сокращенную дизъюнктивную нормальную форму функции F.
2. Дан гиперграф G. Требуется перечислить все минимальные трансверсали гиперграфа G (двойственной задачей является задача перечисления всех минимальных вершинных покрытий гиперграфа G).

Дуализация возникает во многих областях дискретной математики (комбинаторике, теории гиперграфов, целочисленном программировании), в теории игр, в теории баз данных, в теории машинного обучения и т. д.

Асимптотические оценки типичных значений числа решений дуализации [1] показывают, что, как правило, это число растет экспоненциально с ростом размера входных данных. Поэтому дуализация относится к числу труднорешаемых перечислительных задач. Существует несколько подходов к оценке эффективности алгоритмов для перечислительных задач [1–3].

Говорят, что алгоритм работает с (квази)полиномиальной задержкой, если на каждом шаге строится ровно одно решение и сложность шага ограничивается (квази)полиномом от размера входных данных (для дуализации в матричной формулировке — это размер матрицы L). Алгоритмы дуализации с (квази)полиномиальной задержкой удалось построить только для некоторых частных случаев (например, когда в каждой строке исходной матрицы не более двух единиц [3]). Таким образом, статус дуализации в плане полиномиальной разрешимости неизвестен.

В [1, 4] предложен подход к построению асимптотически оптимальных алгоритмов дуализации. В дальнейшем этот подход получил развитие в [5–8]. На каждом шаге асимптотически оптимального алгоритма строится набор столбцов матрицы, удовлетворяющий условию совместимости (2). В отличие от алгоритма дуализации с полиномиальной задержкой асимптотически оптимальный алгоритм может делать полиномиальные «лиш-

ние» шаги, причем доля таких шагов стремится к нулю для почти всех матриц L размера $m\times n$ при $m, n \to \infty$. На «лишнем» шаге либо строится набор столбцов матрицы, не являющийся покрытием, либо строится набор столбцов, найденный ранее. Проверка на повторяемость построенного набора столбцов осуществляется за полиномиальное время от размеров матрицы.

Асимптотически оптимальные алгоритмы являются лидерами по скорости счета среди других известных алгоритмов дуализации. В [9, 10] показано, что наиболее быстро среди асимптотически оптимальных алгоритмов работают алгоритмы RUNC-M и PUNC [10], не делающие «повторных» шагов.

В силу того что число решений дуализации, как правило, растет экспоненциально с ростом размеров входных данных, актуальным является использование параллельных вычислений. Существуют простые и очевидные схемы распараллеливания асимптотически оптимальных алгоритмов дуализации, основным недостатком которых является неравномерная загрузка процессоров, что приводит и к недостаточному ускорению времени работы параллельного алгоритма по сравнению с его последовательной версией. Схема распараллеливания определяется способом выбора вычислительных подзадач и способом распределения этих подзадач между процессорами.

Следует отметить, что за рубежом при создании параллельных алгоритмов дуализации первостепенное внимание уделяется теоретическим оценкам их сложности в зависимости от числа используемых процессоров [11, 12], причем, как правило, строятся алгоритмы, ориентированные на частные случаи, например, когда число единиц в каждой строке исходной матрицы ограничено некоторой небольшой константой.

В [13] предложен подход к построению эффективных в практическом плане параллельных асимптотически оптимальных алгоритмов дуализации. Опишем разработанную в [13] B-схему распараллеливания.

Пусть H — неприводимое покрытие матрицы L, состоящее из столбцов с номерами $j_1, \ldots, j_r$, где $j_1 < \cdots < j_r$. Тогда H назовем j_1-неприводимым покрытием. Подзадача с номером j, $j \in \{1, \ldots, n\}$, состоит в построении множества $P_j(L)$ всех j-неприводимых покрытий матрицы L. Объемы подзадач определяются величинами $\nu_j(L) = |P_j(L)|/|P(L)|$, $j \in \{1, \ldots, n\}$.

B-схема имеет статический характер: распределение подзадач происходит по заранее составленному «расписанию». Статистическая обработка экспериментов показывает, что случайная величина, использующаяся для оценки $\nu_j(L), j \in \{1, \ldots, n\}$, подчиняется бета-биномиальному закону, параметры которого вычисляются при помощи метода максимального правдоподобия по выборке из случайных матриц размера $m\times n$. Для составления «расписания» решается задача оптимизации уровня загрузки процессоров.

B-схема демонстрирует, как правило, достаточно равномерную загрузку процессоров и высокое ускорение времени работы при увеличении числа процессоров. Однако вычисление оценок для $\nu_j(L)$ в этой схеме требует многократного решения задачи дуализации для случайных матриц, имеющих одинаковый размер с матрицей L, и эти оценки недостаточно точны.

