ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/23/11

УДК 519.1, 519.8

ОБ АСИМПТОТИЧЕСКИ ОПТИМАЛЬНОМ ПЕРЕЧИСЛЕНИИ НЕПРИВОДИМЫХ ПОКРЫТИЙ БУЛЕВОЙ МАТРИЦЫ[1]

Е. В. Дюкова*,**, П. А. Прокофьев*

*Вычислительный центр им. А. А. Дородницына, г. Москва, Россия

**Московский государственный университет им. М. В. Ломоносова, г. Москва, Россия

E-mail: edjukova@mail.ru, p_prok@mail.ru

Рассматривается задача перечисления неприводимых покрытий булевой матрицы, известная как задача дуализации. Эффективность алгоритма дуализации принято оценивать сложностью шага алгоритма (построения очередного неприводимого покрытия). Рассматривается подход к построению алгоритма дуализации, эффективного в «типичном случае». Подход основан на понятии асимптотически оптимального алгоритма с полиномиальной задержкой. Строятся две модификации предложенного ранее алгоритма АО2, позволяющие существенно сократить временные затраты в экспериментах со случайными булевыми матрицами.

Ключевые слова: *булева матрица, перечисление неприводимых покрытий, асимптотически оптимальный алгоритм дуализации, нормальная форма монотонной булевой функции, минимальное вершинное покрытие гиперграфа.*

Введение

Пусть $L = \|a_{ij}\|_{m\times n}$ — булева матрица и H — набор столбцов матрицы L. Набор H называется *покрытием* матрицы L, если каждая строка матрицы L в пересечении хотя бы с одним столбцом из H даёт 1. Покрытие H называется *неприводимым*, если любое собственное подмножество H не является покрытием L.

Через $\mathcal{P}(L)$ обозначается множество всех неприводимых покрытий матрицы L. Рассматривается задача перечисления элементов множества $\mathcal{P}(L)$, называемая *дуализацией*.

Задача дуализации может быть также переформулирована с использованием понятий теории булевых функций и теории графов и гиперграфов. Приведём эти формулировки:

1) Дана конъюнктивная нормальная форма из m элементарных дизъюнкций, реализующая монотонную булеву функцию F от n переменных. Требуется построить (перечислить) все максимальные конъюнкции функции F.
2) Дан гиперграф G с n вершинами и m ребрами. Требуется найти все минимальные вершинные покрытия гиперграфа G.

Двойственной к 2 является задача поиска максимальных независимых подмножеств вершин гиперграфа G.

[1]Работа частично поддержана грантом РФФИ №13-01-00787-а и грантом президента РФ НШ-4652.2012.1.

В большинстве случаев мощность множества $\mathcal{P}(L)$ экспоненциально растёт с ростом размера матрицы L. Эффективность алгоритмов перечисления неприводимых покрытий принято оценивать сложностью получения очередного решения (сложностью шага алгоритма) [1]. Различают алгоритмы, эффективные в «худшем случае», и алгоритмы, эффективные в «типичном случае». К алгоритмам, эффективным в худшем случае, относят, в частности, алгоритмы с (*квази*)*полиномиальными задержками* [1, 2] и *инкрементальные* (*квази*)*полиномиальные* алгоритмы [3 – 6].

Говорят, что алгоритм работает с (квази)полиномиальной задержкой, если каждый его шаг (построение очередного неприводимого покрытия) осуществляется за (квази)полиномиальное время от размера задачи. В общем случае алгоритм, находящий на каждом шаге неприводимое покрытие за (квази)полиномиальное время, до сих пор не построен, и неизвестно, существует ли он. Для некоторых частных случаев построены алгоритмы с (квази)полиномиальной задержкой. Например, в [1] с полиномиальной задержкой перечисляются максимальные независимые наборы вершин графа, что соответствует случаю, когда матрица L в каждой строке имеет не более двух единичных элементов.

Для построения алгоритмов, эффективных в типичном случае, используется подход, основанный на понятии *асимптотически оптимального алгоритма с полиномиальной задержкой*, который впервые предложен в [7] и развит в последующих работах [8 – 10]. В отличие от «точного» алгоритма с полиномиальной задержкой, асимптотически оптимальному алгоритму разрешено делать «лишние» полиномиальные шаги, но число таких шагов для почти всех матриц размера $m \times n$ должно иметь более низкий порядок роста, чем число всех шагов алгоритма, с ростом m и n.

К настоящему моменту для случая, когда $\log m \leqslant (1-\varepsilon)\log n$, при $0 < \varepsilon < 1$, построен ряд асимптотически оптимальных алгоритмов дуализации [7 – 9]. В этих алгоритмах встречаются лишние шаги двух типов: 1) построение набора столбцов, не являющегося покрытием, и 2) нахождение неприводимого покрытия, которое было построено на предыдущих шагах. Среди построенных алгоритмов есть такие, в которых присутствуют лишние шаги обоих типов, например алгоритм АО1 [7], и есть алгоритмы с лишними шагами только одного вида, например алгоритм АО2 [8]. Обзор в области синтеза асимптотически оптимальных алгоритмов перечисления неприводимых покрытий можно найти в [9].

