МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2019 № 46

Зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций

Свидетельство о регистрации ПИ № ФС 77-33762 от 16 октября 2008 г.

Подписной индекс в объединённом каталоге «Пресса России» 38696

УЧРЕДИТЕЛЬ
Томский государственный университет

Адрес редакции и издателя: 634050, г. Томск, пр. Ленина, 36
E-mail: vestnik_pdm@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надёжности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Верстка *И. А. Панкратовой*

Подписано к печати 20.12.2019. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 14,78. Тираж 300 экз.
Заказ № 4180. Цена свободная. Дата выхода в свет 17.01.2020.

Отпечатано на оборудовании
Издательского Дома Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел.: 8(3822)53-15-28, 52-98-49

СОДЕРЖАНИЕ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

УДК 621.391:519.7+621.391.1:004.7

НЕКОТОРЫЕ СПОСОБЫ ПОСТРОЕНИЯ MDS-МАТРИЦ НАД КОНЕЧНЫМ ПОЛЕМ

О. Кой Пуэнте, Р. А. Де Ла Крус Хименес

ООО «Центр сертификационных исследований», г. Москва, Россия

Предлагаются новые методы построения MDS-матриц с использованием возведения в степень сопровождающих матриц многочленов над конечным полем. Изучается ряд неприводимых многочленов степени $t = 4$ и 6, сопровождающая матрица которых при возведении в соответствующую степень t является MDS-матрицей. Представлен новый метод построения MDS-матриц, ориентированных на низкоресурсную программную и аппаратную реализации.

Ключевые слова: *MDS-матрицы, сопровождающие матрицы многочленов, неприводимые многочлены, линейные регистры сдвига, конечные поля, XOR-сложность.*

DOI 10.17223/20710410/46/1

SOME METHODS FOR CONSTRUCTING MDS-MATRICES OVER FINITE FIELD

O. Coy Puente, R. A. De La Cruz Jiménez

Certification Research Center, Moscow, Russia

E-mail: o.coypuente@gmail.com, djr.antonio537@gmail.com

In this work, we propose new methods for constructing MDS-matrices over finite field by using recursive ones. For some element $\beta \in \mathrm{GF}(2^n)$ and naturals numbers s and k, we study polynomials of the form $x^4 + \beta^k x^3 + \beta x^2 + \beta^k x + 1$ and $x^6 + \beta^s x^5 + \beta^2 x^4 + \beta x^3 + \beta^2 x^2 + \beta^s x + 1$, for which, when $t = 4, 6$, the t-th power of it's companion matrices yields MDS-matrices with irreducible characteristic polynomial. Also, for some finite field elements β and γ, we have found MDS-matrices of the form $\mathcal{M}^4_{(\beta,\gamma)} = (\beta \cdot \mathcal{I}_{4,4} \oplus \gamma \cdot \mathcal{J}_{4,4} \oplus \mathcal{H}_{4,4})^4$, where for appropriate (4×4)-binary matrices $\mathcal{I}_{4,4}, \mathcal{J}_{4,4}, \mathcal{H}_{4,4}$ the resulting linear mappings can be simplified by some special schemes, very attractive for the so-called lightweight cryptography. The multiplication of any vector by the matrices obtained in the paper can be represented by some circuits which improve the cost of this operation implementation in terms of bitwise XOR's.

Keywords: *MDS-matrices, companion matrices, irreducible polynomials, LFSR, finite field, lightweight cryptography, XOR-count.*

Введение

MDS-матрицы часто используются для реализации линейного слоя алгоритмов блочного шифрования и хеш-функций с целью наилучшего рассеивания входных битов при выполнении требований к шифрам, определенных К. Шенноном [1].

Задача нахождения новых методов построения таких матриц, оптимальных с точки зрения реализации, оказывается довольно сложной. В данной работе представлены новые конструкции MDS-матриц на основе сопровождающих матриц многочленов степени $t = 4$ и 6 над полем $\mathrm{GF}(2^8)$. Они реализуются с использованием линейных регистров сдвига на векторах из множества $\mathrm{GF}(2^8)^t$. Эти конструкции ориентированы на использование в низкоресурсной криптографии.

1. Обзор известных результатов

Пусть $P = \mathrm{GF}(2^n) = \mathrm{GF}(2)[x]/g(x)$ — конечное поле из 2^n элементов, где $g(x)$ — неприводимый многочлен степени n над полем $\mathrm{GF}(2)$. Множество всех вектор-строк длины t над полем P обозначим через P^t, множество всех матриц размера $n \times n$ над полем P — через $P_{n,n}$, а множество всех обратимых матриц над полем P — через $P^*_{n,n}$.

1.1. MDS-матрицы

Определение 1 [2]. Показатель рассеивания ρ матрицы $A \in P_{t,t}$ определяется равенством

$$\rho(A) = \min_{\mathbf{a} \neq \mathbf{0}} \{\mathrm{w}(\mathbf{a}) + \mathrm{w}(\mathbf{a}A)\},$$

где $\mathrm{w}(\mathbf{a})$ — вес Хэмминга вектора $\mathbf{a} \in P^t$, т. е. количество его ненулевых элементов.

Определение 2 [2, 3]. Матрица $A \in P_{t,t}$ называется MDS-матрицей, если $\rho(A) = t + 1$.

Лемма 1 [4]. Пусть все элементы матрицы A^{-1} ненулевые, где A^{-1} — обратная к матрице $A \in P^*_{t,t}$. Тогда все подматрицы размера $(t-1) \times (t-1)$ матрицы A принадлежат множеству $P^*_{t-1,t-1}$.

Лемма 2 [4]. Матрица A является MDS-матрицей тогда и только тогда, когда все её квадратные подматрицы являются обратимыми матрицами над полем P.