Основным результатом данной работы является разработка S-схемы распараллеливания асимптотически оптимальных алгоритмов дуализации. Эта схема отличается от B-схемы методом получения оценок для $\nu_j(L)$, $j \in \{1, \ldots, n\}$, который является менее трудоемким и учитывает не только размеры матрицы. Метод основан на обработке случайных подматриц данной матрицы, подматрицы имеют размер $r\times n$, где r является параметром и не превосходит m. Выявлено, что при параметре r, равном $m/2$, полученные оценки

являются достаточно точными с точки зрения критерия Хи-квадрат. Работа S-схемы продемонстрирована на примере алгоритма RUNC-M [10], который является модификацией асимптотически оптимального алгоритма ОПТ [8].

При тестировании S-схемы исследуется ее сильная масштабируемость (зависимость основных показателей работы параллельного алгоритма от числа процессоров при фиксированном размере задачи). Показано, что S-схема демонстрирует такие же показатели сильной масштабируемости, как B-схема, и в то же время позволяет обрабатывать матрицы еще бóльших размеров за счет более быстрого вычисления оценок для объемов подзадач.

2 Описание асимптотически оптимального алгоритма дуализации RUNC-M

В данном разделе приводится описание асимптотически оптимального алгоритма дуализации RUNC-M [10]. Подробно описана структура дерева решений, которое строит данный алгоритм.

Обозначим через M_{mn} множество булевых матриц размера $m\times n$, а через J_u множество $\{1, 2, \ldots, u\}$. Пусть $L = (a_{ij}) \in M_{mn}$. В данном разделе будем отождествлять набор столбцов (строк) матрицы L с набором их номеров.

Будем говорить, что столбец j (столбец с номером j) покрывает строку i (строку с номером i) матрицы L, если $a_{ij} = 1$.

Строка i матрицы L является опорной для пары (H, j), $j \in H$, если $a_{ij} = 1$ и $a_{ij} = 0\ \forall u \in H \setminus \{j\}$. Множество всех опорных строк для (H, j) обозначим через $S(H, j)$. Очевидно, набор H является совместимым тогда и только тогда, когда $S(H, j) \neq \varnothing\ \forall j \in H$.

Говорят, что столбец j матрицы L совместим с совместимым набором H, если набор $H \cup \{j\}$ совместимый. Очевидно, столбец j не совместим с совместимым набором H тогда и только тогда, когда $\exists u \in H$ такой, что столбец j покрывает все строки из $S(H, u)$.

Асимптотически оптимальный алгоритм дуализации RUNC-M перечисляет с полиномиальной задержкой $O(qmn), q = \min\{m, n\}$, некоторое подмножество совместимых наборов столбцов матрицы L, содержащее множество $P(L)$. Алгоритм строит дерево решений, совершая его обход в глубину. Построение одной висячей вершины — это шаг алгоритма.

Вершина (H, R, C) дерева решений описывается совместимым набором столбцов H, набором строк R и набором столбцов C. В висячей вершине имеет место один из двух случаев: (1) $R = \varnothing$; (2) $R \neq \varnothing, C = \varnothing$. В первом случае H — неприводимое покрытие. Во втором случае висячая вершина соответствует «лишнему» шагу. Корню дерева соответствует $(H = \varnothing, R = J_m, C = J_n)$. Пусть построена внутренняя (не висячая) вершина (H, R, C), тогда переход к следующей построенной вершине будет осуществляться путем добавления к H столбца из C и удаления некоторых строк и столбцов из R и C соответственно.

Пусть построена внутренняя вершина (H_0, R_0, C_0). Тогда при построении следующей вершины к H_0 добавляется первый по порядку столбец из $C_0^{\min} = \{j \in C_0 | a_{ij} = 1\}$, где $i \in R_0$ — номер строки с наименьшей суммой $\sum_{j\in C_0} a_{ij}$ (если таких строк несколько, то выбирается строка с наименьшим номером среди них).

Для того чтобы схемы распараллеливания, описанные далее, были применимы к алгоритму RUNC-M, требуется его немного модифицировать: на первом ярусе дерева решений, или когда глубина рекурсии равна нулю, вместо множества $C_0^{\min}$ берется множество $C_0 = J_n$.