В настоящей работе получены следующие результаты. Описана схема построения алгоритма типа АО2. Приведено описание алгоритма АО2 в рамках предложенной схемы. Показано, что для ответа на вопрос, является ли очередной шаг алгоритма типа АО2 лишним, необходимо решить NP-полную задачу. Предложены две модификации алгоритма АО2, названные АО2К и АО2М. Проведено тестирование алгоритмов АО2, АО2К и АО2М на случайных булевых матрицах.

1. Основные понятия и обозначения

Введём обозначение $E(L) = \{(i,j) : a_{ij} = 1, i \in \{1,\ldots,m\}, j \in \{1,\ldots,n\}\}$.

Два элемента (i,j) и (t,l) из $E(L)$ называются *совместимыми*, если $a_{il} = 0$, $a_{tj} = 0$. Набор $Q \subseteq E(L)$ называется *совместимым*, если любые два различных элемента (i,j) и (t,l) из Q совместимы. Пусть $B \subseteq E(L)$. Через $H(B)$ обозначается множество столбцов с номерами из множества $\{j : \exists i\,((i,j) \in B)\}$.

Строка с номером i матрицы L называется *покрытой* совместимым набором Q, если эта строка в пересечении с хотя бы одним столбцом из $H(Q)$ даёт 1. Совместимый

набор Q называется *покрывающим*, если все строки L покрыты Q. Очевидно следующее утверждение.

Утверждение 1. Набор столбцов H является неприводимым покрытием тогда и только тогда, когда найдётся покрывающий набор Q, для которого $H(Q) = H$.

Следовательно, перечисляя все покрывающие наборы, можно построить множество $\mathcal{P}(L)$.

Покрывающий набор $Q = \{(i_1, j_1), \ldots, (i_r, j_r)\}$ называется *верхним*, если для любого покрывающего набора $Q' = \{(t_1, j_1), \ldots, (t_r, j_r)\}$ верны неравенства $t_u \geqslant i_u$, $u \in \{1, \ldots, r\}$. Очевидно следующее утверждение.

Утверждение 2. Для любого неприводимого покрытия H существует единственный верхний набор Q, такой, что $H(Q) = H$.

Таким образом, задача перечисления неприводимых покрытий матрицы L может быть сведена к перечислению верхних наборов для матрицы L.

Алгоритм АО2 на каждом шаге строит очередной покрывающий набор Q. В случае, когда Q является верхним, происходит пополнение множества найденных неприводимых покрытий матрицы L набором столбцов $H(Q)$. В противном случае шаг считается лишним. Будем говорить, что алгоритм A имеет *тип АО2*, если A для поиска неприводимых покрытий перечисляет покрывающие наборы и выбирает среди них верхние наборы.

2. Общая схема построения алгоритмов типа АО2

Совместимый набор Q называется *правильным относительно* $B \subseteq E(L)$, если существует покрывающий набор $Q' \supseteq Q$, такой, что $Q' \setminus Q \subseteq B$.

Элемент $(i, j) \in B$ называется *запрещённым для* Q *относительно* B, если набор $Q' = Q \cup \{(i, j)\}$ либо не является совместимым, либо не является правильным относительно B. Обозначим через $\Delta(B, Q)$ множество запрещённых для Q относительно B элементов.

Конечная последовательность пар $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$ называется *траекторией*, если:

1) набор Q_t, $1 \leqslant t \leqslant l$, является правильным относительно $B_t \subseteq E(L)$;

2.1) либо B_t, $1 < t \leqslant l$, получается удалением одного или нескольких элементов из B_{t-1}, а Q_{t-1} не меняется, то есть $(B_t, Q_t) = (B_{t-1} \setminus A_t, Q_{t-1})$ для некоторого $A_t \neq \varnothing$, $A_t \subseteq B_{t-1}$;

2.2) либо Q_t, $1 < t \leqslant l$, получается добавлением к Q_{t-1} некоторого незапрещённого для Q_{t-1} относительно B_{t-1} элемента (i_t, j_t), а набор B_t получается удалением из B_{t-1} элемента (i_t, j_t), то есть $(B_t, Q_t) = (B_{t-1} \setminus \{(i_t, j_t)\}, Q_{t-1} \cup \{(i_t, j_t)\})$, где элемент $(i_t, j_t) \in B_{t-1} \setminus \Delta(B_{t-1}, Q_{t-1})$.

Траектория $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$ называется *законченной*, если $B_l = \varnothing$. При этом набор Q_l является покрывающим и называется *результатом* законченной траектории α.