Лемма 3 [4]. Матрица $A \in P_{4,4}$ является MDS-матрицей тогда и только тогда, когда все элементы её обратной матрицы ненулевые и все её подматрицы размера 2×2 являются обратимыми матрицами.

Определение 3 [3]. Пусть $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} + x^t \in P[x]$. Матрица $S_f \in P_{t,t}$, определённая равенством

$$S_f = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_0 & a_1 & a_2 & \cdots & a_{t-1} \end{pmatrix},$$

называется сопровождающей матрицей многочлена $f(x)$.

1.2. XOR-сложность

Способ восприятия и оценки сложности реализации линейного слоя развивался в течение какого-то времени. Было распространено мнение, что конечные элементы

поля с малым весом Хэмминга имеют более низкую сложность аппаратной реализации. В [5] авторы предложили оценивать сложность реализации путём подсчёта количества вентилей XOR, необходимых для умножения элементов поля. Они также показали, что, в отличие от распространённого мнения, элементы с более высоким весом Хэмминга также могут иметь низкую сложность реализации. Используем эту новую характеристику для расчёта сложности реализации линейного слоя.

Определение 4 [5]. XOR-сложность элемента $\alpha \in P$ — это наибольшее количество операций XOR, необходимых для реализации умножения α на произвольный элемент $\beta \in P$.

Пример 1 [5]. Пусть $\mathrm{GF}(2^3) = \mathrm{GF}(2)[x]/(x^3 + x + 1)$ и α — корень многочлена $x^3 + x + 1$. Рассмотрим $\{1, \alpha, \alpha^2\}$ — базис пространства $\mathrm{GF}(2^3)$ над полем $\mathrm{GF}(2)$. Умножение элемента $\alpha^4 = \alpha \oplus \alpha^2$ на произвольный элемент $\beta = b_0 \oplus b_1\alpha \oplus b_2\alpha^2$, где $b_i \in \mathrm{GF}(2)$, $i = 0, 1, 2$, имеет вид

$$(b_0 \oplus b_1\alpha \oplus b_2\alpha^2)(\alpha \oplus \alpha^2) = (b_0 \oplus b_2) \oplus (b_0 \oplus b_1)\alpha \oplus (b_0 \oplus b_1 \oplus b_2)\alpha^2.$$

Элемент $\alpha^4 \cdot \beta$ можно отождествить с упорядоченным набором из $\mathrm{GF}(2)^3$ его координат

$$(b_0 \oplus b_2, b_0 \oplus b_1, b_0 \oplus b_1 \oplus b_2),$$

в котором есть четыре операции XOR. Тогда XOR-сложность элемента α^4 равна 4.

Будем обозначать XOR-сложность элемента $\alpha \in \mathrm{GF}(2^n)$ как $\mathsf{XOR}(\alpha)$. Нетрудно проверить, что $\mathsf{XOR}(0) = \mathsf{XOR}(1) = 0$. XOR-сложность строки с номером i матрицы $M = (m_{i,j})$ размера $t \times t$ можно найти по формуле [5]

$$\sum_{j=1}^{t} \mathsf{XOR}(m_{i,j}) + (l_i - 1)n,$$

где l_i — количество ненулевых элементов в i-й строке. Тогда можно определить XOR-сложность любой матрицы $M = (m_{i,j}) \in \mathrm{GF}(2^n)_{t,t}$ по формуле

$$\mathsf{XOR}(M) = \sum_{i=1}^{t} \sum_{j=1}^{t} \mathsf{XOR}(m_{i,j}) + n \sum_{i=1}^{t} (l_i - 1).$$

Пусть $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{t-1}x^{t-1} + x^t$ и $\{c_1, \ldots, c_s\}$ — множество всех различных ненулевых коэффициентов многочлена $f(x)$. В [6] показано, что для сопровождающей матрицы S_f многочлена $f(x)$ верны следующие равенства:

$$\mathsf{XOR}(S_f) = \sum_{j=1}^{s} \mathsf{XOR}(c_j) + (l_f - 1)n; \tag{1}$$

$$\mathsf{XOR}(S_f^k) = k \cdot \mathsf{XOR}(S_f), \quad k \in \mathbb{N}, \tag{2}$$

где l_f — количество ненулевых элементов в последней строке матрицы S_f.

2. MDS-матрицы, построенные с использованием сопровождающих матриц многочленов

В работе [3] авторы осуществляли перебор элементов $a_0, a_1, a_2, \ldots, a_{t-1}$ таким образом, чтобы S_f^t была MDS-матрицей. В данной работе мы применим другой подход.

2.1. Способы построения MDS-матриц размера 4×4

Предложим метод построения MDS-матриц размера 4×4 с характеристическим многочленом

$$f(x) = x^4 + \beta^k x^3 + \beta x^2 + \beta^k x + 1 \in P[x], \tag{3}$$

где $\beta \in \mathrm{GF}(2^n)$; $k \in \mathbb{N}$.

Теорема 1. Пусть $P = \mathrm{GF}(2^n)$, $n \in \mathbb{N}$, $n \geqslant 4$, $f(x) \in P[x]$ — многочлен вида (3). Пусть $\beta \in P \backslash \{0, 1\}$ и $k \in \{2, \ldots, 2^n - 2\}$ подобраны так, что

$$\begin{aligned}
k^2(8k^2+7) &\not\equiv k(14k^2+1) \pmod{\mathrm{ord}(\beta)},\\
\beta &\neq (\beta^k \oplus 1)^2,\\
\beta &\neq (\beta^k \oplus \beta \oplus 1)^2,\\
\beta &\neq (\beta^{k-1} \oplus \beta^{-1} \oplus 1)^2,\\
\beta &\neq \beta^{2k+1} \oplus \beta^2 \oplus 1,\\
\beta &\neq \beta^k(\beta^k \oplus 1),\\
\beta &\neq (\beta^k \oplus \beta \oplus 1)^2 \beta^{-k},\\
\beta &\neq (\beta^k \oplus \beta \oplus 1)\beta^{k+1},\\
\beta^{2k+1} &\neq (\beta^k \oplus 1)^2\\
\beta^{2k+1} &\neq (\beta^{2k} \oplus \beta \oplus 1)^2,\\
\beta^{4k} &\neq (\beta \oplus 1)^2(\beta^{2k} \oplus \beta).
\end{aligned}$$

Тогда S_f^4 — MDS-матрица.