Алгоритм 1 BuildSubtreeRUNCM

Вход: L, H_0, R_0, C_0;
Выход: $\varnothing$;
1: $C_0^{\min} = \{j \in C_0 | a_{ij} = 1\}$, где $i \in R_0$ — номер строки с наименьшей суммой $\sum_{j \in C_0} a_{ij}$
2: **для всех** $j \in C_0^{\min}$
3: $R \leftarrow R_0$
4: $C_0 \leftarrow C_0 \setminus \{j\}$
5: $C \leftarrow C_0$
6: $H \leftarrow H_0 \cup \{j\}$
7: Удалить из R строки, покрытые столбцом j
8: **если** $R = \varnothing$ **то**
9: Сохранить набор H, который является неприводимым покрытием
10: **иначе**
11: Удалить из C не совместимые с набором H столбцы
12: BuildSubtreeRUNCM(L, H, R, C)

Опишем рекурсивную процедуру BuildSubtreeRUNCM(L, H, R, C) (см. Алгоритм 1) построения поддерева решений. Для запуска алгоритма эту функцию следует вызывать с параметрами $H = \varnothing, R = J_m, C = J_n$. Отметим, что все аргументы процедуры передаются по значению или копируются.

Настоящая реализация алгоритма RUNC-M написана на языке `C++` с интенсивным использованием побитовых операций. Кроме того, для некоторых частей алгоритма используется динамический выбор функций для минимизации числа операций (например, этот прием используется при удалении несовместимых строк).

3 Оценки для объемов подзадач, используемые в S-схеме

В данном разделе описаны способы оценки объемов подзадач или величин $\nu_j(L) = |P_j(L)|/|P(L)|$, $j \in J_n$, используемые соответственно в B-схеме [13] и S-схеме.

В B-схеме на пространстве равновероятных элементарных событий $\Omega = \{(L, H) \,|\, L \in M_{mn}, H \in P(L)\}$ вводится случайная величина $\eta(L, H)$, равная j, если $H \in P_j(L), j \in J_n$. При помощи критерия Хи-квадрат проверяется гипотеза о виде распределения $H_0 : f(j) = \psi_{\alpha\beta}(j)$, где $f(j)$ — вероятность события $\eta(L, H) = j$, а $\psi_{\alpha\beta}(j)$ — функция вероятности бета-биномиального распределения с параметрами α и β, которые оцениваются при помощи метода максимального правдоподобия. B-схема использует величины $\psi_{\alpha\beta}(j)$ в качестве приближенного значения искомой величины $\nu_j(L), j \in J_n$. Эта схема обладает двумя основными недостатками: оценка для $\nu_j(L)$ одна и та же для всех матриц данного размера, и для ее вычисления требуется многократно решить задачу дуализации для случайных матриц из M_{mn}.

Теперь опишем S-схему.

Пусть $L \in M_{mn}$ и $r \leqslant m$. Через W_m^r обозначим множество всех подмножеств мощности r множества J_m. Пусть $w \in W_m^r$, тогда через L^w обозначим подматрицу матрицы L, составленную из строк матрицы L с номерами из w.

Пусть $\Omega_r = \{(L^w, H) \,|\, w \in W_m^r, H \in P(L)\}$ — пространство равновероятных элементарных событий. На указанном пространстве определим случайную величину $\eta_r(L^w, H)$,

Таблица 1 Значения пар $(Z_r(\boldsymbol{x}), \gamma_r^*(\boldsymbol{x}))$ для критерия Хи-квадрат

$r \setminus m{\times}n$	30×120	40×120	50×100	70×70
10	$(159, < 10^{-4})$	$(167, < 10^{-4})$	$(235, < 10^{-4})$	$(382, < 10^{-4})$
13	$(99, < 10^{-4})$	$(132, < 10^{-4})$	$(157, < 10^{-4})$	$(234, < 10^{-4})$
15	$(77, 0{,}0134)$	$(112, < 10^{-4})$	$(117, < 10^{-4})$	$(187, < 10^{-4})$
18	$(74, 0{,}028)$	$(90, 0{,}0002)$	$(96, < 10^{-4})$	$(147, < 10^{-4})$
20	$(60, 0{,}0815)$	$(63, 0{,}0546)$	$(89, < 10^{-4})$	$(131, < 10^{-4})$
25	$(54, 0{,}315)$	$(60, 0{,}0876)$	$(50, 0{,}1382)$	$(85, < 10^{-4})$
30	—	—	—	$(68, 0{,}0001)$
35	—	—	—	$(54, 0{,}0478)$

которая равна $j, j \in J_n$, если $H \in P_j(L)$. Через $f_r(j)$ обозначим вероятность события $\eta_r(L^w, H) = j$.

Предлагается использовать величину $f_r(j)$ в качестве приближенного значения искомой величины $\nu_j(L), j \in J_n$. Встает вопрос, при каких r указанные оценки являются достаточно точными. С одной стороны, число r должно быть как можно меньшим, чтобы время получения оценок было относительно невелико. С другой стороны, оценки должны быть достоверными.

