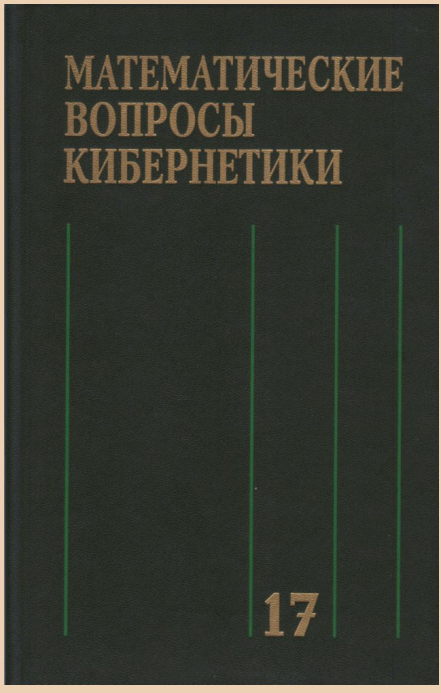
ИПМ им. М.В. Келдыша РАН • Электронная библиотека
Математические вопросы кибернетики • Выпуск 17




**Е. В. Дюкова,**
**А. С. Инякин**

# Об асимптотически оптимальном построении тупиковых покрытий целочисленной матрицы

**Рекомендуемая форма библиографической ссылки:** Дюкова Е. В., Инякин А. С. Об асимптотически оптимальном построении тупиковых покрытий целочисленной матрицы // Математические вопросы кибернетики. Вып. 17. — М.: ФИЗМАТЛИТ, 2008. — С. 247–262. URL: http://library.keldysh.ru/mvk.asp?id=2008-247

# ОБ АСИМПТОТИЧЕСКИ ОПТИМАЛЬНОМ ПОСТРОЕНИИ ТУПИКОВЫХ ПОКРЫТИЙ ЦЕЛОЧИСЛЕННОЙ МАТРИЦЫ *)

***Е. В. ДЮКОВА, А. С. ИНЯКИН***

(МОСКВА)

## Введение

Задача построения тупиковых покрытий целочисленной матрицы возникает при конструировании логических процедур распознавания и классификации и относится к числу трудных в вычислительном плане задач [2–8, 15]. Данная задача может быть также сформулирована как задача построения максимальных конъюнкций двузначной логической функции, заданной множеством нулей. Поиски эффективных алгоритмов ее решения ведутся с середины 1950-х годов [13, 14].

Пусть $M_{mn}^k$ — множество всех матриц размера $m \times n$ с элементами из $\{0, 1, \ldots, k-1\}$, $k \geqslant 2$; $E_k^r$, $k \geqslant 2$, $r \leqslant n$, — множество всех наборов вида $(\sigma_1, \ldots, \sigma_r)$, где $\sigma_i \in \{0, 1, \ldots, k-1\}$.

Рассмотрим $\sigma \in E_k^r$, $\sigma = (\sigma_1, \ldots, \sigma_r)$. Через $Q_p(\sigma)$, $p \in \{1, 2, \ldots, r\}$, обозначим множество всех наборов $(\beta_1, \ldots, \beta_r)$ в $E_k^r$ таких, что $\beta_p \neq \sigma_p$ и $\beta_j = \sigma_j$ при $j \in \{1, 2, \ldots, r\} \setminus \{p\}$.

Пусть $L \in M_{mn}^k$. *Тупиковым $\sigma$-покрытием* матрицы $L$ называется набор $H$ из $r$ различных столбцов этой матрицы такой, что подматрица $L^H$ матрицы $L$, образованная столбцами набора $H$, обладает следующими двумя свойствами: 1) $L^H$ не содержит строку $\sigma$; 2) если $p \in \{1, 2, \ldots, r\}$, то $L^H$ содержит хотя бы одну строку из множества $Q_p(\sigma)$. Если выполнено только условие 1), то набор столбцов $H$ называется *$\sigma$-покрытием* матрицы $L$. Подматрица матрицы $L$, имеющая с точностью до перестановки строк вид

$$\begin{bmatrix} \beta_1 & \sigma_2 & \sigma_3 & \ldots & \sigma_{r-1} & \sigma_r \\ \sigma_1 & \beta_2 & \sigma_3 & \ldots & \sigma_{r-1} & \sigma_r \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \sigma_1 & \sigma_2 & \sigma_3 & \ldots & \sigma_{r-1} & \beta_r \end{bmatrix},$$

где $\beta_p \neq \sigma_p$ для $p = 1, 2, \ldots, r$, называется *$\sigma$-подматрицей*.

Таким образом, $H$ является тупиковым $\sigma$-покрытием тогда и только тогда, когда $L^H$ не содержит строку $\sigma$ и содержит $\sigma$-подматрицу. Понятие (тупикового) $\sigma$-покрытия введено в [4, 5, 7].

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 07–01–00516), Программы Президента РФ поддержки ведущих научных школ РФ (проект НШ–5294.2008.1), Программы Президента РФ поддержки молодых кандидатов наук (проект МК–6500.2008.9).

Понятие тупикового $(0,\ldots,0)$-покрытия булевой матрицы совпадает с хорошо известным понятием неприводимого покрытия булевой матрицы [13]. Отметим, что $(0,\ldots,0)$-подматрица булевой матрицы является единичной (перестановочной) подматрицей. Единичную подматрицу назовем максимальной, если она не содержится в других единичных подматрицах.

Положим $B(L,\sigma)$, $\sigma\in E_k^r$, — множество всех тупиковых $\sigma$-покрытий матрицы $L$. Пусть далее

$$B_r(L)=\bigcap_{\sigma\in E_k^r}B(L,\sigma),\qquad B(L)=\bigcap_{r=1}^{n}B_r(L).$$

Для почти всех матриц $L$ из $M_{mn}^k$, число покрытий из $B(L)$ растет экспоненциально с ростом размера матрицы, поэтому эффективность алгоритмов построения покрытий из $B(L)$ имеет смысл оценивать в терминах полиномиальной задержки [16].

Пусть $Q(L)$ — конечная совокупность наборов столбцов матрицы $L$, содержащая $B(L)$. Предполагается, что каждый набор из $Q(L)$ не содержит одинаковых столбцов, и некоторые наборы столбцов в $Q(L)$ могут встречаться более одного раза.

Пусть алгоритм $A$ строит покрытия из $B(L)$ путем последовательного просмотра всех наборов из $Q(L)$. При этом каждый набор из $Q(L)$ просматривается столько раз, сколько раз он встречается в $Q(L)$. Таким образом, на каждом шаге алгоритма $A$ строится некоторый набор столбцов $H$ из $Q(L)$ и проверяется принадлежность $H$ к $B(L)$. Число шагов алгоритма $A$ обозначим через $N_A(L)$.

Нас будет интересовать вычислительная сложность алгоритма $A$ в типичном случае (для почти всех матриц из $M_{mn}^k$ при $n\to\infty$).

Будем говорить, что алгоритм $A$ строит $Q(L)$ *с полиномиальной задержкой*, если на каждом шаге выполняется не более $d(m,n)$ элементарных операций и $d(m,n)$ ограничено сверху полиномом от $m,n$. При этом под элементарной операцией понимается просмотр одного элемента матрицы $L$.

Алгоритм $A$ назовем *асимптотически оптимальным*, если $A$ строит $Q(L)$ с полиномиальной задержкой и для почти всех матриц $L$ из $M_{mn}^k$ при $n\to\infty$ величина $N_A(L)$ асимптотически равна числу покрытий из $B(L)$.

Наиболее широкую область практического применения имеют методы построения множества $P(L)$ всех неприводимых покрытий (или тупиковых $(0,\ldots,0)$-покрытий) булевой матрицы $L$ (методам построения сокращенной дизъюнктивной нормальной формы монотонной булевой функции). По сравнению с методами построения тупиковых покрытий общего вида методы построения множества $P(L)$ чаще используются при конструировании и реализации логических процедур распознавания. Логический анализ данных в задачах распознавания сводится к поиску логических закономерностей в обучающей выборке. С этой целью обычно по обучающей выборке строятся специальные булевы матрицы, в которых ищутся определенные семейства $(0,\ldots,0)$-покрытий (в частности тупиковых $(0,\ldots,0)$-покрытий). В случае целочисленной информации можно обойтись без построения вспомогательных булевых матриц и для поиска логических закономерностей использовать методы построения тупиковых покрытий целочисленной матрицы, применяя их непосредственно к обучающей выборке (см. [4–8, 15]). Понятие покрытия общего вида впервые было введено в [4] в связи с усовершенствованием существовавших моделей алгоритмов голосования по представительным наборам (алгоритмов распознавания типа «Кора»).