Доказательство. Сопровождающая матрица многочлена $f(x)$ имеет вид

$$S_f = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & \beta^k & \beta & \beta^k \end{pmatrix}.$$

Несложно проверить, что

$$S_f^{-1} = \begin{pmatrix} \beta^k & \beta & \beta^k & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$S_f^4 = \begin{pmatrix} 1 & \beta^k & \beta & \beta^k \\ \beta^k & 1 \oplus \beta^{2k} & \beta^k \oplus \beta^{1+k} & \beta \oplus \beta^{2k} \\ \beta \oplus \beta^{2k} & \beta^k \oplus \beta^{1+k} \oplus \beta^{3k} & 1 \oplus \beta^2 \oplus \beta^{2k} \oplus \beta^{2k+1} & \beta^k \oplus \beta^{3k} \\ \beta^k \oplus \beta^{3k} & \beta \oplus \beta^{4k} & \beta^k \oplus \beta^{3k} \oplus \beta^{1+3k} & 1 \oplus \beta^2 \oplus \beta^{1+2k} \oplus \beta^{4k} \end{pmatrix}.$$

Тогда получим, что матрица S_f^4 и её обратная отличаются только перестановкой строк и столбцов. Из леммы 3 следует, что достаточно показать, что элементы $s'_{i,j}$ матрицы S_f^4 ненулевые для любых $i, j \in \{1, \ldots, 4\}$, так же как и все миноры размера 2×2. Очевидно, что $s'_{1,j} \neq 0$, $j = 1, \ldots, 4$, и $s'_{2,1} \neq 0$. Из условия

$$k^2(8k^2+7) \not\equiv k(14k^2+1) \bmod (\mathrm{ord}(\beta))$$

получим, что $\mathrm{ord}(\beta) \nmid (8k^4 - 14k^3 + 7k^2 - k)$. Так как

$$8k^4 - 14k^3 + 7k^2 - k = k(k-1)(2k-1)(4k-1),$$

имеем $\mathrm{ord}(\beta) \nmid k(k-1)(2k-1)(4k-1)$, тогда $s'_{2,2} \neq 0$, $s'_{2,3} \neq 0$, $s'_{2,4} \neq 0$, $s'_{3,1} \neq 0$, $s'_{3,4} \neq 0$, $s'_{4,1} \neq 0$ и $s'_{4,2} \neq 0$. Для остальных элементов рассмотрим такие утверждения:

1) $s'_{3,2} \neq 0$ и $s'_{3,3} \neq 0$ тогда и только тогда, когда $\beta \neq (\beta^k \oplus 1)^2$;
2) $s'_{4,3} \neq 0$ тогда и только тогда, когда $\beta^{2k+1} \neq (\beta^k \oplus 1)^2$;
3) $s'_{4,4} \neq 0$ тогда и только тогда, когда $\beta^{2k+1} \neq (\beta^{2k} \oplus \beta \oplus 1)^2$.

Докажем, что все миноры размера 2×2 ненулевые. Любая матрица размера 4×4 имеет 36 миноров размера 2×2. Построим семейство (мультимножество) Y, состоящее из всех таких миноров:

$$
\begin{aligned}
Y = \{\ & 1, && \beta^k, && \beta,\\
& \beta \oplus \beta^{2k}, && \beta^k \oplus \beta^{1+k}, && \beta^2 \oplus \beta^{2k},\\
& \beta^k, && 1 \oplus \beta^{2k}, && \beta^k \oplus \beta^{1+k},\\
& \beta^k \oplus \beta^{3k} \oplus \beta^{1+k}, && \beta^{1+2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{1+k} \oplus \beta^{2+k},\\
& \beta \oplus \beta^{2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{1+k}, && 1 \oplus \beta^2 \oplus \beta^{2k} \oplus \beta^{1+2k},\\
& \beta^2 \oplus \beta^{2k} \oplus \beta^{4k}, && \beta^k \oplus \beta^{1+k} \oplus \beta^{2+k} \oplus \beta^{1+3k}, && \beta \oplus \beta^3 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k},\\
& \beta, && \beta^k \oplus \beta^{1+k}, && \beta^2 \oplus \beta^{2k},\\
& 1 \oplus \beta^2 \oplus \beta^{2k} \oplus \beta^{1+2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{1+k} \oplus \beta^{2+k}, && \beta \oplus \beta^3,\\
& \beta^k \oplus \beta^{1+k}, && \beta^{1+2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{1+k} \oplus \beta^{2+k},\\
& \beta^k \oplus \beta^{1+k} \oplus \beta^{2+k} \oplus \beta^{1+3k}, && 1 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{2+k} \oplus \beta^{3+k},\\
& \beta^2 \oplus \beta^{2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{1+k} \oplus \beta^{2+k}, && \beta \oplus \beta^3,\\
& \beta \oplus \beta^3 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{2+k} \oplus \beta^{3+k}, && 1 \oplus \beta^4 \oplus \beta^{4k} \oplus \beta^{2+2k}\}.
\end{aligned}
$$