Пусть $\boldsymbol{x} = (x_1, \ldots, x_N)$ — выборка из распределения $f_r(j)$. Для проверки статистической гипотезы $H_0 : f_r(j) = \nu_j(L)$ о виде распределения случайной величины $\eta_r(L^w, H)$ предлагается использовать критерий Хи-квадрат со статистикой

$$Z_r(\boldsymbol{x}) = N \sum_{j=1}^{n} \frac{(f^*(j) - \nu_j(L))^2}{\nu_j(L)},$$

где $f^*(j)$ — доля элементов выборки $\boldsymbol{x} = (x_1, \ldots, x_N)$, равных j.

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

Для получения выборки $\boldsymbol{x} = (x_1, \ldots, x_N)$ из распределения $f_r(j)$ построим t случайных подматриц L^w матрицы L размера $r{\times}n$. Выберем N пар $(L^w, H), H \in P(L^w)$, и из значений случайной величины $\eta_r(L^w, H)$ сформируем выборку.

Проведем эксперимент. Для каждой из конфигураций 30×150, 40×120, 50×100 и 70×70 выберем по 20 случайных матриц. Для каждой матрицы сформируем выборку $\boldsymbol{x} = (x_1, \ldots, x_N)$ из распределения $f_r(j)$, где $N = 1000$. В табл. 1 приведены медианные значений статистики $Z_r(\boldsymbol{x})$ и достигаемых уровней значимости $\gamma_r^*(\boldsymbol{x})$. На рис. 1 приведены графики величин $\nu_j(L)$ и $f_r^*(j)$.

Согласно табл. 1 минимальное значение r, при котором достигнутый уровень значимости $\gamma_r^*(\boldsymbol{x})$ не является пренебрежимо малым, равняется $m/2$. На примере конфигурации 30×150 можно заметить, что при $r = 15$ имеет место «фазовый переход»: при пересечении этой точки функция $Z_r(\boldsymbol{x})$ начинает стабилизироваться. Это говорит о том, что дальнейшее увеличение r не принесет существенного выигрыша в приближении $\nu_j(L)$.

4 Распределение вычислительных заданий между процессорами

Пусть $L \in M_{mn}$ и пусть дано $p \leqslant n$ процессоров. Пусть j-я подзадача обрабатывается процессором с номером N_j. Вектор $\boldsymbol{N^p} = (N_1, \ldots, N_n)$ назовем расписанием. Уровнем

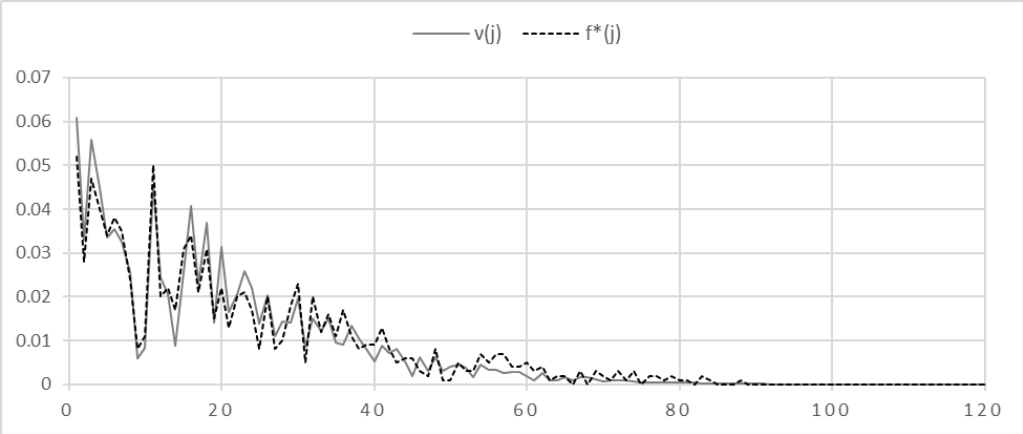


Рис. 1 Графики $\nu_j(L)$ и $f_r^*(r)$ как величин, зависящих от j, при $m = 30$, $n = 120$ и $r = 15$

загрузки k-го процессора назовем величину

$$\sigma_k(\boldsymbol{N^p}) = \sum_{j \in J_n : N_j = k} \nu_j(L).$$

Для эффективного распределения вычислительных заданий между процессорами, требуется решить задачу минимизации уровня загрузки процессоров

$$\sigma(\boldsymbol{N^p}) = \max_{k \in J_p} \sigma_k(\boldsymbol{N^p}) \to \min_{\boldsymbol{N^p}}. \tag{1}$$

Ниже приведено описание процедуры 2, которая ищет приближенное решение задачи 1 при помощи жадного алгоритма. На вход этой процедуры подается число процессоров p, число столбцов n матрицы L и вектор $\widetilde{\boldsymbol{\nu}} = (\widetilde{\nu}_1, \ldots, \widetilde{\nu}_n)$, состоящий из оценок для величин $\nu_j(L)$. Способы получения оценок $\widetilde{\nu}_j$ описаны ранее в этой работе.