В [2–5, 7] рассмотрен случай, когда число строк $m$ булевой матрицы $L$ имеет более низкий порядок роста, чем число столбцов $n$, при условии, что

$n \to \infty$. Для этого случая построен асимптотически оптимальный алгоритм поиска неприводимых покрытий (алгоритм АО1). Данный алгоритм строит с задержкой, не превосходящей $O(mn)$, приближенное решение, в качестве которого рассматривается совокупность наборов столбцов, содержащих единичные подматрицы. Каждый такой набор алгоритм АО1 строит столько раз, сколько единичных подматриц он содержит. Показано, что если $m^\alpha \leqslant n \leqslant 2^{m^\beta}$, $\alpha > 1$, $\beta < 1$, то при $n \to \infty$ число шагов данного алгоритма, равное числу единичных подматриц, почти всегда (для почти всех матриц размера $m \times n$) асимптотически равно мощности $P(L)$. При конструировании алгоритма АО1 в качестве приближенного решения может быть рассмотрена совокупность наборов столбцов, содержащих максимальные единичные подматрицы (каждый такой набор столбцов строится столько раз, сколько максимальных единичных подматриц он содержит). Тогда алгоритм будет делать меньшее число шагов и работать с задержкой, не превосходящей $O(qmn)$; здесь и далее $q = \min(m, n)$. Данная модификация алгоритма АО1 обычно используется при его реализации на ЭВМ [1]. В дальнейшем речь будет идти именно об этой модификации.

В [6, 15] построен алгоритм, основанный на переборе с задержкой, не превосходящей $O(qm^2n)$ неприводимых покрытий матрицы $L$ (алгоритм АО2). Однако данный алгоритм строит каждый набор из $P(L)$ столько раз, сколько единичных подматриц он содержит. Из сказанного выше следует, что указанный недостаток не существенен при $m^\alpha \leqslant n \leqslant 2^{m^\beta}$, $\alpha > 1$, $\beta < 1$ (в этом случае число шагов алгоритма АО2, равное числу единичных подматриц, порождающих неприводимые покрытия, почти всегда при $n \to \infty$ асимптотически равно мощности $P(L)$).

Отметим, что проверка на повторяемость построенного на очередном шаге набора столбцов в алгоритмах АО1 и АО2 требует просмотра не более $O(qm)$ элементов матрицы $L$.

В [9, 10] построен алгоритм поиска неприводимых покрытий булевой матрицы (алгоритм ОПТ), асимптотически оптимальный при тех же условиях, что и алгоритмы АО1 и АО2. Данный алгоритм основан на переборе с задержкой, не превосходящей $O(qmn(m+q))$ наборов столбцов матрицы, содержащих единичные подматрицы и удовлетворяющих некоторым дополнительным условиям. Результаты численных экспериментов со случайными матрицами при различных соотношениях между $m$ и $n$ показали, что практически во всех случаях алгоритм ОПТ делает существенно меньшее число шагов и поэтому работает существенно быстрее по сравнению с алгоритмами АО1 и АО2, а также другими известными алгоритмами поиска неприводимых покрытий булевой матрицы. Указанные результаты приведены в §5.

Алгоритм АО1 несложно модифицировать для поиска покрытий целочисленной матрицы [7]. Для построения аналогичных модификаций алгоритмов поиска неприводимых покрытий, использующих выбрасывание «охватывающих» строк в подматрицах матрицы $L$ (например, алгоритмов АО2 и ОПТ), можно воспользоваться предложенным в [8, 11] способом сведения задачи построения тупиковых покрытий целочисленной матрицы к построению неприводимых покрытий булевой матрицы. Указанный способ основан на преобразовании $L$ во вспомогательную булеву матрицу $L^*$ из $m$ строк и $kn$ столбцов. Столбцы матрицы $L^*$ разбиты на $n$ групп по $k$ столбцов в каждой группе. Задача построения $B(L)$ сводится к построению неприводимых покрытий матрицы $L^*$, содержащих не более одного столбца из каждой группы. Данная схема используется в настоящей работе для модификации алгоритма ОПТ на случай поиска покрытий из $B(L)$, $L \in M_{mn}^k$. В результате построен алгоритм ОПТ+ поиска покрытий из $B(L)$,

асимптотически оптимальный при $m^{\alpha} \leqslant n \leqslant k^{m^{\beta}}$, $\alpha > 1$, $\beta < 1$, и работающий с задержкой $O(kumn(m+u))$; здесь и далее $u = \min(m, kn)$. Асимптотическая оптимальность алгоритма ОПТ+ следует из того, что число его шагов не превосходит числа всех $\sigma$-подматриц, которое в рассматриваемом случае, как показано в [4, 5, 7], почти всегда асимптотически равно числу покрытий из $B(L)$.

## § 1. Сведение задачи построения тупиковых покрытий целочисленной матрицы к задаче построения неприводимых покрытий булевой матрицы

Пусть $L = (a_{ij})$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, — матрица из $M_{mn}^{k}$. Покажем, что задачу построения множества $B(L)$ можно свести к задаче построения тупиковых $(0, \ldots, 0)$ покрытий (неприводимых покрытий) булевой матрицы, которая специальным образом строится по $L$ [8, 11].

Пусть $a \in \{0, 1, \ldots, k-1\}$. Положим

$$\delta(a_{ij}, a) = \begin{cases} 1, & a_{ij} \neq a, \\ 0, & a_{ij} = a, \end{cases}$$

$i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$.

Построим булеву матрицу $L^{*}$, состоящую из $m$ строк и $kn$ столбцов, в которой строка с номером $i$, $i \in \{1, 2, \ldots, m\}$, имеет вид:

$$(\delta(a_{i1},0), \ldots, \delta(a_{i1},k-1), \delta(a_{i2},0), \ldots, \delta(a_{i2},k-1), \ldots, \delta(a_{in},0), \ldots, \delta(a_{in},k-1)).$$

Нетрудно заметить, что столбцу с номером $j$, $j \in \{1, 2, \ldots, n\}$, исходной матрицы $L$ соответствует группа из $k$ столбцов матрицы $L^{*}$ с номерами $k(j-1)+1, k(j-1)+2, \ldots, kj$. Числа $k(j-1)+1, k(j-1)+2, \ldots, kj$, $j \in \{1, 2, \ldots, n\}$, назовем родственными. Очевидным является

Утверждение 1. *Набор из $r$ различных столбцов матрицы $L$ с номерами $j_1, j_2, \ldots, j_r$ является тупиковым $(\sigma_1, \ldots, \sigma_r)$-покрытием, $(\sigma_1, \ldots, \sigma_r) \in E_k^r$, тогда и только тогда, когда набор столбцов матрицы $L^{*}$, с номерами $p_1, p_2 \ldots, p_r$, где $p_i = (j_i - 1)k + \sigma_i + 1$ при $i = 1, 2, \ldots, r$ принадлежит $P(L^{*})$.*

Из утверждения 1 следует, что задача построения множества $B(L)$ сводится к построению подмножества $\widetilde{P}(L^{*})$ множества $P(L^{*})$, состоящего из всех таких неприводимых покрытий, которые не содержат столбцов с родственными номерами.

Набор $G$ из $r$ различных элементов множества $\{1, 2, \ldots, kn\}$ назовем *совместимым* для $L^{*}$, если подматрица матрицы $L^{*}$, образованная столбцами с номерами из $G$, содержит единичную подматрицу размера $r \times r$.

Проверка набора $G = \{p_1, \ldots, p_r\}$ на совместимость достаточно очевидна и заключается в просмотре строк подматрицы матрицы $L^{*}$, образованной столбцами с номерами $p_1, \ldots, p_r$. Если указанная подматрица содержит каждую из строк $(1, 0, 0, \ldots, 0, 0)$, $(0, 1, 0, \ldots, 0, 0)$, $\ldots$, $(0, 0, 0, \ldots, 0, 1)$ длины $r$, то набор $G$ совместим. Требуемое число элементарных операций не превосходит $O(kmn)$.