Удалив совпадающие элементы, построим множество M с элементами семейства Y:

$$
\begin{aligned}
M = \{\ & 1, && \beta^k, && \beta,\\
& \beta \oplus \beta^{2k}, && \beta^k \oplus \beta^{1+k}, && \beta^2 \oplus \beta^{2k},\\
& 1 \oplus \beta^{2k}, && \beta^{1+2k}, && \beta^k \oplus \beta^{3k} \oplus \beta^{1+k} \oplus \beta^{2+k},\\
& 1 \oplus \beta^2 \oplus \beta^{2k} \oplus \beta^{1+2k}, && \beta^2 \oplus \beta^{2k} \oplus \beta^{4k}, && \beta^k \oplus \beta^{1+k} \oplus \beta^{2+k} \oplus \beta^{1+3k},\\
& \beta \oplus \beta^3 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k}, && \beta \oplus \beta^3, && 1 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k},\\
& \beta^k \oplus \beta^{3k} \oplus \beta^{2+k} \oplus \beta^{3+k}, && 1 \oplus \beta^4 \oplus \beta^{4k} \oplus \beta^{2+2k}\}. &&
\end{aligned}
$$

Для элементов множества M, совпадающих с элементами матрицы S^4_f, уже показано, что они ненулевые. Доказательство теоремы вытекает из справедливости следующих соотношений:

1) $\beta^2 \oplus \beta^{2k} \neq 0$ тогда и только тогда, когда $\mathrm{ord}(\beta) \nmid (k-1)$;
2) очевидно, что $\beta^{1+2k} \neq 0$ и $\beta \oplus \beta^3 \neq 0$;
3) $\beta^k \oplus \beta^{3k} \oplus \beta^{1+k} \oplus \beta^{2+k} \neq 0$ тогда и только тогда, когда $\beta \neq (\beta^k \oplus \beta \oplus 1)^2$;
4) $1 \oplus \beta^2 \oplus \beta^{2k} \oplus \beta^{1+2k} \neq 0$ тогда и только тогда, когда $\beta \neq (\beta^k \oplus 1)^2$;
5) $\beta^2 \oplus \beta^{2k} \oplus \beta^{4k} \neq 0$ тогда и только тогда, когда $\beta \neq \beta^k(\beta^k \oplus 1)$;
6) $\beta^k \oplus \beta^{1+k} \oplus \beta^{2+k} \oplus \beta^{1+3k} \neq 0$ тогда и только тогда, когда $\beta \neq \beta^{2k+1} \oplus \beta^2 \oplus 1$;
7) $\beta \oplus \beta^3 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k} \neq 0$ тогда и только тогда, когда $\beta^{4k} \neq (\beta \oplus 1)^2(\beta^{2k} \oplus \beta)$;
8) $1 \oplus \beta^{2k} \oplus \beta^{4k} \oplus \beta^{2+2k} \neq 0$ тогда и только тогда, когда $\beta \neq (\beta^k \oplus \beta \oplus 1)\beta^{k+1}$;
9) $\beta^k \oplus \beta^{3k} \oplus \beta^{2+k} \oplus \beta^{3+k} \neq 0$ тогда и только тогда, когда $\beta \neq (\beta^{k-1} \oplus \beta^{-1} \oplus 1)^2$;
10) $1 \oplus \beta^4 \oplus \beta^{4k} \oplus \beta^{2+2k} \neq 0$ тогда и только тогда, когда $\beta \neq (\beta^k \oplus \beta \oplus 1)^2\beta^{-k}$.

Теорема доказана. ■

Следствие 1. Пусть $P = \mathrm{GF}(2^n)$, $n \in \mathbb{N}$, $n \geqslant 4$, $f(x) = x^4 + \beta^2 x^3 + \beta x^2 + \beta^2 x + 1$ и элемент $\beta \in P \backslash \{0, 1\}$ такой, что

$$\begin{aligned}
\operatorname{ord}(\beta) &\nmid 21,\\
\beta &\neq (\beta\oplus 1)^3,\\
\beta &\neq (\beta\oplus 1)^4,\\
\beta^3 &\neq \beta\oplus 1,\\
\beta^3 &\neq (\beta\oplus 1)^2(\beta^5\oplus 1),\\
\beta^5 &\neq (\beta\oplus 1)^4,\\
\beta^5 &\neq (\beta\oplus 1)(\beta^6\oplus 1).
\end{aligned}$$

Тогда S_f^4 — MDS-матрица.

2.2. MDS-матрицы размера 6×6

Предложим метод построения MDS-матриц размера 6×6 над полем $P=\mathrm{GF}(2^8)$ с использованием сопровождающих матриц многочлена

$$f(x)=x^6+\beta^4x^5+\beta^2x^4+\beta x^3+\beta^2x^2+\beta^4x+1\in P[x]. \tag{4}$$

Теорема 2. Пусть $P=\mathrm{GF}(2^8)$, $f(x)\in P[x]$ — многочлен вида (4) и элемент $\beta\in$ $\in P\backslash\{0,1\}$ подобран так, что

$$\begin{aligned}
1 &\neq (\beta\oplus 1)^3(\beta^6(\beta\oplus 1)^2\oplus\beta)^2,\\
(\beta\oplus 1)^2 &\neq \beta^5(\beta^5\oplus 1),\\
(\beta\oplus 1)^{11} &\neq \beta^2(\beta^2\oplus\beta\oplus 1),\\
\beta(\beta\oplus 1)^8 &\neq (\beta^5\oplus 1)(\beta^6\oplus 1),\\
\beta^2(\beta(\beta\oplus 1)^2\oplus 1)^2 &\neq (\beta^5\oplus 1)(\beta^6\oplus\beta^3\oplus 1)^2.
\end{aligned}$$

Тогда S_f^6 — MDS-матрица.