Алгоритм 2 DistributeTasks

Вход: $p, n, \widetilde{\boldsymbol{\nu}}$;
Выход: $\boldsymbol{N^p}$;

1: **для всех** $k \in \{1, \ldots, p\}$
2: $\quad \sigma_k \leftarrow 0$
3: **для** $j \in \{1, \ldots, n\}$
4: $\quad k_0 \leftarrow \arg\min_{k \in J_p} \sigma_k$
5: $\quad N_j \leftarrow k_0$
6: $\quad \sigma_k \leftarrow \sigma_k + \widetilde{\nu}_j$

5 Тестирование S-схемы распараллеливания

В данном разделе даны описания среды тестирования и исследуемых показателей работы параллельных алгоритмов, приведены результаты сравнения S-схемы и B-схемы и результаты дополнительного тестирования S-схемы на больших матрицах.

Таблица 2 Сравнение схем распараллеливания при $m = 65$ и $n = 80$

p	B-схема			S-схема		
	$T(p)$	$\sigma(p)$	$s(p)$	$T(p)$	$\sigma(p)$	$s(p)$
1	17,40	1,000	1,000	18,35	1,000	1,000
2	9,01	0,500	0,514	9,40	0,500	0,502
4	5,30	0,250	0,291	4,92	0,250	0,261
8	2,71	0,125	0,147	2,52	0,125	0,135
16	1,55	0,079	0,090	1,62	0,084	0,087
32	1,55	0,079	0,090	1,61	0,089	0,087
64	1,55	0,079	0,090	1,61	0,089	0,087

Таблица 3 Сравнение схем распараллеливания при $m = 80$ и $n = 65$

p	B-схема			S-схема		
	$T(p)$	$\sigma(p)$	$s(p)$	$T(p)$	$\sigma(p)$	$s(p)$
1	26,1	1,000	1,000	26,2	1,000	1,000
2	13,7	0,500	0,514	13,8	0,500	0,507
4	7,17	0,250	0,271	7,01	0,250	0,255
8	3,87	0,125	0,140	3,79	0,126	0,137
16	2,83	0,102	0,114	3,13	0,086	0,123
32	2,83	0,102	0,114	3,13	0,092	0,123
64	2,83	0,102	0,114	3,13	0,092	0,123

Тестирование проводилось на суперкомпьютере IBM Blue Gene/P, располагающегося в МГУ им. М. В. Ломоносова в здании факультета Вычислительной математики и кибернетики и являющегося массивно-параллельной вычислительной системой. Каждый вычислительный узел включает в себя четырехъядерный процессор PowerPC 450 (850 МГц), 2 ГБ общей памяти и сетевые интерфейсы. При запуске вычислительных заданий использовался режим виртуальных вычислительных узлов (VN (virtual network) режим). В этом режиме на каждом вычислительном узле запущено четыре MPI (message passing interface) процесса, которые делят между собой доступные ресурсы.

Через p обозначим число процессоров, через $T_k(p)$ — время (в секундах) работы k-го процессора параллельного алгоритма при использовании p процессоров. Пусть $T(p) = \max\limits_k T_k(p)$ и $T_\Sigma(p) = \sum_k T_k(p)$. Достигнутым уровнем загрузки k-го процессора назовем величину $s_k(p) = T_k(p)/T_\Sigma(p)$. Исследуются три показателя:

(1) ускорение алгоритма $S(p) = T(1)/T(p)$;
(2) равномерность загрузки процессоров $E(p) = S(p)/p$;
(3) достигнутый уровень загрузки $s(p) = \max\limits_k s_k(p)$.

Ускорение, соответствующее линейной функции $S(p) = p$ при $p \geqslant 1$, является практически максимальным. Близость функции $E(p)$ к единице свидетельствует о равномерной загрузке процессоров. Показатель $s(p)$ является аналогом показателя уровня загрузки процессоров $\sigma(\boldsymbol{N^p})$, определенного ранее в этой работе.

Поскольку используемые в B-схеме оценки для $\nu_j(L)$ одинаковы для всех матриц данного размера, далее эти оценки считаются известными во время работы параллельного