Ясно, что для любого покрывающего набора Q можно построить хотя бы одну законченную траекторию, результат которой равен Q. То есть, перечисляя всевозможные законченные траектории, можно найти все покрывающие наборы. Однако число траекторий, как правило, значительно больше числа покрывающих наборов. Поэтому желательно исключить перечисление траекторий, приводящих к одним и тем же покрывающим наборам, но при этом не пропустить ни одного покрывающего набора.

Пусть последовательность $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$ является траекторией. Последовательность $\alpha_t^* = \{(B_1, Q_1), \ldots, (B_{t-1}, Q_{t-1}), (B_t, Q_{t-1})\}$, $t \in \{2, \ldots, l-1\}$, являющаяся траекторией, но не совпадающая с началом α, называется *альтернативной* траекторией для α. Последовательность α_t^* альтернативна для α в том и только в том случае, когда набор Q_{t-1} является правильным относительно B_t, и $Q_{t-1} \neq Q_t$.

Для α может не существовать ни одной альтернативной траектории. В случае, когда есть хотя бы одна альтернативная траектория для α, самая длинная из них называется *соседней* траекторией для α. Легко доказывается следующее полезное свойство соседних траекторий.

Утверждение 3. Пусть $\alpha_1, \ldots, \alpha_r$ — упорядоченное семейство законченных траекторий, такое, что началом траектории α_u, $u \in \{2, \ldots, r\}$, является соседняя траектория для α_{u-1}. Тогда результаты всех траекторий семейства различны.

Законченная траектория $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$ называется *реализацией*, если для всех $t \in \{2, \ldots, l\}$ либо $Q_{t-1} \neq Q_t$, либо $(B_{t-1} \setminus B_t) \subseteq \Delta(B_{t-1}, Q_{t-1})$.

Алгоритм типа АО2 (общая схема)

Вход: L — булева матрица.

На шаге 1 проверить, является ли набор $\varnothing$ правильным относительно $E(L)$. Если не является, то закончить работу и вернуть результат $\mathcal{P}(L) = \varnothing$. В противном случае построить реализацию α_1, начинающуюся с $(E(L), \varnothing)$. В случае, когда результат реализации α_1 — набор $Q^{(1)}$ — является верхним, пополнить множество решений неприводимым покрытием $H(Q^{(1)})$. В противном случае множество решений не пополняется. Перейти к шагу 2.

Пусть на шаге t построена траектория α_t.

На шаге $t+1$, $t \geqslant 1$, проверить, существует ли соседняя траектория для α_t. Если не существует, то закончить работу алгоритма. В противном случае взять соседнюю для α_t траекторию α^*. Построить реализацию α', первый элемент которой совпадает с последним элементом α^*. Удалить из α^* последний элемент и добавить к ней элементы реализации α'. Полученная траектория α_{t+1} является законченной. В случае, когда результат реализации α_{t+1} — набор $Q^{(t+1)}$ — является верхним, пополнить множество решений неприводимым покрытием $H(Q^{(t+1)})$. В противном случае множество решений не пополняется. Перейти к шагу $t+2$.

Для обоснования корректности алгоритма, построенного по описанной схеме, достаточно показать, что каждый покрывающий набор строится на каком-либо шаге алгоритма и при этом один и тот же набор не строится на разных шагах.

Заметим, что реализация отличается от законченной траектории тем, что если при переходе от (B_{t-1}, Q_{t-1}) к (B_t, Q_t) набор Q_{t-1} не пополняется, то из B_{t-1} удаляются только запрещённые для Q_{t-1} относительно B_{t-1} элементы. Это свойство позволяет гарантировать, что ни один покрывающий набор не будет пропущен алгоритмом. Отсутствие повторов следует из утверждения 3.

Число шагов алгоритма равно числу покрывающих наборов матрицы L. Сложность шага складывается из сложности поиска соседней траектории и сложности построения реализации с заданным началом. Согласно [8], алгоритм, решающий указанные подзадачи на каждом шаге за полиномиальное время, является асимптотически оптимальным с полиномиальной задержкой для случая, когда размер входа удовлетворяет ограничению $\log m \leqslant (1-\varepsilon)\log n$, $0 < \varepsilon < 1$.

Приведём описание асимптотически оптимального алгоритма АО2 в рамках предложенной схемы.

3. Алгоритм АО2

Пусть $B \subseteq E(L)$ и $(i,j) \in E(L)$. Обозначим через $\Delta_{ij}(B)$ набор элементов из B, не совместимых с (i,j). Очевидно, если $(i,j) \in Q$, то $\Delta_{ij}(B) \subseteq \Delta(B,Q)$.

Пусть H — набор столбцов матрицы L. Строка с номером i называется *охватывающей* для строки с номером t относительно H, если для любого столбца из H с номером j выполняется неравенство $a_{ij} \geqslant a_{tj}$.