Доказательство. Проводится аналогично доказательству теоремы 1. ■

Предложим метод построения MDS-матриц размера 6×6 над полем $P=\mathrm{GF}(2^8)$, представимых в виде шестой степени сопровождающей матрицы многочлена

$$f(x)=x^6+\beta^7x^5+\beta^2x^4+\beta x^3+\beta^2x^2+\beta^7x+1\in P[x]. \tag{5}$$

Теорема 3. Пусть $P=\mathrm{GF}(2^8)$ и $f(x)\in P[x]$ — многочлен вида (5). Тогда если существует элемент $\beta\in P\backslash\{0,1\}$, такой, что

$$(\beta\oplus 1)^{10}(\beta(\beta\oplus 1)^2\oplus 1)^8 \neq (\beta(\beta\oplus 1))^4(\beta^6\oplus 1),$$

то S_f^6 является MDS-матрицей.

Доказательство. Справедливость теоремы проверяется аналогично доказательству теоремы 1. ■

Рассмотрим наиболее интересный случай, когда многочлены вида (3), (4) и (5) являются неприводимыми над соответствующими полями. В этом случае нельзя выделить классы слабых ключей, основанные на наличии у линейного преобразования алгоритма блочного шифрования инвариантных подпространств [7]. В табл. 1 приведены несколько примеров таких многочленов над полем $\mathrm{GF}(2^n)=\mathrm{GF}(2)[x]/g(x)$, где $\deg(g)=n$. Элементы поля записаны в шестнадцатеричном виде. Например, для элемента $\beta\in\mathrm{GF}(2^8)$, такого, что $\beta=x^7+x^5+x^2+1$, используется запись a5.

Таблица 1

Примеры неприводимых многочленов степени t, сопровождающие матрицы которых при возведении в степень t являются MDS-матрицами

n	$g(x)$	t	$f(x)$	S_f^t
4	x^4+x+1	4	$x^4+\mathsf{c}x^3+8x^2+\mathsf{c}x+1$	$\begin{pmatrix} 1 & \mathsf{c} & 8 & \mathsf{c} \\ \mathsf{c} & \mathsf{e} & 6 & 7 \\ 7 & \mathsf{e} & 3 & 4 \\ 4 & 2 & 8 & 6 \end{pmatrix}$
8	$x^8+x^7+x^6+x+1$	4	$x^4+4x^3+2x^2+4x+1$	$\begin{pmatrix} 1 & 4 & 2 & 4 \\ 4 & 11 & \mathsf{c} & 12 \\ 12 & 4\mathsf{c} & 35 & 44 \\ 44 & \mathsf{c}1 & \mathsf{c}4 & \mathsf{e}6 \end{pmatrix}$
16	$x^{16}+x^5+x^3+x^2+1$	4	$x^4+112\mathsf{d}x^3+1\mathsf{c}x^2+112\mathsf{d}x+1$	$\begin{pmatrix} 1 & 112\mathsf{d} & 1\mathsf{c} & 112\mathsf{d} \\ 112\mathsf{d} & 297\mathsf{d} & \mathsf{ce}0\mathsf{c} & 2960 \\ 2960 & \mathsf{d}022 & 518\mathsf{a} & \mathsf{f}03 \\ \mathsf{f}03 & \mathsf{aa}21 & 6406 & \mathsf{d}2\mathsf{cb} \end{pmatrix}$
8	$x^8+x^7+x^6+x+1$	6	$x^6+13x^5+\mathsf{f}8x^4+\mathsf{a}3x^3+\mathsf{f}8x^2+13x+1$	$\begin{pmatrix} 1 & 13 & \mathsf{f}8 & \mathsf{a}3 & \mathsf{f}8 & 13 \\ 13 & \mathsf{c}7 & 83 & 7\mathsf{c} & 33 & 3\mathsf{e} \\ 3\mathsf{e} & 37 & \mathsf{b}3 & \mathsf{bb} & 8 & 17 \\ 17 & \mathsf{b}4 & \mathsf{c}1 & \mathsf{fe} & 4\mathsf{d} & 82 \\ 82 & \mathsf{a}5 & 1\mathsf{a} & 1\mathsf{d} & 50 & \mathsf{ff} \\ \mathsf{ff} & 6\mathsf{b} & 1\mathsf{b} & 15 & \mathsf{a}3 & \mathsf{b}9 \end{pmatrix}$
8	$x^8+x^7+x^6+x+1$	6	$x^6+\mathsf{ba}x^5+2\mathsf{a}x^4+\mathsf{b}6x^3+2\mathsf{a}x^2+\mathsf{ba}x+1$	$\begin{pmatrix} 1 & \mathsf{ba} & 2\mathsf{a} & \mathsf{b}6 & 2\mathsf{a} & \mathsf{ba} \\ \mathsf{ba} & 7\mathsf{b} & \mathsf{b}5 & 64 & \mathsf{b}9 & 50 \\ 50 & 9 & 8\mathsf{c} & 40 & 93 & \mathsf{a} \\ \mathsf{a} & 7\mathsf{e} & \mathsf{ce} & \mathsf{da} & 87 & \mathsf{bd} \\ \mathsf{bd} & \mathsf{d}0 & \mathsf{a}7 & 4 & 3 & 5\mathsf{d} \\ 5\mathsf{d} & 80 & 36 & 80 & \mathsf{e}2 & 3\mathsf{e} \end{pmatrix}$

3. MDS-отображения, построенные с использованием линейных регистров сдвига

Определение 5. Будем говорить, что линейное отображение $L : P^t \longrightarrow P^t$, заданное по правилу

$$L(\mathbf{a}) = \mathbf{a} \cdot A_\gamma(L),$$

является MDS-отображением, если $\rho(A_\gamma(L)) = t+1$, где $A_\gamma(L)$ — матрица линейного отображения L в фиксированном базисе γ пространства P^t.