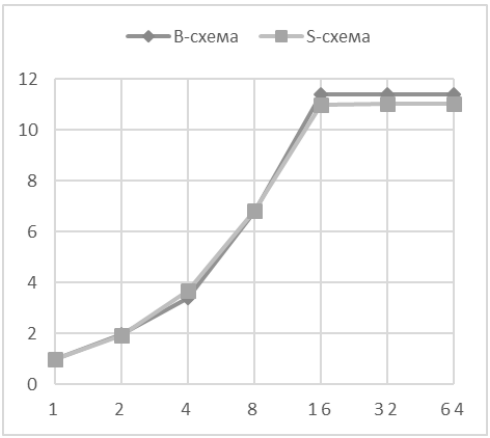
Таблица 4 Сравнение схем распараллеливания при $m = 80$ и $n = 80$

p	B-схема			S-схема		
	$T(p)$	$\sigma(p)$	$s(p)$	$T(p)$	$\sigma(p)$	$s(p)$
1	34,4	1,000	1,000	36,5	1,000	1,000
2	17,3	0,500	0,502	18,8	0,500	0,508
4	10,1	0,250	0,286	9,94	0,250	0,257
8	5,10	0,125	0,147	5,35	0,125	0,137
16	3,03	0,078	0,091	3,33	0,074	0,085
32	3,03	0,078	0,091	3,32	0,076	0,085
64	3,03	0,078	0,091	3,32	0,076	0,085

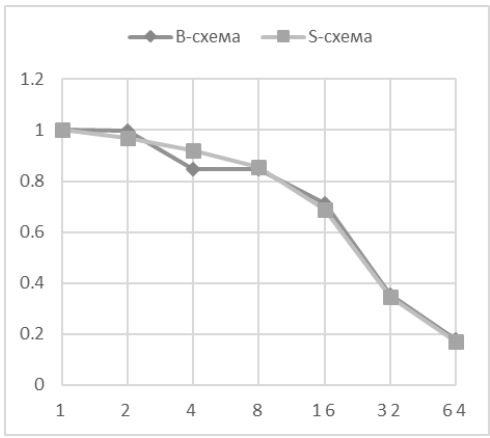
алгоритма. В S-схеме, напротив, оценки для объемов подзадач подсчитываются во время работы параллельного алгоритма.

Сравнение B-схемы и S-схемы проводится на матрицах размера 65×80, 80×65 и 80×80. Матрицы бóльших конфигураций при сравнении не рассматриваются ввиду трудоемкости получения оценок для B-схемы. Результаты представлены в табл. 2–4. На рис. 2 представлены графики функций $S(p)$ и $E(p)$ при $m = n = 80$.

Из указанных таблиц и графиков следует, что обе схемы демонстрируют практически одинаковое ускорение $S(p)$ и равномерность загрузки $E(p)$. При этом достигнутый уровень загрузки $s(p)$ у S-схемы наиболее низкий и $s(p) \approx \sigma(p)$, что свидетельствует о качественной балансировке нагрузки.



(*а*) График $S(p)$ при $m = 80$ и $n = 80$



(*б*) График $E(p)$ при $m = 80$ и $n = 80$

Рис. 2 Сравнение схем распараллеливания

Для каждого из размеров матриц можно указать число p^* такое, что при $p \leqslant p^*$ обе схемы эффективны: $S(p) \approx p$ и $E(p) \approx 1$. Например, при $m = n = 80$ число p^* равно 16. При $p > p^*$ значения $T(p)$ «стабилизируются». Это связано с тем, что распараллеливание происходит на первом ярусе дерева решений, которое строит алгоритм RUNC-M. При таком подходе объемы подзадач сильно различаются, поэтому их принципиально невозможно равномерно распределить между большим числом процессоров.

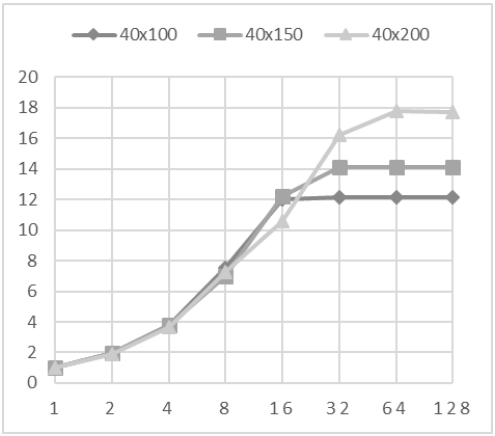
Вычисление оценок для объемов подзадач, используемых в S-схеме, гораздо менее трудоемкое, чем в B-схеме. Поэтому S-схема применима и для больших матриц. Это демон-