Пусть $B \subseteq E(L)$ и пусть не покрытая совместимым набором Q строка с номером i охватывает не покрытую Q строку с номером t относительно $H(B)$. Элемент $(i,j) \in B$, для которого $a_{tj} = 0$, является запрещённым для Q относительно B. Обозначим множество таких элементов через $\Delta^*(B,Q)$.

Процедура построения реализации в алгоритме АО2

Вход: (B_1, Q_1) — начало траектории.

Выполнить в цикле следующие шаги.

Ш а г 1. Пусть построена траектория длины $t \geqslant 1$. Если $B_t = \varnothing$, то *закончить цикл.*

Ш а г 2. Если $\Delta^*(B_t, Q_t) \neq \varnothing$, то построить $(B_{t+1}, Q_{t+1}) = (B_t \setminus \Delta^*(B_t, Q_t), Q_t)$ и *продолжить цикл.*

Ш а г 3. Взять элемент (i,j) из B_t с наименьшим порядковым номером $m(j-1)+i$. Построить пару $(B_{t+1}, Q_{t+1}) = (B_t \setminus \{(i,j)\}, Q_t \cup \{(i,j)\})$.

Ш а г 4. Если $\Delta_{ij}(B_{t+1}) \neq \varnothing$, то построить пару $(B_{t+2}, Q_{t+2}) = (B_{t+1} \setminus \Delta_{ij}(B_{t+1}), Q_{t+1})$. *Продолжить цикл.*

Процедура поиска соседней траектории в алгоритме АО2

Вход: $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$ — траектория.

Выполнить в цикле следующие шаги.

Ш а г 1. Пусть текущая длина α равна t. Если $t \leqslant 1$, то вернуть ответ, что соседней траектории не существует.

Ш а г 2. Если $t > 1$ и $Q_t = Q_{t-1}$, то удалить из α последний элемент и *продолжить цикл.*

Ш а г 3. Проверить, что для любой не покрытой Q_{t-1} строки с номером i есть хотя бы один элемент $(i,j) \in B_t$. Если условие выполняется, то заменить в α последний элемент на пару (B_t, Q_{t-1}) и вернуть α. В противном случае удалить из α последний элемент и *продолжить цикл.*

Проведём анализ временной сложности шага алгоритма АО2. При поиске элементов множества $\Delta_{ij}(B)$ максимальное число просматриваемых элементов матрицы не превосходит mn. При поиске элементов множества $\Delta^*(B,Q)$ максимальное число просматриваемых элементов матрицы не превосходит m^2n (поиск охватывающих строк). Длина любой траектории не превосходит $3q$, где $q = \min\{m,n\}$. Поэтому время работы процедуры построения реализации равно $\mathrm{O}(qm^2n)$. Время работы процедуры поиска соседней траектории равно $\mathrm{O}(qmn)$. Проверка того, что покрывающий набор Q является верхним, осуществляется за $\mathrm{O}(mn)$. Итак, временная сложность шага алгоритма АО2 составляет $\mathrm{O}(qm^2n)$.

Проведём анализ затрат памяти алгоритма АО2. Прежде всего отметим, что алгоритм АО2 так же, как любой алгоритм, построенный по описанной выше схеме, относится к типу «построил и забыл». Это означает, что для работы алгоритма нет необходимости хранить все найденные ранее покрывающие наборы или неприводимые покрытия. Каждая траектория $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$ обладает свойством

$B_1 \supset \ldots \supset B_l$ и $Q_1 \subseteq \ldots \subseteq Q_l$. Поэтому α может быть закодирована в одной целочисленной матрице $D_\alpha = \|d_{ij}\|_{m\times n}$, элементы которой определяются следующим образом: $d_{ij} = 0$, $(i,j) \notin E(L)$; $d_{ij} = 0$, $(i,j) \in B_l$; $d_{ij} = -t$, $(i,j) \in Q_{t+1} \setminus Q_t$; $d_{ij} = t$, $(i,j) \in B_t \setminus B_{t+1}$. При таком представлении затраты памяти составляют $\mathrm{O}(mn \log q)$ бит.

4. О перечислении верхних наборов

Строка матрицы L с номером t называется *конкурентной* для совместимого набора Q, если существует совместимый набор Q', такой, что $H(Q) = H(Q')$, и для $(t,j) \in Q'$ и $(i,j) \in Q$ выполняется неравенство $i > t$. Ясно, что верхний набор Q не может иметь конкурентных строк.

Правильный относительно B набор Q называется *верным относительно* B, если существует верхний набор $Q' \supseteq Q$, такой, что $Q' \setminus Q \subseteq B$.

Привлекательной является идея строить при поиске покрывающего набора такую траекторию $\alpha = \{(B_1, Q_1), \ldots, (B_l, Q_l)\}$, что набор Q_t, $t \in \{1, \ldots, l\}$, является верным относительно B_t. Тогда при $B_l = \varnothing$ набор Q_l является верхним. Алгоритм, строящий такую траекторию за полиномиальное время от m и n, возвращал бы на каждом шаге очередное неприводимое покрытие с полиномиальной задержкой. Это эквивалентно тому, что исходя из траектории, построенной на текущем шаге, можно за полиномиальное время определить, является ли очередной шаг алгоритма лишним. Однако справедлива следующая теорема.