Обозначим через $V(L^{*})$ совокупность всех совместимых наборов из элементов множества $\{1, 2, \ldots, kn\}$, через $\widetilde{V}(L^{*})$ совокупность наборов из $V(L^{*})$, не содержащих родственных номеров, и, наконец, через $V^{(2)}(L^{*})$ совокупность наборов из $V(L^{*})$, состоящих в точности из двух родственных чисел.

Утверждение 2. *Имеет место*

$$\widetilde{V}(L^*) = V(L^*) \setminus V^{(2)}(L^*).$$

Утверждение 2 прямо следует из того, что в подматрице матрицы $L^*$, составленной из группы родственных столбцов, каждая строка имеет ровно один элемент, равный 0. Следовательно, если длина совместимого набора больше двух, то он не содержит родственных чисел.

Из утверждения 2 следует

Утверждение 3. *Набор столбцов $H$ матрицы $L^*$ с номерами $p_1, \ldots, p_r$ принадлежит $\widetilde{P}(L^*)$ тогда и только тогда, когда $\{p_1, \ldots, p_r\} \in V(L^*) \setminus V^{(2)}(L^*)$ и $H$ является $(0, \ldots, 0)$-покрытием.*

Таким образом, задача построения множества $B(L)$ сводится к построению всех $(0, \ldots, 0)$-покрытий матрицы $L^*$, состоящих из столбцов с номерами из $V(L^*) \setminus V^{(2)}(L^*)$.

## § 2. Асимптотически оптимальный алгоритм поиска тупиковых покрытий целочисленной матрицы ОПТ+

По матрице $L$ из $M_{mn}^k$, $k \geqslant 2$, построим булеву матрицу $L^*$ описанным в §1 способом. Пусть $L^* = (b_{ij})$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, kn$.

Будем говорить, что строка с номером $i$ матрицы $L^*$ *охватывает* строку с номером $p$, если $b_{ij} \geqslant b_{pj}$ при $j = 1, 2, \ldots, kn$.

Заметим, множество $(0, \ldots, 0)$-покрытий булевой матрицы не меняется при выбрасывании из нее охватывающей строки, в частности сохраняются все тупиковые $(0, \ldots, 0)$-покрытия и не появляется новых тупиковых $(0, \ldots, 0)$-покрытий. Очевидно также, что нулевой столбец не входит ни в одно тупиковое $(0, \ldots, 0)$-покрытие множество.

В случае одинаковых строк каждая из них согласно определению является охватывающей. Процедура удаления охватывающих строк заключается в последовательном просмотре строк матрицы и удалении очередной строки, если она охватывает хотя бы одну из строк, не удаленных к данному моменту. В результате из двух одинаковых строк может остаться только одна, и она остается, если нет других строк, которые она охватывает.

Пусть $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, kn\}$. Если $b_{ij} = 1$, то будем говорить, что столбец с номером $j$ (строка с номером $i$) матрицы $L^*$ *покрывает* строку с номером $i$ (столбец с номером $j$).

Если $G \in V(L^*)$, $G = \{p_1, \ldots, p_r\}$, то будем предполагать, что $p_1 < \ldots < p_r$. Будем говорить, что элемент $j$ множества $\{1, 2, \ldots, kn\}$, $j > p_r$, *совместим* с набором $G$, если $G \cap \{j\} \in V(L^*)$.

Пусть $L_0(G)$ — подматрица матрицы $L^*$, образованная столбцами с номерами $p_1, p_1 + 1, \ldots, kn$. Если $L_0(G)$ не содержит нулевых строк, то через $\widehat{L}_0(G)$ обозначим подматрицу матрицы $L^*$, полученную из $L_0(G)$ путем удаления охватывающих строк с последующим удалением нулевых столбцов. При $r \geqslant 2$ через $L_t(G)$, $t \in \{1, 2, \ldots, r-1\}$, обозначим подматрицу матрицы $L^*$, образованную строками, не покрытыми столбцами с номерами $p_1, \ldots, p_t$, и столбцами, номера которых совместимы с $\{p_1, \ldots, p_t\}$. Если $L_t(G)$ не содержит нулевых строк, то удалим из нее охватывающие строки и затем нулевые столбцы. Получим подматрицу $\widehat{L}_t(G)$. В каждой из подматриц $L_t(G)$ и $\widehat{L}_t(G)$, $t \in \{0, 1, \ldots, r-1\}$, сохраним исходную нумерацию строк и столбцов.

Набор $G$ из $V(L^*)$ вида $\{p_1, \ldots, p_r\}$ назовем *правильным* для $L^*$, если при $t \in \{0, 1, \ldots, r-1\}$, подматрица $L_t(G)$ не содержит нулевых строк и подматрица $\widehat{L}_t(G)$ содержит столбцы с номерами $p_{t+1}, \ldots, p_r$.

Обозначим через $W(L^*)$ совокупность всех правильных наборов из элементов множества $j \in \{1, 2, \ldots, kn\}$.

Набор $G$ из $W(L^*)$ вида $\{p_1, \ldots, p_r\}$ назовем *максимальным* для $L^*$, если при любом $p$, $p_r < p \leqslant kn$, набор $G \cap \{p\}$ не принадлежит $W(L^*)$.

Обозначим через $W_{\max}(L^*)$ совокупность всех максимальных наборов для $L^*$, через $\widetilde{W}_{\max}(L^*)$ совокупность всех максимальных наборов для $L^*$, не содержащих родственных чисел, и наконец, через $W^{(2)}_{\max}(L^*)$ совокупность всех максимальных наборов, состоящих в точности из двух родственных чисел. Очевидным является

Утверждение 4. *Имеет место*

$$\widetilde{W}_{max}(L^*) = W_{max}(L^*) \setminus W^{(2)}_{max}(L^*).$$

Утверждение 5. *Набор столбцов $H$ матрицы $L^*$ с номерами $p_1, \ldots, p_r$ принадлежит $\widetilde{P}(L^*)$ тогда и только тогда, когда $\{p_1, \ldots, p_r\} \in \widetilde{W}_{max}(L^*)$ и $H$ — покрытие матрицы $L^*$.*

Доказательство. Положим $G = \{p_1, \ldots, p_r\}$. Пусть $G \in \widetilde{W}_{\max}(L^*)$ и $H$ — покрытие матрицы $L^*$. Очевидно, в $\widehat{L}_{r-1}(G)$ столбец с номером $p_r$ является единичным и, следовательно, $H \in P(L^*)$. Из того, что $G$ не содержит родственных чисел, следует, что $H \in \widetilde{P}(L^*)$.

Пусть $H \in \widetilde{P}(L^*)$. Покажем, что $G \in \widetilde{W}_{\max}(L^*)$. Так как $G$ не содержит родственных чисел, то достаточно показать, что $G \in W_{\max}(L^*)$. По определению набор столбцов $H$ не содержит нулевой строки. Пусть $t \in \{0, 1, \ldots, r-1\}$. Тогда столбцы матрицы $L_t(G)$ с номерами $p_{t+1}, \ldots, p_r$ образуют неприводимое покрытие матрицы $L_t(G)$. Следовательно, и столбцы матрицы $\widehat{L}_t(G)$ с номерами $p_{t+1}, \ldots, p_r$ образуют неприводимое покрытие, т. е. матрица $\widehat{L}_t(G)$ содержит столбцы с номерами $p_{t+1}, \ldots, p_r$. Таким образом, $G \in W_{\max}(L^*)$, что и требовалось доказать.

Описываемый ниже алгоритм ОПТ+ строит множество $B(L)$, перебирая наборы из $\widetilde{W}_{\max}(L^*)$.

Пусть $p \in \{1, 2, \ldots, kn\}$. Число $p$ назовем *корректным*, если выполнено одно из двух следующих условий: 1) $\{p\} \in W_{\max}(L^*)$; 2) $\{p\} \notin W_{\max}(L^*)$, но в подматрице матрицы $L_0(\{p\})$, составленной из столбца с номером $p$ и столбцов с номерами не родственными $p$, нет нулевых строк. Обозначим через $R(L)$ множество всех корректных чисел в $\{1, 2, \ldots, kn\}$. Множество $R(L)$ упорядочим по возрастанию его элементов.