В криптосистемах, ориентированных на низкоресурсную реализацию, существенное значение имеет XOR-сложность используемых криптографических примитивов. Нахождение MDS-отображений с небольшим значением данного параметра является актуальной задачей. Предложим несколько способов построения таких отображений.

3.1. Классы MDS-отображений множества $\mathrm{GF}(2^8)^4$

Предложим способ построения MDS-отображений над полем P с помощью многочленов вида $x^t+x+\beta$.

Теорема 4. Пусть $P=\mathrm{GF}(2^8)$, $f(x)=x^4+x+\beta \in P[x]$, $\mathrm{ord}(\beta) \nmid 9$, $(\beta(\beta^3 \oplus 1))^3 \neq 1$. Тогда матрица S_f^{22} является MDS-матрицей над P.

Доказательство. Справедливость теоремы проверяется по аналогии с доказательством теоремы 1. ■

Известно [8], что $f(S_f)=0$. Для многочлена $f(x)=x^4+x+\beta$ из теоремы 4 имеем $S_f^4 = S_f \oplus \beta \cdot E$, где E — единичная матрица размера 4×4. Пусть $E_\beta = \beta \cdot E$. Тогда верны следующие равенства:

$$S_f^{22} = S_f^2(S_f \oplus E_\beta)(S_f \oplus E_\beta)^4 = S_f^2(S_f \oplus E_\beta)(S_f^4 \oplus E_\beta^4) =$$

$$= S_f^2(S_f \oplus E_\beta)(S_f \oplus E_\beta \oplus E_\beta^4) = S_f^2(S_f \oplus E_\beta)(S_f \oplus E_{\beta(\beta^3 \oplus 1)}).$$

Рассмотрим отображение $\psi_\beta : P^4 \longrightarrow P^4$, определённое по правилу

$$\psi_\beta(\mathbf{a}) = \mathbf{a} \cdot E_\beta,$$

для любых $\beta \in P$, $\mathbf{a} \in P^4$. Тогда отображение $L_{f,\beta} : P^4 \longrightarrow P^4$, задаваемое равенством

$$L_{f,\beta}(\mathbf{a}) = \mathbf{a}\,(S_f^{22})^{\mathrm{T}},$$

является MDS-отображением. Пусть $\beta' = \beta(\beta^3 \oplus 1)$. Обозначим через F отображение, соответствующее линейному регистру сдвига с характеристическим многочленом $f(x)$, т. е. $F(\mathbf{a}) = \mathbf{a} \cdot S_f^{\mathrm{T}}$. Действие отображения $L_{f,\beta}$ на векторах из множества P^4 можно представить схемой рис. 1.

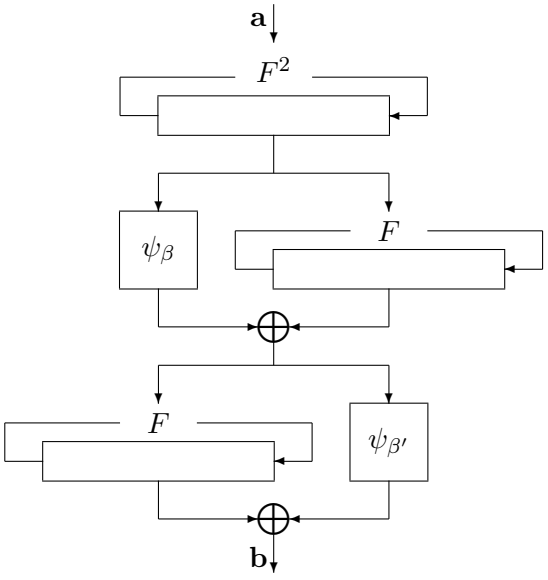


Рис. 1. Действие отображения $L_{f,\beta}$

Рассмотрим теперь MDS-отображения вида

$$\mathbf{a} \mapsto \mathbf{a}(S_f^{\top} \oplus E)^k, \quad \mathbf{a} \in P^t, \tag{6}$$

где S_f — сопровождающая матрица некоторого многочлена $f(x)$ степени t; E — единичная матрица размера $t \times t$. Пусть $f(x) = x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$ — унитарный многочлен степени 4 над полем P. С целью обеспечения эффективной реализации коэффициенты f_i выберем из множества $\{0, 1, \beta, \beta^2\}$. Построим следующие отображения для любого $\mathbf{a} \in P^4$:

Отображение $\mathcal{L}_{f,k}^4$ при $k = 3$
Пусть $\lambda_1(x) = x^4 + \beta^2x^3 + \beta x + \beta^2$. Тогда $\mathcal{L}_{\lambda_1,3}^4(\mathbf{a}) = \mathbf{a}\,(S_{\lambda_1}^{\mathrm{T}} \oplus E)^3$.

Отображение $\mathcal{L}_{f,k}^4$ при $k = 6$
Пусть $\lambda_2(x) = x^4 + \beta x^3 + \beta$. Тогда $\mathcal{L}_{\lambda_2,6}^4(\mathbf{a}) = \mathbf{a}\,(S_{\lambda_2}^{\mathrm{T}} \oplus E)^6$.

Отображение $\mathcal{L}_{f,k}^4$ при $k = 9$
Пусть $\lambda_3(x) = x^4 + x + \beta$. Тогда $\mathcal{L}_{\lambda_3,9}^4(\mathbf{a}) = \mathbf{a}\,(S_{\lambda_3}^{\mathrm{T}} \oplus E)^9$.