Теорема 1. Задача проверки того, является ли набор $Q \neq \varnothing$ верным относительно $E(L)$, NP-полна.

Доказательство. Пусть булева функция $f(x_1, \ldots, x_N)$ задана конъюнктивной нормальной формой (КНФ) $C_1 \wedge \ldots \wedge C_M$, где $C_1, \ldots, C_M$ — элементарные дизъюнкции от переменных $x_1, \ldots, x_N$.

Построим граф $G = (V, E)$, где множество V состоит из вершин $1, \varnothing, C_1, \ldots, C_M$, $x_1, \ldots, x_N, \bar{x}_1, \ldots, \bar{x}_N$, и множество E состоит из рёбер четырёх типов:

1) рёбра для конъюнкций $\{1, C_1\}, \ldots, \{1, C_M\}$;
2) вспомогательное ребро $\{1, \varnothing\}$;
3) рёбра для переменных $\{x_1, \bar{x}_1\}, \ldots, \{x_N, \bar{x}_N\}$;
4) рёбра для вхождений переменных в конъюнкции: $\{C_i, x_j\}$, если x_j входит в C_i, и $\{C_i, \bar{x}_j\}$, если $\bar{x}_j$ входит в C_i.

Пусть L — матрица инцидентности графа G и столбцы с номерами $1, \ldots, M+2$ соответствуют вершинам $1, \varnothing, C_1, \ldots, C_M$, а строки с номерами $1, \ldots, M, M+1$ — рёбрам $\{1, C_1\}, \ldots, \{1, C_M\}, \{1, \varnothing\}$.

Набор $Q = \{(M+1, 1)\}$ является правильным относительно $E(L)$. Требуется определить, является ли Q верным относительно $E(L)$.

Набор Q покрывает строки с номерами $1, \ldots, M, M+1$, при этом строки с номерами $1, \ldots, M$ являются конкурентными. Поэтому если верхний набор Q' содержит Q, то в $H(Q')$ должны входить столбцы с номерами $3, \ldots, 2+M$, чтобы строки с номерами $1, \ldots, M$ не были конкурентными для Q'. Из этого следует, что $Q' \supset \{(i_1, 3), \ldots, (i_M, 2+M)\}$, где для любого $l \in \{1, \ldots, M\}$ строка с номером i_l соответствует либо ребру $\{C_l, x_{s_l}\}$, либо ребру $\{C_l, \bar{x}_{s_l}\}$.

При этом Q' покрывает соответствующие рёбрам $\{x_1, \bar{x}_1\}, \ldots, \{x_N, \bar{x}_N\}$ строки тогда и только тогда, когда среди номеров строк $i_1, \ldots, i_M$ нет двух таких i_u, i_v, что строка с номером i_u соответствует ребру $\{C_u, x_{s_u}\}$, а строка с номером i_v — ребру $\{C_v, \bar{x}_{s_v}\}$, и $s_u = s_v$.

Таким образом, существование верхнего набора $Q' \supset Q$ равносильно существованию набора значений переменных $x_1, \ldots, x_N$, для которого истинны все элементарные дизъюнкции $C_1, \ldots, C_M$, а следовательно, выполнима функция $f(x_1, \ldots, x_N)$.

В результате задача проверки выполнимости функции, заданной КНФ, полиномиально сведена к задаче проверки верности правильного набора Q относительно $E(L)$. Как известно, задача проверки выполнимости КНФ NP-полна. ■

Обозначим $E_t(L)$, $t \in \{1, \ldots, m\}$, множество элементов $\{(i,j) \in E(L) : a_{ij} = a_{tj} = 1\}$.

Повысить эффективность алгоритма АО2 позволяет осуществление при построении реализации проверки выполнения следующего необходимого условия верности набора.

Утверждение 4. Пусть Q — верный относительно $B \subseteq E(L)$ набор. Тогда для каждой не покрытой Q или конкурентной для Q строки с номером t выполняется $E_t(L) \cap B \neq \varnothing$.

Теперь, если в процедуре построения реализации в алгоритме АО2 всякий раз останавливаться, как только в траекторию была добавлена пара (B_t, Q_t), не удовлетворяющая необходимому условию утверждения 4, то, во-первых, часть траекторий будет короче, а во-вторых, число шагов алгоритма будет меньше. Эксперименты показали, что, как правило, снижение числа шагов, а следовательно, и времени выполнения алгоритма довольно существенно.

5. Модификации алгоритма АО2

Опишем две модификации алгоритма АО2, названные АО2К и АО2М. Алгоритмы АО2К и АО2М полностью вписываются в общую схему построения алгоритма типа АО2. Отличия от АО2 появляются только в процедуре построения реализации.