Очевидно, что $p \in R(L)$ тогда и только тогда, когда $\widetilde{W}_{\max}(L^*)$ содержит хотя бы один набор с первым элементом $p$.

Пусть матрица $L'$ либо совпадает с $L^*$, либо является подматрицей матрицы $L^*$, образованной столбцами с номерами $p, p+1, \ldots, kn$ и не содержащей нулевых строк. Из $L'$ удалим охватывающие строки и затем нулевые столбцы. В полученной подматрице номер первого столбца обозначим через $p_1$.

Набор $G$ из $\widetilde{W}_{\max}(L^*)$, $G = \{p_1, \ldots, p_r\}$, назовем *начальным* в $L'$, если $p_{t+1}$ — номер первого столбца в $\widehat{L}_t(G)$ при $t \in \{0, 1, \ldots, r-1\}$.

Опишем процедуру построения начального набора в $L'$.

Положим $G = \{p_1\}$. Строим матрицу $L_1(G)$, удаляем из нее столбцы, номера которых родственны числу $p_1$. Полученную подматрицу обозначим через $L_1$.

Если $L_1$ пуста или содержит нулевую строку, то $\{p_1\}$ — начальный набор в $L'$. В противном случае из $L_1$ удаляем охватывающие строки и нулевые столбцы и полагаем $G=\{p_1,p_2\}$, где $p_2$ — номер первого не удаленного столбца. Строим матрицу $L_2(G)$. Если $L_2(G)$ пуста или содержит нулевую строку, то $\{p_1,p_2\}$ — начальный набор в $L'$. В противном случае строится матрица $\widehat{L}_2(G)$ и в ней выбирается столбец с наименьшим номером $p_3$. Согласно утверждению 2 число $p_3$ не будет родственным числу $p_2$.

Процесс продолжается до тех пор, пока на очередной итерации $t$, $t\geqslant 2$, либо подматрица $L_t(G)$ будет содержать нулевую строку, либо будет пуста. Построенный набор $\{p_1,\ldots,p_t\}$ будет начальным набором в $L'$.

Покажем, что построение начального набора в $L'$ требует выполнения не более, чем $O(kumn(m+u))$ элементарных операций.

Пусть $\{p_1,\ldots,p_t\}$ — начальный набор в $L'$. Очевидно, $t\leqslant\min(m,kn)$. При выборе элемента $p_i$, $i\in\{1,2,\ldots,t\}$, и построении матрицы $L_i(G)$ на итерации $i$ выполняются следующие действия:

1) удаление охватывающих строк (не более, чем $O(m^2kn)$ элементарных операций);

2) удаление нулевых столбцов (не более $O(mkn)$ элементарных операций);

3) удаление строк, покрытых столбцом $p_i$ (не более $O(m)$ элементарных операций);

4) удаление несовместимых столбцов (не более $O(mukn)$ элементарных операций);

5) удаление родственных столбцов (не требуется просмотра элементов матрицы).

Отсюда следует, что для построения начального набора требуется не более, чем $O(kumn(m+u))$ элементарных операций.

Пусть $G\in\widetilde{W}_{\max}(L^*)$, $G=\{p_1,\ldots,p_r\}$, $2\leqslant t\leqslant r$, $L'$ — подматрица матрицы $L_{t-1}(G)$, образованная всеми такими столбцами матрицы $L_{t-1}(G)$, номера которых больше $p_t$.

Пусть $L'$ не пуста и не содержит нулевых строк. Из $L'$ удалим охватывающие строки и нулевые столбцы. В полученной подматрице номер первого столбца обозначим через $p'_t$.

Набор $G'=\{p'_t,\ldots,p'_{t+q}\}$ назовем *начальным в* $L_{t-1}(G)$ *относительно* $\{p_1,\ldots,p_t\}$, если $(\{p_1,\ldots,p_{t-1}\}\cap G')$ и $p'_{t+u+1}$ — номер первого столбца в $\widehat{L}_{t+u}(\{p_1,\ldots,p_{t-1}\}\cap G')$ при $u\in\{0,1,\ldots,q-1\}$.

Опишем процедуру построения начального набора в $L_{t-1}(G)$ относительно $\{p_1,\ldots,p_t\}$.

Положим $G'=\{p'_t\}$. Строим матрицу $L_1=L_t(\{p_1,\ldots,p_{t-1}\}\cap G')$.

Если $L_1$ пуста или содержит нулевую строку, то $\{p'_t\}$ — требуемый начальный набор. В противном случае из $L_1$ удаляем охватывающие строки и нулевые столбцы и полагаем $G'=\{p'_t,p'_{t+1}\}$, где $p'_{t+1}$ — номер первого не удаленного столбца. Строим матрицу $L_{t+1}(\{p_1,\ldots,p_{t-1}\}\cap G')$. Если эта матрица пуста или содержит нулевую строку, то $\{p'_t,p'_{t+1}\}$ — требуемый начальный набор. В противном случае строится матрица $\widehat{L}_{t+1}(\{p_1,\ldots,p_{t-1}\}\cap G')$ и в ней выбирается столбец с наименьшим номером.

Процесс продолжается до тех пор, пока на очередной итерации $q$, $q\geqslant 2$, либо подматрица $L_{t+q}(\{p_1,\ldots,p_{t-1}\}\cap G')$, $G'=\{p'_t,\ldots,p'_{t+q}\}$, будет содержать нулевую строку, либо будет пуста. Построенный набор $\{p'_t,\ldots,p'_{t+q}\}$ будет начальным набором в $L_{t-1}(G)$ относительно $\{p_1,\ldots,p_t\}$.

Нетрудно видеть, что построение начального набора в $L_{t-1}(G)$ относительно $\{p_1, \ldots, p_t\}$ требует выполнения не более, чем $O(kumn(m+u))$ элементарных операций.

Упорядочим $\widetilde{W}_{\max}(L^*)$ в лексикографическом порядке, сопоставляя наборам из $\widetilde{W}_{\max}(L^*)$ слова в алфавите $\{1, 2, \ldots, kn\}$. Для каждого набора $G$ из $\widetilde{W}_{\max}(L^*)$, не являющегося максимальным элементом в $\widetilde{W}_{\max}(L^*)$, покажем, как построить непосредственно следующий за ним набор $\triangleleft G$.

Пусть $G = \{p_1, \ldots, p_r\}$. Возможны три случая:

1) $r = 1$;

2) $r > 1$ и подматрица матрицы $L_1(G)$, образованная столбцами с номерами бóльшими $p_2$, либо пуста, либо содержит нулевую строку;

3) $r > 1$ и в $\{2, \ldots, r\}$ можно указать максимальное число $t$ такое, что подматрица $L'$ матрицы $L_{t-1}(G)$, образованная столбцами с номерами бóльшими $p_t$, не пуста и не содержит нулевых строк.

Пусть имеет место один из первых двух случаев. По условию $G$ не является максимальным элементом в $\widetilde{W}_{\max}(L^*)$. Следовательно, в $R(L)$ есть хотя бы одно число, которое больше $p_1$. В качестве $\triangleleft G$ нужно взять начальный набор в подматрице матрицы $L^*$, образованной столбцами с номерами $p, p+1, \ldots, kn$, где $p$ — число, непосредственно следующее за числом $p_1$ в $R(L)$.

Пусть имеет место случай 3. Тогда $\triangleleft G = (G \setminus \{p_t, p_{t+1}, \ldots, p_r\}) \cap G'$, где $G'$ — начальный набор в $L_{t-1}(G)$ относительно $\{p_1, \ldots, p_t\}$.

Алгоритм строит $B(L)$ за $|\widetilde{W}_{\max}(L^*)|$ шагов. На шаге $i$ алгоритм выбирает в $L^*$ максимальный набор $Q[i, L^*]$ и проверяет, является ли покрытием набор столбцов матрицы $L^*$ с номерами из $Q[i, L^*]$. Если проверяемый набор столбцов является покрытием матрицы $L^*$, то строится соответствующее покрытие из $B(L)$. Выбор максимальных наборов происходит по следующим правилам:

1) $Q[1, L^*]$ — начальный набор в $L^*$;

2) если $Q[i, L^*]$ имеет следующий за ним набор в $W_{\max}(L^*)$, то $Q[i+1, L^*] = \triangleleft Q[i, L^*]$, в противном случае алгоритм заканчивает работу.