Таблица 5 Время работы $T(p)$ для S-схемы

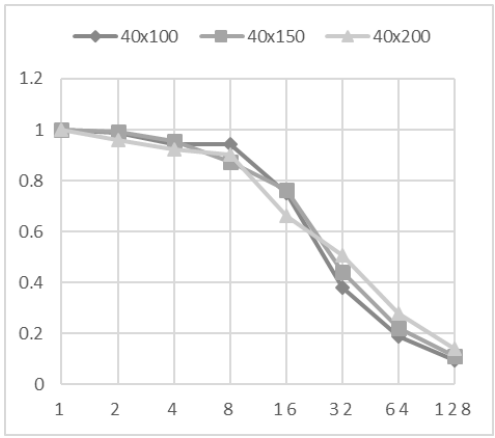
$m\times n \setminus p$	1	2	4	8	16	32	64	128
30×100	3,95	2,03	1,05	0,59	0,37	0,32	0,32	0,32
30×150	39,1	20,0	10,4	5,21	3,46	2,32	2,33	2,32
30×200	231	116	61,5	32,2	18,8	13,8	13,8	13,8
40×100	11,5	5,83	3,05	1,53	0,96	0,95	0,95	0,95
40×150	133	67,1	34,8	19,1	10,9	9,44	9,43	9,43
40×200	654	328	177	90,5	61,8	40,4	36,8	36,8

стрируется на матрицах размера $m\times n$, где $m \in \{30, 40\}$ и $n \in \{100, 150, 200\}$. Значение параметра r полагалось равным $m/2$.

В табл. 5 приведено время работы $T(p)$ параллельного алгоритма дуализации, основанного на S-схеме, при различных m, n и p. На рис. 3 приведены графики $S(p)$ и $E(p)$. На рис. 4 приведена столбчатая диаграмма для достигнутых уровней загрузки $s_k(p), k \in J_p$ при $p = 16$ и 32.



(*а*) График $S(p)$ при $m = 40$



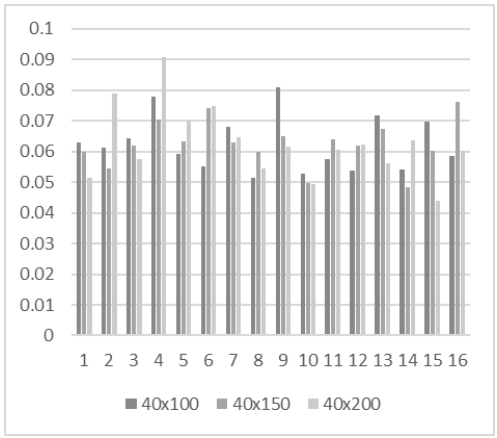
(*б*) График $E(p)$ при $m = 40$

Рис. 3 Сильная масштабируемость S-схемы

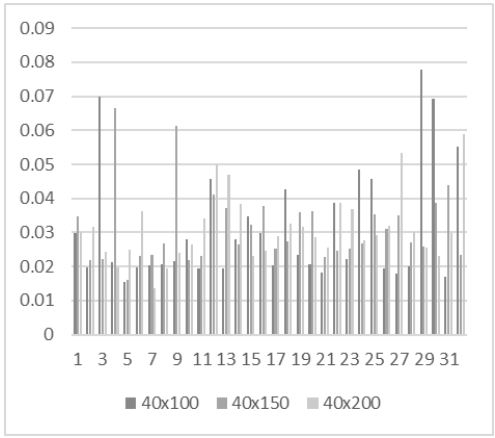
S-схема демонстрирует практически линейное ускорение и высокий уровень загрузки при $p \leqslant p^*$. Например, $p^* = 32$ при $m = 40$ и $n = 200$. Согласно рис. 4 для матрицы 40×200 достигнутый уровень загрузки при $p = 32$ для некоторых процессоров значительно превышает среднее значение этого показателя, что может быть результатом недостаточного качества оценки $f_r^*(j)$ или неоптимальности построенного жадным алгоритмом расписания.

6 Заключение

В данной работе развит предложенный в [13] подход к построению параллельных алгоритмов для дискретных перечислительных задач. Подход основан на статистических оценках объемов вычислительных подзадач. Распределение вычислительных подзадач осуществляется согласно заранее составленному расписанию. Для составления указанного



(*а*) График $s_k(16)$ при $m = 40$



(*б*) График $s_k(32)$ при $m = 40$

Рис. 4 Достигнутый уровень загрузки в S-схеме

расписания определяется вид распределения случайной величины, использующейся для оценки объемов подзадач, и оптимизируется уровень загрузки процессоров. В рамках рассматриваемого подхода разработана новая менее трудоемкая схема распараллеливания асимптотически оптимальных алгоритмов дуализации.

Работа схемы продемонстрирована на примере распараллеливания алгоритма RUNC-M [10], который в настоящее время является лидером по скорости счета среди алгоритмов дуализации. Предлагаемый подход к построению параллельных алгоритмов дуализации обеспечивает высокую точность оценок для объемов подзадач, что при определенных условиях приводит и к высоким показателям эффективности параллельного алгоритма. Однако рассматриваемый подход не эффективен при большом числе процессоров, поскольку вычислительные подзадачи имеют существенно разные размеры (распараллеливание происходит на первом ярусе дерева решений, которое строит асимптотически оптимальный алгоритм дуализации).