Процедура построения реализации в алгоритме АО2К

Вход: (B_1, Q_1) — начало траектории.

Выполнить в цикле следующие шаги.

Ш а г 1. Пусть построена траектория длины $t \geqslant 1$. Если $B_t = \varnothing$, то *закончить цикл.*

Ш а г 2. Если пара (B_t, Q_t) не удовлетворяет необходимому условию утверждения 4, то *закончить цикл.*

Ш а г 3. Если $\Delta^*(B_t, Q_t) \neq \varnothing$, то построить $(B_{t+1}, Q_{t+1}) = (B_t \setminus \Delta^*(B_t, Q_t), Q_t)$ и *продолжить цикл.*

Ш а г 4. Если у набора Q_t есть хотя бы одна конкурентная строка, то найти конкурентную строку с наименьшим номером r и взять элемент (i,j) из $E_r(L) \cap B_t$ с наименьшим порядковым номером $m(j-1)+i$. В противном случае взять элемент (i,j) из B_t с наименьшим порядковым номером $m(j-1)+i$. Построить пару $(B_{t+1}, Q_{t+1}) = (B_t \setminus \{(i,j)\}, Q_t \cup \{(i,j)\})$.

Ш а г 5. Если $\Delta_{ij}(B_{t+1}) \neq \varnothing$, то построить $(B_{t+2}, Q_{t+2}) = (B_{t+1} \setminus \Delta_{ij}(B_{t+1}), Q_{t+1})$. *Продолжить цикл.*

Предположим, что на вход процедуре построения реализации в алгоритме АО2К поступает набор Q_1, не являющийся верным относительно B_1. Чем раньше при построении реализации предпринимаются попытки покрыть конкурентные строки, тем быстрее обнаруживается, что набор Q_1 не является верным относительно B_1. Стратегия алгоритма АО2К характеризуется как «борьба с конкурентными строками».

Процедура построения реализации в алгоритме АО2М

Вход: (B_1, Q_1) — начало траектории.

Выполнить в цикле следующие шаги.

Ш а г и 1–3 и 5 не отличаются от соответствующих шагов процедуры построения реализации в алгоритме АО2К.

Ш а г 4. Найти номер r не покрытой Q_t или конкурентной для Q_t строки с наименьшим числом элементов в пересечении $E_r(L) \cap B_t$. Взять элемент (i, j) из $E_r(L) \cap B_t$ с наименьшим порядковым номером $m(j-1)+i$. Построить $(B_{t+1}, Q_{t+1}) = (B_t \setminus \{(i,j)\}, Q_t \cup \{(i,j)\})$.

Набор $Q_{t+1} = Q_t \cup \{(i,j)\}$ покрывает строку с номером r, только если $(i,j) \in E_r(L) \cap B_t$. На качественном уровне понятно, что чем меньше множество $E_r(L) \cap B_t$, тем «сложнее» строка с номером r для покрытия наборами строящейся реализации. Стратегия алгоритма АО2М характеризуется как «от сложного к простому». Оправдана эта стратегия тем, что результат реализации должен покрывать все, в том числе и «сложные» строки, а покрыть «простые» строки, вероятнее всего, получится и позднее, даже тогда, когда в множестве B_t будет меньше элементов.

В количественном отношении выигрыш алгоритма АО2М по сравнению с алгоритмами АО2 и АО2К проявляется следующим образом. Представим естественным образом каждую траекторию в виде простой цепи. Объединим все простые цепи траекторий, построенных алгоритмом, в один граф. Полученный граф, очевидно, является деревом и далее называется деревом решений алгоритма. Эксперименты показали, что число вершин и рёбер дерева решений, строящегося алгоритмом АО2М при дуализации матрицы L, как правило, меньше, чем соответственно число вершин и рёбер дерева решений, строящегося алгоритмом АО2 или АО2К при дуализации той же матрицы L.

6. Результаты экспериментов

Проведено тестирование алгоритмов АО2, АО2К и АО2М на случайных матрицах, для которых выполняется $50 \leqslant mn \leqslant 500$. Для каждой матрицы и каждого алгоритма измерялись:

1) T — время перечисления всех неприводимых покрытий;
2) $\delta_{\max}$ — максимальное время получения очередного неприводимого покрытия;
3) N — число узлов дерева решений;
4) N_0 — число лишних шагов алгоритма.

Результаты счёта приведены на рис. 1–6.

На рис. 1 видно, что алгоритм АО2М работает быстрее алгоритмов АО2 и АО2К. В среднем скорость АО2М в 4 раза больше скорости АО2. По показателю $\delta_{\max}$ алгоритм АО2М также лидирует (рис. 3 и 4). Скорость работы АО2К сравнима со скоростью АО2.