Оценка сложности шага алгоритма $O(kumn(m+u))$ получается из приведенной выше оценки сложности построения на данном шаге начального набора.

Описанная схема модификации алгоритма ОПТ для поиска покрытий из $B(L)$ применима и к другим алгоритмам поиска неприводимых покрытий, например к алгоритмам АО1 и АО2. Модификации этих алгоритмов на целочисленный случай обозначим соответственно АО1+ и АО2+. Алгоритмы АО1+ и АО2+ будут асимптотически оптимальными при $m^\alpha \leqslant n \leqslant k^{m^\beta}$, $\alpha > 1$, $\beta < 1$, $n \to \infty$.

## § 3. Использование аппарата деревьев для описания алгоритмов поиска тупиковых покрытий

Из вышеизложенного следует, что первостепенное значение имеет разработка эффективных алгоритмов поиска неприводимых покрытий булевой матрицы.

Как правило, работа алгоритмов поиска неприводимых покрытий булевой матрицы $L$ заключается в одностороннем обходе дерева решений, вершинам которого соответствуют пары $(L_H, H)$, где $H$ — набор столбцов матрицы $L$, $L_H$ — подматрица матрицы $L$, образованная строками, не «покрытыми» набором $H$, и некоторыми из столбцов матрицы $L$, не входящими

в набор $H$. Корневая вершина остается «пустой» (имеет вид $(L, \varnothing)$). Для висячих вершин проверяется принадлежность $H$ к множеству неприводимых покрытий матрицы $L$ (очевидно, такая проверка требует просмотра не более $qm$ элементов матрицы $L$). Под шагом алгоритма понимается переход от одной висячей вершины к следующей. Переход по ветви дерева решений от одной вершины к следующей осуществляется путем добавления к набору $H$ некоторого столбца матрицы $L$. При этом предполагается, что число элементарных операций, выполняемых на каждом шаге, ограничено сверху полиномом от размера матрицы $L$.

В алгоритме АО1 каждая вершина дерева решений порождается единичной подматрицей матрицы $L$. Пусть единичная подматрица матрицы $L$ образована единичными элементами $a_{i_1 j_1}, \ldots, a_{i_r j_r}$, где $j_1 < \ldots < j_r$. Тогда эта подматрица порождает вершину, которой соответствует пара $(L'_H, H)$, где $H$ — набор столбцов с номерами $j_1, \ldots, j_r$, а $L'_H$ образована строками, не покрытыми столбцами из $H$, и столбцами с номерами бóльшими $j_r$, не покрытыми строками с номерами $i_1, \ldots, i_r$. Висячим вершинам соответствуют наборы столбцов, содержащие максимальные единичные подматрицы, т. е. единичные подматрицы, не содержащиеся в других единичных подматрицах.

Заметим, что алгоритм АО1 нетрудно модифицировать на более общий случай (для поиска тупиковых покрытий целочисленной матрицы) без построения вспомогательной матрицы $L^*$.

В алгоритме АО2 каждая вершина дерева решений также порождается единичной подматрицей матрицы $L$. Пусть единичная подматрица матрицы $L$ образована единичными элементами $a_{i_1 j_1}, \ldots, a_{i_r j_r}$, где $j_1 < \ldots < j_r$. Тогда эта подматрица порождает вершину, которой соответствует пара $(L''_H, H)$, где $H$ — набор столбцов с номерами $j_1, \ldots, j_r$, а $L''_H$ получается из $L'_H$ выбрасыванием охватывающих строк. Висячим вершинам соответствуют неприводимые покрытия матрицы $L$.

В [11] построен эффективный при практическом применении алгоритм поиска неприводимых покрытий (алгоритм спуска с удалением охватывающих столбцов, обозначаемый далее УОС). Каждой вершине в алгоритме УОС соответствует пара $(L^*_H, H)$. Здесь $H$ — некоторый набор столбцов матрицы $L$, состоящий из попарно несравнимых столбцов с номерами $j_1, \ldots, j_r$, $j_1 < \ldots < j_r$, а $L^*_H$ — подматрица матрицы $L$, образованная строками, не покрытыми столбцами из $H$, и столбцами с номерами бóльшими $j_r$, несравнимыми (не охватывающими) столбец с номером $j_r$ в подматрице $L^*_{H'}$, где $H'$ — набор столбцов матрицы $L$ номерами $j_1, \ldots, j_{r-1}$. Алгоритм УОС имеет задержку, не превосходящую $O(m^2 n)$, но не является асимптотически оптимальным.

Работу алгоритма ОПТ также удобно представить в виде процесса построения дерева решений $\text{Д}_L$. Вершины в $\text{Д}_L$ (за исключением корневой вершины) порождаются наборами из $W(L)$. Пусть $G$ — набор из $W(L)$, $G = \{j_1, \ldots, j_r\}$. Тогда $G$ порождает вершину $(L_r(G), H)$, где $H$ — набор столбцов матрицы $L$ с номерами $j_1, \ldots, j_r$. Висячие вершины порождаются наборами из $W_{\max}(L)$. Если $(L_r(G), H)$ — висячая вершина и матрица $L_r(G)$ пуста, то $H$ — неприводимое покрытие матрицы $L$. Общее число вершин в $\text{Д}_L$ равно $|W(L)| + 1$, число висячих вершин равно $|W_{\max}(L)|$. Формулируются правила, позволяющие при построении дерева $\text{Д}_L$ переходить от одной вершины к другой в таком порядке, который соответствует последовательному просмотру его ветвей. Переход от одной внутренней вершины к следующей вершине полиномиален относительно размера матрицы $L$ (требует просмотра не более $O(mn(m+q))$ элементов матрицы $L$). Переход от одной висячей вершины к следующей висячей вершине также полиномиален (требует просмотра не более $O(qmn(m+q))$ элементов мат-

рицы $L$). Построение первой висячей вершины осуществляется за время, не превосходящее $O(qmn(m+q))$. Таким образом, вычислительная сложность алгоритма не превосходит $O(m^3n)|W_{\max}(L)|$, если $m \leqslant n$, и не превосходит $O(m^2n^2+mn^3)|W_{\max}(L)|$, если $m \geqslant n$.

Обозначим через $W^{(2)}(L^*)$ совокупность всех правильных наборов из двух родственных элементов множества $\{0, 1, \ldots, kn\}$.

Нетрудно видеть, что алгоритм ОПТ+ строит дерево решений, в котором вершины порождаются наборами из $\widetilde{W}(L^*) = W(L^*) \setminus W^{(2)}(L^*)$, и покрытия из $B(L)$ находятся среди висячих вершин. Аналогичное построение деревьев решений может быть проведено и для алгоритмов АО1+, АО2+ и УОС+ (УОС+ — целочисленная модификация алгоритма УОС).

## § 4. Асимптотически оптимальное построение максимальных конъюнкций двузначной логической функции

Пусть функция $F(x_1, \ldots, x_n)$ определена на $E_k^n$ и принимает значения из $\{0, 1\}$; $B_F$ — множество наборов, на которых $F = 0$, $|B_F| = m$. Обозначим через $N_{mn}^k$ множество всех таких функций. Положим

$$x^\sigma = \begin{cases} 1, & \text{если } x = \sigma, \\ 0, & \text{если } x \neq \sigma, \end{cases}$$

$x, \sigma \in \{0, 1, \ldots, k-1\}$.

Обычным образом введем понятие элементарной конъюнкции. Элементарной конъюнкцией над переменными $x_1, \ldots, x_n$ назовем выражение вида $x_{j_1}^{\sigma_1} \ldots x_{j_r}^{\sigma_r}$, где $x_{j_i} \in \{x_1, \ldots, x_n\}$ при $i = 1, 2, \ldots, r$ и $x_{j_q} \neq x_{j_t}$ при $t, q \in \{1, 2, \ldots, r\}$. Элементарная конъюнкция принимает значение 1 тогда и только тогда, когда каждый ее множитель равен 1.