Пусть Λ_i — линейное преобразование, соответствующее линейному регистру сдвига с характеристическим многочленом $\lambda_i(x)$ для любого $i = 1, 2, 3$. Тогда действие отображения $\mathcal{L}^4_{\lambda_i,k}$ можно схематично представить рис. 2, где

$$(k_1, k_2) = \begin{cases} (k-i, i), & i = 1, 2, \\ (8, 1), & i = 3. \end{cases}$$

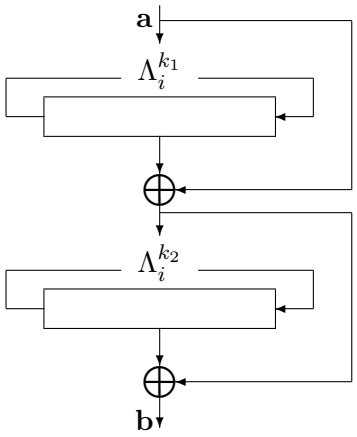


Рис. 2. Действие отображения $\mathcal{L}^4_{\lambda_i,k}$ при $i = 1, 2, 3$

Пусть $P = \mathrm{GF}(2^8)$, $D_1 = \{\beta \in P^* : \mathrm{ord}(\beta) \nmid 9,\ (\beta(\beta^3 \oplus 1))^3 \neq 1\}$. Следующая теорема доказывается аналогично теореме 1.

Теорема 5. Отображения $\mathcal{L}^4_{\lambda_1,3}$, $\mathcal{L}^4_{\lambda_2,6}$ и $\mathcal{L}^4_{\lambda_3,9}$ являются MDS-отображениями для любого $\beta \in D_1$.

3.2. Классы MDS-отображений множества $\mathrm{GF}(2^8)^6$

Пусть $P = \mathrm{GF}(2^8)$. Предложим методы построения MDS-отображений вида (6). Рассмотрим следующие отображения:

Отображение $\mathcal{L}^6_{f,k}$ при $k = 9$
Пусть $\phi_1(x) = x^6 + x^3 + \beta^2 x^2 + \beta$ и $\phi_2(x) = x^6 + \beta^2 x^5 + \beta^2 x^2 + \beta$. Тогда $\mathcal{L}^6_{\phi_1,9}(\mathbf{a}) = \mathbf{a}\,(S^{\mathrm{T}}_{\phi_1} \oplus E)^9$, $\mathcal{L}^6_{\phi_2,9}(\mathbf{a}) = \mathbf{a}\,(S^{\mathrm{T}}_{\phi_2} \oplus E)^9$.

Отображение $\mathcal{L}^6_{f,k}$ при $k = 10$
Пусть $\phi_3(x) = x^6 + \beta^2 x^5 + \beta x^4 + \beta$. Тогда $\mathcal{L}^6_{\phi_3,12}(\mathbf{a}) = \mathbf{a}\,(S^{\mathrm{T}}_{\phi_3} \oplus E)^{10}$.

Пусть Φ_i — линейное преобразование, соответствующее линейному регистру сдвига с характеристическим многочленом $\phi_i(x)$ для любого $i = 1, 2, 3$. Тогда действие отображения $\mathcal{L}^6_{\phi_i,k}$ при соответствующих значениях k можно схематично представить рис. 3, где $k' \equiv k \pmod 8$.

Рассмотрим следующие неприводимые многочлены степени 8 над полем $\mathrm{GF}(2)$:

$$\begin{aligned}
r_1(x) &= x^8 + x^4 + x^3 + x + 1, & r_2(x) &= x^8 + x^5 + x^4 + x^3 + 1,\\
r_3(x) &= x^8 + x^6 + x^5 + x + 1, & r_4(x) &= x^8 + x^6 + x^5 + x^4 + x^2 + x + 1,\\
r_5(x) &= x^8 + x^7 + x^3 + x + 1, & r_6(x) &= x^8 + x^7 + x^4 + x^3 + x^2 + x + 1,\\
r_7(x) &= x^8 + x^7 + x^5 + x + 1, & r_8(x) &= x^8 + x^7 + x^5 + x^3 + 1,\\
r_9(x) &= x^8 + x^7 + x^6 + x + 1, & r_{10}(x) &= x^8 + x^7 + x^6 + x^4 + x^2 + x + 1,\\
r_{11}(x) &= x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + 1, & r_{12}(x) &= x^8 + x^7 + x^6 + x^5 + x^2 + x + 1,\\
r_{13}(x) &= x^8 + x^7 + x^6 + x^5 + x^4 + x + 1, & r_{14}(x) &= x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1.
\end{aligned}$$

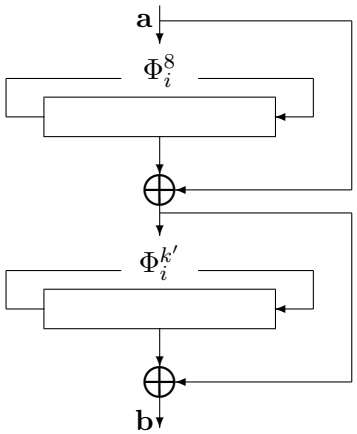


Рис. 3. Действие отображения $\mathcal{L}^6_{\phi_i,k}$ при $i = 1, 2, 3$

Обозначим через $W(r_i)$ множество всех корней многочлена $r_i(x)$ над полем $P = \mathrm{GF}(2^8)$, т. е. $W(r_i) = \{\beta \in P : r_i(\beta) = 0\}$. Следующая теорема доказывается аналогично теореме 1.