Анализ числа лишних шагов показывает, что алгоритмы АО2К и АО2М почти всегда делают меньше лишних шагов, чем алгоритм АО2 (рис. 2). Однако наименьшее значение этого показателя у модификации АО2К.

Заметим, что зависимости полного времени дуализации от числа узлов дерева решений у алгоритмов АО, АО2К и АО2М имеют довольно схожий характер (рис. 6). Число узлов дерева решений при росте числа неприводимых покрытий медленней всего увеличивается у алгоритма АО2М (рис. 5). По-видимому, число узлов дерева решений в большей степени влияет на время работы алгоритма, нежели число лишних шагов алгоритма.

Полученные выводы разнятся с первоначальными представлениями о факторах, влияющих на производительность алгоритма АО2. Предполагалось, что построенные модификации позволят снизить временные затраты на дуализацию за счёт уменьшения числа лишних шагов алгоритма. Однако оказалось, что значительное увеличение скорости дуализации достигается именно за счёт сокращения числа узлов дерева решений.

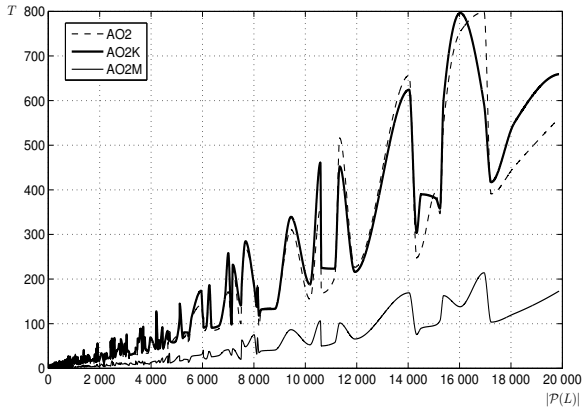

Рис. 1. Зависимость T от $|\mathcal{P}(L)|$

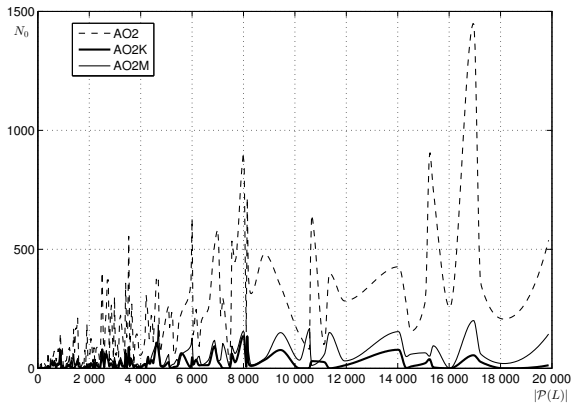

Рис. 2. Зависимость N_0 от $|\mathcal{P}(L)|$

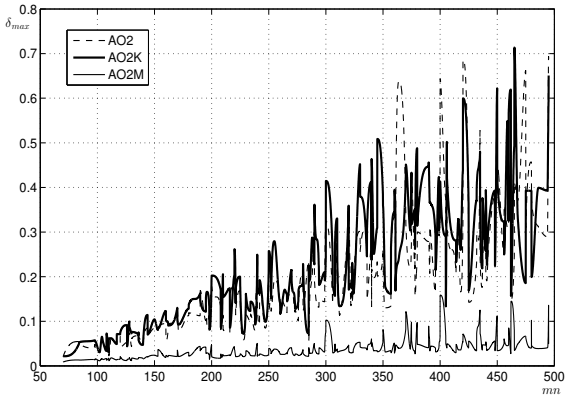

Рис. 3. Зависимость $\delta_{\max}$ от mn

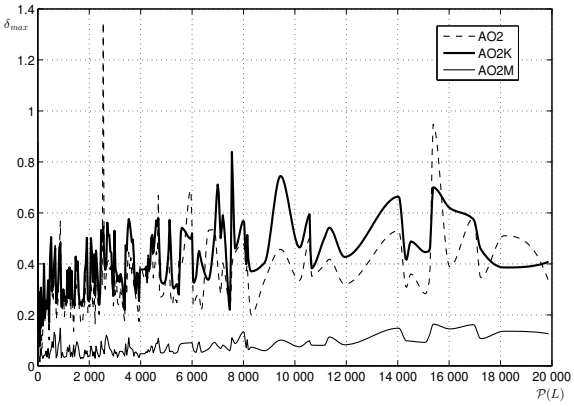

Рис. 4. Зависимость $\delta_{\max}$ от $|\mathcal{P}(L)|$

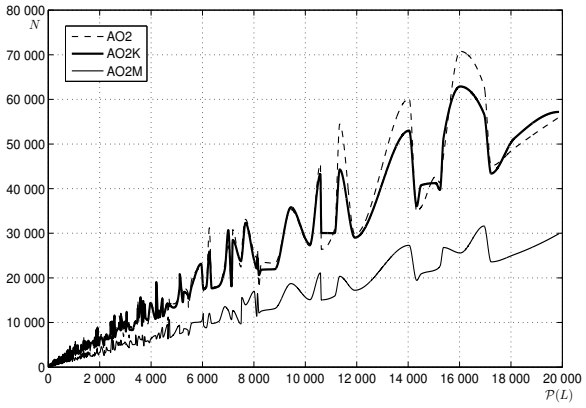

Рис. 5. Зависимость N от $|\mathcal{P}(L)|$

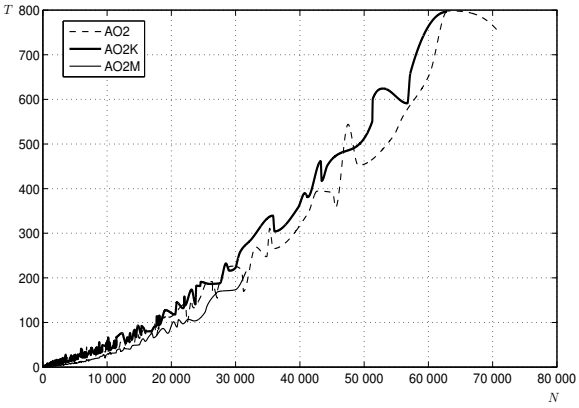

Рис. 6. Зависимость T от N

Заключение

В работе используется подход к построению алгоритмов дуализации, эффективных в «типичном случае», в основе которого лежит понятие асимптотически оптимального алгоритма с полиномиальной задержкой.

В качестве отправной точки рассматривается асимптотически оптимальный алгоритм AO2, который на каждом шаге за полиномиальное время строит неприводимое покрытие, но построенное покрытие может совпадать с найденными на предыдущих шагах. Фактически, в алгоритме AO2 перечисление неприводимых покрытий сводится к перечислению покрывающих наборов единичных элементов матрицы. Каждому неприводимому покрытию соответствует единственный верхний покрывающий набор. Шаг, результат которого не является верхним набором, считается лишним.