Пусть $N_B$ — интервал истинности элементарной конъюнкции $B$. Элементарную конъюнкцию $B$ назовем *допустимой* для $F$, если $N_B \cap B_F = \varnothing$. Элементарную конъюнкцию $B$ назовем *максимальной* для $F$, если она является допустимой и не существует допустимой для $F$ конъюнкции $B'$ такой, что $N_{B'} \supset N_B$.

Пусть $L_F$ — матрица, строками которой являются наборы из $B_F$. Очевидным является

Утверждение 6. *Элементарная конъюнкция $x_{j_1}^{\sigma_1} \ldots x_{j_r}^{\sigma_r}$ является максимальной для $F$ тогда и только тогда, когда набор столбцов матрицы $L$ с номерами $j_1, \ldots, j_r$ является тупиковым $(\sigma_1, \ldots, \sigma_r)$-покрытием.*

Модифицируя алгоритм ОПТ+, описанный в §2, для построения множества $R(F)$ максимальных конъюнкций функции $F(x_1, \ldots, x_n)$ из $N_{mn}^k$ получаем асимптотически оптимальный алгоритм, вычислительная сложность которого почти всегда асимптотически не превосходит $O(km^3n)|R(F)|$, если $m \leqslant kn$, и не превосходит $O(k^2m^2n^2+k^3mn^3)$, если $m \geqslant kn$.

## § 5. Результаты численных экспериментов

Алгоритмы АО1, АО2, УОС, ОПТ, УОС+ и ОПТ+ были реализованы на языке С++ для ЭВМ на базе процессоров семейства x86. Для сравнения алгоритмов АО1, АО2, УОС и ОПТ была проведена серия экспериментов на случайных булевых матрицах размера $m \times n$, полученных с помощью $mn$-кратного обращения к датчику случайных чисел с равновероятным по-

явлением 0 и 1. Для каждой пары $(m, n)$ обсчитывалось по 20 матриц. По результатам счета вычислялась статистическая эффективность (СЭ) каждого алгоритма по формуле $\text{СЭ} = \frac{1}{20}\sum_{i=1}^{20}\frac{|P(L_i)|}{N(L_i)}$, где $|P(L_i)|$ — число неприводимых покрытий матрицы $L_i$, $N(L_i)$ — число шагов алгоритма для матрицы $L_i$.

Алгоритмы сравнивались в случаях **а)** $m \ll n$, **б)** $m < n$, **в)** $m = n$ и **г)** $m > n$. Результаты счета приведены в табл. 1–4 и на рис. 1–4.

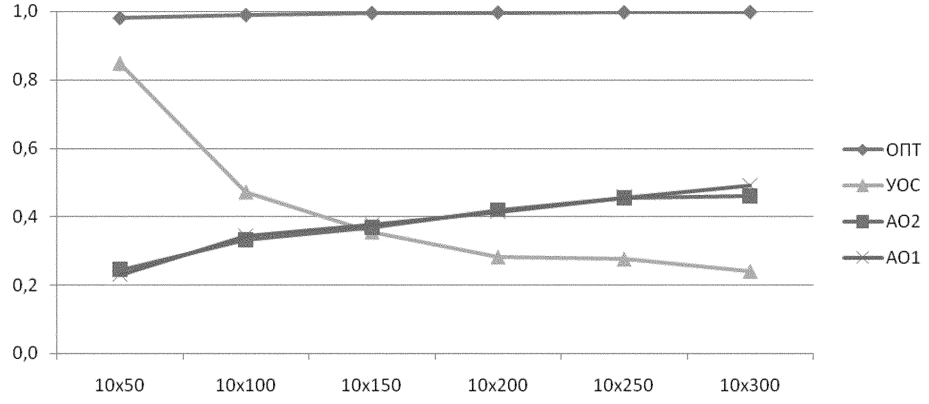



Рис. 1. Статистическая эффективность, случай $m \ll n$

Таблица 1. Случай $m \ll n$

| Размер матрицы $m \times n$ | Среднее время счета в миллисекундах | | | | Среднее число покрытий | Статистическая эффективность (СЭ) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | АО1 | АО2 | УОС | ОПТ | | АО1 | АО2 | УОС | ОПТ |
| $10 \times 50$ | 79 | 82 | 16 | 15 | 7 922 | 0,2303 | 0,2443 | 0,8479 | 0,9796 |
| $10 \times 100$ | 1 499 | 1 421 | 328 | 297 | 155 688 | 0,3435 | 0,3317 | 0,4725 | 0,9884 |
| $10 \times 150$ | 8 307 | 10 672 | 2 266 | 1 828 | 816 479 | 0,3776 | 0,3678 | 0,3556 | 0,9944 |
| $10 \times 200$ | 30 359 | 38 448 | 10 500 | 7 281 | 3 056 941 | 0,4143 | 0,4188 | 0,2821 | 0,9958 |
| $10 \times 250$ | 79 652 | 102 853 | 32 563 | 19 703 | 8 091 615 | 0,4569 | 0,4535 | 0,2761 | 0,9964 |
| $10 \times 300$ | 158 438 | 246 736 | 97 375 | 40 609 | 16 945 056 | 0,4917 | 0,4603 | 0,2402 | 0,9973 |

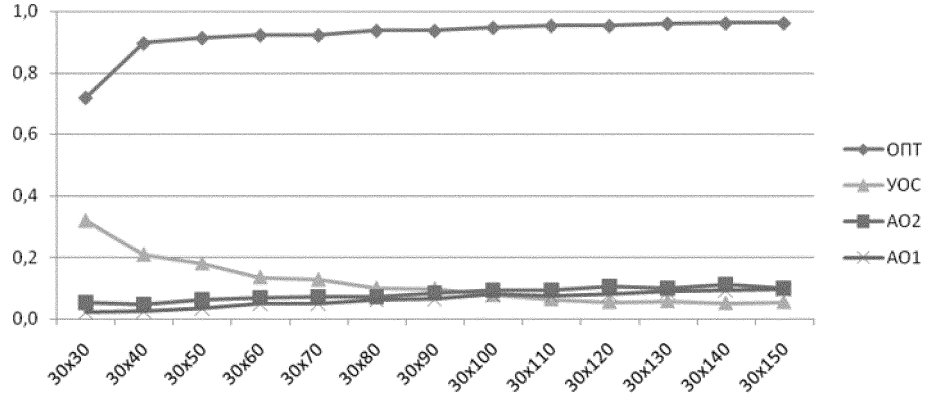



Рис. 2. Статистическая эффективность, случай $m < n$

Таблица 2. Случай $m < n$

| Размер матрицы $m \times n$ | Среднее время счета в миллисекундах | | | | Среднее число покрытий | Статистическая эффективность (СЭ) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | АО1 | АО2 | УОС | ОПТ | | АО1 | АО2 | УОС | ОПТ |
| $30 \times 40$ | 4 273 | 1 600 | 282 | 297 | 33 050 | 0,0233 | 0,0475 | 0,2088 | 0,8965 |
| $30 \times 50$ | 13 608 | 6 780 | 1 188 | 1 250 | 113 468 | 0,0320 | 0,0621 | 0,1805 | 0,9137 |
| $30 \times 60$ | 29 518 | 16 273 | 3 282 | 2 922 | 262 326 | 0,0479 | 0,0675 | 0,1356 | 0,9226 |
| $30 \times 70$ | 55 810 | 36 559 | 7 047 | 5 640 | 552 520 | 0,0498 | 0,0715 | 0,1275 | 0,9227 |
| $30 \times 80$ | 130 390 | 82 787 | 17 547 | 11 297 | 1 296 193 | 0,0602 | 0,0710 | 0,1000 | 0,9375 |
| $30 \times 90$ | 277 317 | 195 758 | 44 572 | 27 344 | 2 715 924 | 0,0633 | 0,0838 | 0,0976 | 0,9371 |
| $30 \times 100$ | 463 957 | 317 633 | 64 219 | 38 047 | 3 993 025 | 0,0779 | 0,0923 | 0,0780 | 0,9473 |
| $30 \times 110$ | 764 677 | 450 592 | 98 922 | 57 094 | 8 180 375 | 0,0731 | 0,0914 | 0,0640 | 0,9534 |
| $30 \times 120$ | 1 161 997 | 966 211 | 168 360 | 83 719 | 12 166 618 | 0,0789 | 0,1060 | 0,0552 | 0,9532 |
| $30 \times 130$ | 1 804 238 | 1 327 980 | 276 109 | 132 438 | 18 395 540 | 0,0877 | 0,0987 | 0,0575 | 0,9606 |
| $30 \times 140$ | 2 538 168 | 2 091 863 | 489 797 | 181 218 | 27 735 580 | 0,0914 | 0,1108 | 0,0510 | 0,9619 |
| $30 \times 150$ | 4 388 952 | 3 485 716 | 952 281 | 307 140 | 44 583 908 | 0,0948 | 0,0999 | 0,0553 | 0,9622 |