Литература

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

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

[3] *Khachiyan L., Fredman M.* On the complexity of dualization of monotone disjunctive normal forms // J. Algorithms, 1996. Vol. 21, No. 3. P. 618–628.

[4] *Дюкова Е. В.* Асимптотически оптимальные тестовые алгоритмы в задачах распознавания // Пробл. кибернетики, 1982. Т. 1. № 39. С. 165–199.

[5] *Djukova E. V., Zhuravlev Y. I.* Discrete methods of information analysis in recognition and algorithm synthesis // Pattern Recogn. Image Anal., 1997. Vol. 7. No. 2. P. 192–207.

[6] *Дюкова Е. В., Журавлев Ю. И.* Дискретный анализ признаковых описаний в задачах распознавания большой размерности // Ж. вычисл. матем. и матем. физ., 2000. Т. 40. № 8. С. 1264–1278.

[7] *Дюкова Е. В.* О сложности реализации дискретных (логических) процедур распознавания // Ж. вычисл. матем. и матем. физ., 2004. Т. 44. № 3. С. 551–561.

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

[9] *Murakami K., Uno T.* Efficient algorithms for dualizing large-scale hypergraphs // Discrete Appl. Math., 2014. Vol. 170. P. 83–94.

[10] *Дюкова Е. В., Прокофьев П. А.* Построение и исследование новых асимптотически оптимальных алгоритмов дуализации // Машинное обучение и анализ данных, 2014. Т. 1. № 8. С. 1048–1067.

[11] *Khachiyan L., Boros E., Elbassioni K., Gurvich V.* A new algorithm for the hypergraph transversal proble // Computing and combinatorics / Ed. L. Wang. — Lecture notes in computer science ser. — Springer, 2005. Vol. 3595. P. 767–776.

[12] *Khachiyan L., Boros E., Gurvich V., Elbassioni K.* Computing many maximal independent sets for hypergraphs in parallel // Parallel Processing Lett., 2007. Vol. 17, No. 2. P. 141–152.

[13] *Дюкова Е. В., Никифоров А. Г., Прокофьев П. А.* Статистически эффективная схема распараллеливания алгоритмов дуализации // Машинное обучение и анализ данных, 2014. Т. 1. № 7. С. 846–853.

Поступила в редакцию 16.06.2015

References

[1] Djukova, E. V. 1977. On an asympotically optimal algorithm for constructing irredundant tests. *Dokl. Akad. Nauk SSSR* 223(4):527–530.

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

[3] Khachiyan L., and M. Fredman. 1996. On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms* 21(3):618–628.

[4] Djukova, E. V. 1982. Asimptoticheski optimal'nye testovye algoritmy v zadachakh raspoznavaniya. *Problemy Kibernetiki* 1(39):165–199.

[5] Djukova, E. V., and Y. I. Zhuravlev. 1997. Discrete methods of information analysis in recognition and algorithm synthesis. *Pattern Recogn. Image Anal.* 7(2):192–207.

[6] Djukova, E. V., and Y. I. Zhuravlev. 2000. Diskretnyy analiz priznakovykh opisaniy v zadachakh raspoznavaniya bol'shoy razmernosti. *J. Vychisl. Matem. i Matem. Fiz.* 40(8):1264–1278.

[7] Djukova, E. V. O slozhnosti realizatsii diskretnykh (logicheskikh) protsedur raspoznavaniya. *J. Vychisl. Matem. i Matem. Fiz.* 44(3):551–561.

[8] Djukova, E. V., and A. S. Inyakin. 2008. Asymptoticheski optimal'noe postroenie tupikovykh pokrytiy tselochislennoy matritsy. *Matematicheskie Vorposy Kibernetiki* 17:235–246.

[9] Murakami, K., and T. Uno. 2014. Efficient algorithms for dualizing large-scale hypergraphs. *Discrete Appl. Math.* 170:83–94.

[10] Djukova, E. V., and P. A. Prokofyev. 2014. Construction and investigation of new asymptotically optimal algorithms for dualization. *Machine Learning Data Anal.* 1(8):1048–1067.

[11] Khachiyan, L., E. Boros, K. Elbassioni, and V. Gurvich. 2005. A new algorithm for the hypergraph transversal proble. *Computing and combinatorics*. Ed. L. Wang. Lecture notes in computer science ser. Springer. 3595:767–776.

[12] Khachiyan, L., E. Boros, V. Gurvich, and K. Elbassioni. 2007. Computing many maximal independent sets for hypergraphs in parallel. *Parallel Processing Lett.* 17(2):141–152.

[13] Djukova, E. V., A. G. Nikiforov, and P. A. Prokofyev. 2014. Statistically efficient parallel scheme for dualization algorithms. *Machine Learning Data Anal.* 1(7):846–853.

Received June 16, 2015