Покрывающий набор строится последовательным добавлением единичных элементов матрицы. Алгоритм AO2 добавляет элементы в порядке возрастания их порядковых номеров. Предложены две модификации данного алгоритма, основное отличие которых от AO2 в стратегии выбора очередного добавляемого единичного элемента. Тестирование алгоритмов AO2, AO2K и AO2M показало, что внесённые в первоначальный алгоритм изменения позволяют снизить временные затраты дуализации за счёт уменьшения числа узлов дерева решений.

ЛИТЕРАТУРА

1. *Johnson D., Yannakakis M., and Papadimitriou C.* On generating all maximal independent sets // Inform. Process. Lett. 1988. V. 27. No. 3. P. 119–123. URL: `http://www.sciencedirect.com/science/article/pii/0020019088900658`.
2. *Eiter T., Gottlob G., and Makino K.* New results on monotone dualization and generating hypergraph transversals // SIAM J. Comput. 2003. V. 32. No. 2. P. 514–537.
3. *Boros E. and Elbassioni K.* Generating maximal independent sets for hypergraphs with bounded edge-intersections // LATIN 2004: Theoretical Informatics. Berlin; Heidelberg: Springer, 2004. P. 488–498. URL: `http://www.springerlink.com/index/b9t9wx6ue9t9hvw3.pdf`.
4. *Boros E., Gurvich V., Elbassioni K., and Khachiyan L.* An efficient incremental algorithm for generating all maximal independent sets in hypergraphs of bounded dimension // Parallel Process. Lett. 2000. V. 10. No. 4. P. 253–266. URL: `http://www.worldscientific.com/doi/abs/10.1142/S0129626400000251`.
5. *Fredman M. L. and Khachiyan L.* On the complexity of dualization of monotone disjunctive normal forms // J. Algorithms. 1996. V. 21. P. 618–628.
6. *Khachiyan L., Boros E., Elbassioni K., and Gurvich V.* An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation // Discr. Appl. Math. 2006. V. 154. No. 16. P. 2350–2372. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0166218X06001910`.
7. *Дюкова Е. В.* Об асимптотически оптимальном алгоритме построения тупиковых тестов // ДАН СССР. 1977. Т. 233. № 4. С. 527–530.
8. *Дюкова Е. В.* О сложности реализации дискретных (логических) процедур распознавания // Журн. вычисл. матем. и матем. физики. 2004. Т. 44. № 3. С. 562–572.
9. *Дюкова Е. В., Инякин А. С.* Асимптотически оптимальное построение тупиковых покрытий целочисленной матрицы // Математические вопросы кибернетики. 2008. Т. 17. С. 235–246.
10. *Дюкова Е. В., Сотнезов Р. М.* О сложности задачи дуализации // Журн. вычисл. матем. и матем. физ. 2012. Т. 52. № 10. С. 1926–1935.