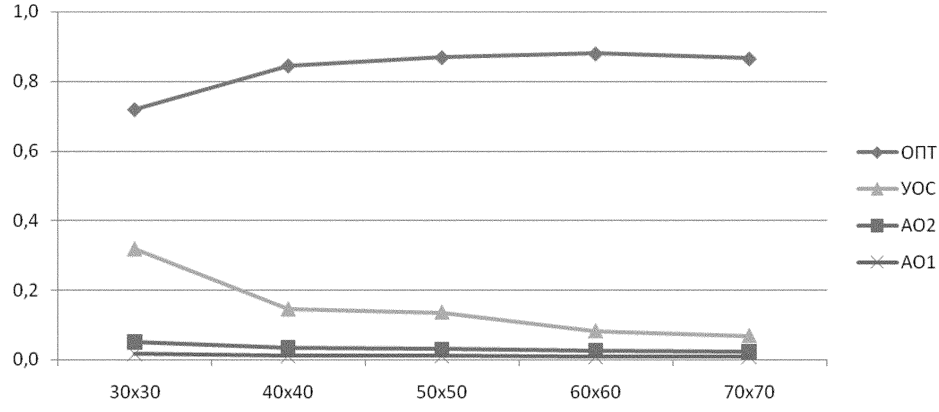



Рис. 3. Статистическая эффективность, случай $m = n$

Таблица 3. Случай $m = n$

| Размер матрицы $m \times n$ | Среднее время счета в миллисекундах | | | | Среднее число покрытий | Статистическая эффективность (СЭ) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | АО1 | АО2 | УОС | ОПТ | | АО1 | АО2 | УОС | ОПТ |
| $30 \times 30$ | 1 248 | 419 | 63 | 78 | 8 246 | 0,0181 | 0,0508 | 0,3187 | 0,7192 |
| $40 \times 40$ | 19 365 | 5 099 | 859 | 1 016 | 65 726 | 0,0126 | 0,0347 | 0,1454 | 0,8449 |
| $50 \times 50$ | 190 288 | 42 292 | 6 469 | 6 578 | 369 631 | 0,0112 | 0,0298 | 0,1362 | 0,8688 |
| $60 \times 60$ | 1 232 725 | 255 064 | 35 797 | 34 000 | 1 556 177 | 0,0089 | 0,0267 | 0,0823 | 0,8796 |
| $70 \times 70$ | 9 566 964 | 1 529 775 | 218 438 | 157 735 | 6 562 676 | 0,0096 | 0,0227 | 0,0686 | 0,8651 |

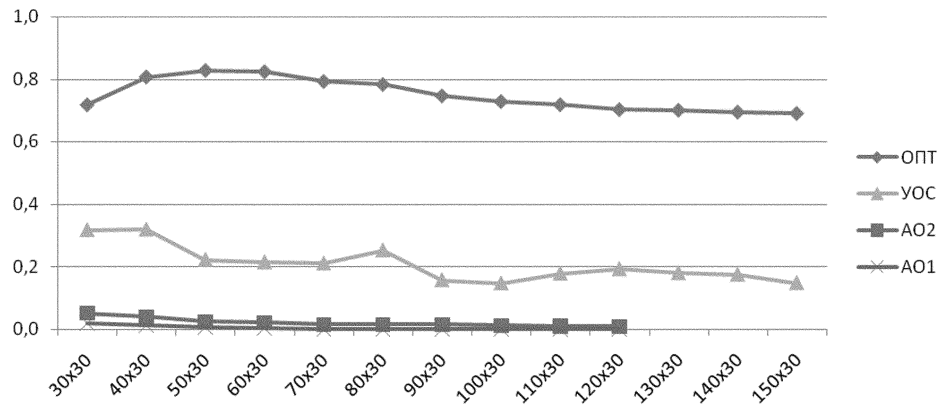



Рис. 4. Статистическая эффективность, случай $m > n$

Таблица 4. Случай $m > n$

| Размер матрицы $m \times n$ | Среднее время счета в миллисекундах | | | | Среднее число покрытий | Статистическая эффективность (СЭ) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | АО1 | АО2 | УОС | ОПТ | | АО1 | АО2 | УОС | ОПТ |
| $40 \times 30$ | 5 227 | 853 | 125 | 187 | 12 847 | 0,0112 | 0,0397 | 0,3203 | 0,8078 |
| $50 \times 30$ | 14 031 | 1 596 | 281 | 375 | 17 103 | 0,0055 | 0,0238 | 0,2228 | 0,8284 |
| $60 \times 30$ | 36 379 | 3 294 | 625 | 844 | 24 868 | 0,0029 | 0,0201 | 0,2149 | 0,8246 |
| $70 \times 30$ | 71 276 | 4 407 | 735 | 922 | 29 747 | 0,0013 | 0,0134 | 0,2111 | 0,7933 |
| $80 \times 30$ | 179 136 | 7 735 | 937 | 1 353 | 37 570 | 0,0016 | 0,0139 | 0,2532 | 0,7838 |
| $90 \times 30$ | 313 266 | 11 232 | 1 750 | 2 313 | 40 845 | 0,0010 | 0,0133 | 0,1576 | 0,7475 |
| $100 \times 30$ | 503 811 | 13 484 | 1 922 | 2 650 | 48 234 | 0,0008 | 0,0103 | 0,1471 | 0,7296 |
| $110 \times 30$ | 925 864 | 20 691 | 2 641 | 3 941 | 61 205 | 0,0005 | 0,0097 | 0,1784 | 0,7192 |
| $120 \times 30$ | 1 782 056 | 25 458 | 3 171 | 4 625 | 66 148 | 0,0004 | 0,0083 | 0,1928 | 0,7037 |
| $130 \times 30$ | | 31 877 | 3 818 | 5 519 | 69 590 | | | 0,1800 | 0,7018 |
| $140 \times 30$ | | 38 754 | 4 642 | 6 688 | 75 480 | | | 0,1745 | 0,6955 |
| $150 \times 30$ | | 44 654 | 4 875 | 7 109 | 80 129 | | | 0,1483 | 0,6914 |

Из приведенных таблиц видно, что применение алгоритма ОПТ в случаях **а)**, **б)** и **в)** позволяет существенно сократить затрачиваемое на решение задачи время.

Заметим, что в случае **а)** (табл. 1) наблюдается постепенное сокращение разрыва во времени счета между алгоритмами ОПТ и АО1, уменьшается также разница между показателями статистической эффективности алгоритмов АО1, АО2 и ОПТ. Следовательно, можно предположить, что в случае, когда $n$ имеет больший порядок роста, чем $m$, то при достаточно большом $n$ алгоритм АО1 за счет меньшей вычислительной сложности шага будет затрачивать на решение задачи меньше времени, чем алгоритм ОПТ.

Аналогичная картина (табл. 4) наблюдается при сравнении алгоритмов ОПТ и УОС в случае **г)** (за счет меньшей вычислительной сложности шага алгоритм УОС работает быстрее).

Для сравнения алгоритмов ОПТ+ и УОС+ была проведена серия экспериментов на случайных целочисленных матрицах размера $m \times n$, полученных с помощью $mn$-кратного обращения к датчику случайных чисел с равновероятным появлением чисел $\{0, 1, \ldots, k-1\}$ при $k = 2$ и $k = 5$. Для каждой тройки $(m, n, k)$ обсчитывалось по 20 матриц.

Алгоритмы ОПТ+ и УОС+ сравнивались в случаях **а)** $m < n$ и **б)** $m = n$.

Результаты счета приведены в табл. 5, 6, 7, 8.

Таблица 5. Случай $m < n$, $k = 2$

| Размер матрицы $m \times n$ | Среднее число покрытий | Среднее время счета в миллисекундах | | Отношение времен счета | Число покрытий в миллисекунду | |
|---|---|---|---|---|---|---|
| | | ОПТ+ | УОС+ | ОПТ+/УОС+ | ОПТ+ | УОС+ |
| 30 × 40 | 1 122 003 | 15 563 | 15 594 | 0,9980 | 72 | 72 |
| 30 × 50 | 4 065 542 | 52 579 | 66 543 | 0,7902 | 77 | 61 |
| 30 × 60 | 11 333 204 | 143 016 | 249 297 | 0,5737 | 79 | 45 |
| 30 × 70 | 25 863 238 | 317 969 | 559 907 | 0,5679 | 81 | 46 |
| 30 × 80 | 57 595 028 | 715 266 | 1 642 969 | 0,4353 | 81 | 35 |

Таблица 6. Случай $m < n$, $k = 5$

| Размер матрицы $m \times n$ | Среднее число покрытий | Среднее время счета в миллисекундах | | Отношение времен счета | Число покрытий в миллисекунду | |
|---|---|---|---|---|---|---|
| | | ОПТ+ | УОС+ | ОПТ+/УОС+ | ОПТ+ | УОС+ |
| 30 × 40 | 281 955 | 1 485 | 1 282 | 1,1583 | 190 | 220 |
| 30 × 50 | 571 476 | 2 985 | 2 672 | 1,1171 | 191 | 214 |
| 30 × 60 | 1 037 647 | 5 984 | 5 641 | 1,0608 | 173 | 184 |
| 30 × 70 | 1 715 233 | 10 703 | 10 453 | 1,0239 | 160 | 164 |
| 30 × 80 | 2 680 883 | 17 156 | 18 313 | 0,9368 | 156 | 146 |
| 30 × 90 | 3 987 492 | 27 500 | 29 562 | 0,9302 | 145 | 135 |
| 30 × 100 | 5 700 343 | 41 078 | 46 266 | 0,8879 | 139 | 123 |

Таблица 7. Случай $m = n$, $k = 2$

| Размер матрицы $m \times n$ | Среднее число покрытий | Среднее время счета в миллисекундах | | Отношение времен счета | Число покрытий в миллисекунду | |
|---|---|---|---|---|---|---|
| | | ОПТ+ | УОС+ | ОПТ+/УОС+ | ОПТ+ | УОС+ |
| 30 × 30 | 273 474 | 3 906 | 3 031 | 1,2887 | 70 | 90 |
| 40 × 40 | 2 479 136 | 50 797 | 61 125 | 0,8310 | 49 | 41 |
| 50 × 50 | 19 071 627 | 491 250 | 806 219 | 0,6093 | 39 | 24 |
| 60 × 60 | 101 582 632 | 3 119 797 | 5 361 000 | 0,5819 | 33 | 19 |

Таблица 8. Случай $m = n$, $k = 5$

| Размер матрицы $m \times n$ | Среднее число покрытий | Среднее время счета в миллисекундах | | Отношение времен счета | Число покрытий в миллисекунду | |
|---|---|---|---|---|---|---|
| | | ОПТ+ | УОС+ | ОПТ+/УОС+ | ОПТ+ | УОС+ |
| $30 \times 30$ | 113 878 | 625 | 593 | 1,0540 | 182 | 192 |
| $40 \times 40$ | 480 971 | 3 594 | 2 781 | 1,2923 | 134 | 173 |
| $50 \times 50$ | 1 562 271 | 16 687 | 13 094 | 1,2744 | 94 | 119 |
| $60 \times 60$ | 4 441 960 | 60 828 | 49 125 | 1,2382 | 73 | 90 |
| $70 \times 70$ | 11 254 475 | 177 156 | 160 766 | 1,1019 | 64 | 70 |
| $80 \times 80$ | 26 684 386 | 464 734 | 460 656 | 1,0089 | 57 | 58 |
| $90 \times 90$ | 57 582 190 | 1 069 562 | 1 210 156 | 0,8838 | 54 | 48 |
| $100 \times 100$ | 116 689 714 | 2 188 719 | 2 687 031 | 0,8145 | 53 | 43 |

Из приведенных таблиц видно, что применение алгоритма ОПТ+ позволяет сократить затрачиваемое на решение задачи время. Особенно эффективен этот алгоритм оказывается при $k = 2$. Результаты экспериментов показывают, что при $k = 2$ и $k = 5$ число тупиковых покрытий матрицы из $M_{mn}^k$ практически совпадает с числом неприводимых покрытий булевой матрицы размера $m \times kn$ с вероятностью появления 1, равной $1 - 1/k$. Данный факт подтверждает теоретические выводы о том, что число неприводимых покрытий матрицы $L^*$, содержащих родственные столбцы, невелико по сравнению с числом тупиковых покрытий матрицы $L$ (см. утверждение 3).

## СПИСОК ЛИТЕРАТУРЫ


1. Дюкова Е. В. Об одном алгоритме построения тупиковых тестов для бинарных таблиц // Сборник работ по дискретной математике. Вып. 1. — М.: ВЦ АН СССР, 1976. — С. 167–185.
2. Дюкова Е. В. Об асимптотически оптимальном алгоритме построения тупиковых тестов // ДАН СССР. — 1977. — Т. 233, № 4. — С. 527–530.
3. Дюкова Е. В. Асимптотически оптимальные тестовые алгоритмы в задачах распознавания // Проблемы кибернетики. Вып. 39.— М.: Наука, 1982. — С. 165–199.
4. Дюкова Е. В. О сложности реализации некоторых процедур распознавания // Ж. вычисл. матем. и матем. физ. — 1987. — Т. 27, № 1. — С. 114–127.
5. Дюкова Е. В. Алгоритмы распознавания типа Кора: сложность реализации и метрические свойства // Распознавание, классификация, прогноз (математические методы и их применение). Вып. 2. — М.: Наука, 1989. — С. 99–125.
6. Дюкова Е. В. О сложности реализации дискретных (логических) процедур распознавания // Ж. вычисл. матем. и матем. физ. — 2004. — Т. 44, № 3. — С. 550–572.
7. Дюкова Е. В., Журавлёв Ю. И. Дискретный анализ признаковых описаний в задачах распознавания большой размерности // Ж. вычисл. матем. и матем. физ. — 2000. — Т. 40, № 8. — С. 1264–1278.
8. Дюкова Е. В., Инякин А. С. О процедурах классификации, основанных на построении покрытий классов // Ж. вычисл. матем. и матем. физ. — 2003. — Т. 43, № 12. — С. 1910–1921.
9. Дюкова Е. В., Инякин А. С. О сложности решения задачи построения неприводимых покрытий булевой матрицы. — М.: ВЦ РАН, 2006.
10. Дюкова Е. В., Инякин А. С. Построение неприводимых покрытий булевой матрицы с полиномиальной задержкой // Доклады РАН. — 2007. — Т. 413, № 5. — С. 596–598.
11. Инякин А. С. Об одном алгоритме поиска тупиковых $\sigma$-покрытий // Тез. докл. Междунар. конф. «Интеллектуализация обработки информации». — Симферополь, 2002. — С. 47–48.
12. Инякин А. С. Алгоритмы поиска неприводимых покрытий булевых матриц. — М.: ВЦ РАН, 2004.

13. Чегис И. А., Яблонский С. В. Логические способы контроля электрических схем // Тр. МИАН СССР.— 1958.— Т. 51. — С. 270–360.
14. Яблонский С. В. Введение в дискретную математику. — М.: Наука, 1986.
15. Djukova E. V. Discrete (logical) recognition procedures: principles of construction, complexity of realization and basic models // Pattern recognition and image analysis. — 2003. — V. 13, No. 3. — P. 417–425.
16. Jonson D. S., Yannakakis M., Papadimitriou C. H. On general all maximal independent sets // Information processing letters. — 1988. — V. 27. — P. 119–123.

Поступило в редакцию 15 VIII 2007