Теорема 6. Справедливы утверждения:

1) пусть $\Delta_1 = \{1, 2, 4, 5, 6, 7, 9, 11, 12\} \subset \mathbb{N}$, тогда для любого $\beta \in \bigcup\limits_{i\in\Delta_1} W(r_i)$ отображение $\mathcal{L}^6_{\phi_1,9}$ является MDS-отображением;
2) пусть $\Delta_2 = \{1, 5, 9, 14\} \subset \mathbb{N}$, тогда для любого $\beta \in \bigcup\limits_{i\in\Delta_2} W(r_i)$ отображение $\mathcal{L}^6_{\phi_2,9}$ является MDS-отображением;
3) пусть $\Delta_3 = \{1, 3, 6, 8, 9, 10, 11, 13\} \subset \mathbb{N}$, тогда для любого $\beta \in \bigcup\limits_{i\in\Delta_3} W(r_i)$ отображение $\mathcal{L}^6_{\phi_3,10}$ является MDS-отображением.

4. Построение MDS-матриц вида $(\beta \cdot \mathcal{I}_{4,4} \oplus \gamma \cdot \mathcal{J}_{4,4} \oplus \mathcal{H}_{4,4})^k$

Рассмотрим вопрос о построении MDS-матриц вида $\mathcal{M}^k_{(\beta,\gamma)}$, где

$$\mathcal{M}_{(\beta,\gamma)} = (\beta \cdot \mathcal{I}_{4,4} \oplus \gamma \cdot \mathcal{J}_{4,4} \oplus \mathcal{H}_{4,4});$$

$\mathcal{I}_{4,4}$, $\mathcal{J}_{4,4}$ и $\mathcal{H}_{4,4}$ — произвольные матрицы из $\mathrm{GF}(2)_{4,4}$; элементы β, γ не равны нулю одновременно и принадлежат множеству $\{0, \alpha, \alpha^2\}$, построенному с использованием примитивного элемента $\alpha \in P = \mathrm{GF}(2^8)$. Причиной последнего ограничения является стремление к эффективной реализации при малых значениях k линейного преобразования $\mathcal{M}^k_{(\beta,\gamma)}$. Заметим, что если большинство элементов матрицы $\mathcal{M}_{(\beta,\gamma)}$ равны нулю и среди ненулевых встречаются больше единиц, чем элементов β, γ, то при небольших k матрица $\mathcal{M}^k_{(\beta,\gamma)}$ представляет наибольший интерес.

Был осуществлен поиск на основе случайной генерации матриц $\mathcal{I}_{4,4}$, $\mathcal{J}_{4,4}$ и $\mathcal{H}_{4,4}$ с небольшим количеством единиц; в результате получены матрицы, которые имеют эффективную реализацию (табл. 2). В следующей теореме мы покажем, что эти матрицы являются MDS-матрицами над $P = \mathrm{GF}(2^8)$.

Пусть $p_1(x) = x^2 \oplus x \oplus 1$, $p_2(x) = x^3 \oplus x \oplus 1$, $p_3(x) = x^3 \oplus x^2 \oplus 1$, $p_4(x) = x^4 \oplus x^3 \oplus 1$, $p_5(x) = x^6 \oplus x^3 \oplus 1$, $p_6(x) = x^4 \oplus x^3 \oplus x^2 \oplus x \oplus 1$ — многочлены над полем P.

Теорема 7. Пусть α — примитивный элемент поля $P = \mathrm{GF}(2^8)$. Тогда верны следующие утверждения:

1) если $p_i(\alpha) \neq 0$, $i = 1, 2, 3, 4$, то $\mathcal{M}^4_{(\alpha,0)}$ является MDS-матрицей над P;
2) если $p_i(\alpha) \neq 0$, $i = 1, 2, 3, 4, 5$, то $\mathcal{M}^4_{(\alpha,\alpha^2)}$ является MDS-матрицей над P;

Таблица 2

Несколько матриц вида $(\beta \cdot \mathcal{I}_{4,4} \oplus \gamma \cdot \mathcal{J}_{4,4} \oplus \mathcal{H}_{4,4})^4$

(β,γ)	$\mathcal{M}^4_{(\beta,\gamma)}$	$\mathcal{I}_{4,4}$	$\mathcal{J}_{4,4}$	$\mathcal{H}_{4,4}$	$\mathcal{M}_{(\beta,\gamma)}$
$(\alpha,0)$	$\begin{pmatrix} \alpha & \alpha^2\oplus 1 & \alpha & \alpha \\ \alpha & \alpha^2\oplus\alpha & \alpha^2\oplus 1 & 1 \\ \alpha^2 & \alpha^3 & \alpha & \alpha^3\oplus\alpha \\ \alpha^2\oplus 1 & \alpha^3 & \alpha^2 & \alpha^2\oplus\alpha \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \end{pmatrix}$	—	$\begin{pmatrix} 0&1&0&0 \\ 0&0&1&1 \\ 0&0&0&1 \\ 1&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0 \\ 0&0&1&1 \\ 0&0&0&\alpha \\ 1&\alpha&0&0 \end{pmatrix}$
(α^2,α)	$\begin{pmatrix} \alpha^3 & 1 & \alpha^2\oplus\alpha & \alpha^2 \\ \alpha^3\oplus\alpha^2 & \alpha^3\oplus 1 & \alpha^2 & \alpha^2 \\ \alpha & \alpha & \alpha^3\oplus 1 & \alpha^4\oplus\alpha^3 \\ \alpha & \alpha^2\oplus\alpha & 1 & \alpha^3 \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&1&0 \\ 0&0&1&0 \\ 0&1&0&0 \\ 0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&1&0 \\ a&0&1&0 \\ 0&1&0&\alpha^2 \\ 0&1&0&0 \end{pmatrix}$
(α,α)	$\begin{pmatrix} \alpha & \alpha^2\oplus 1 & 1 & \alpha^2\oplus\alpha \\ 1 & \alpha^2\oplus\alpha\oplus 1 & \alpha & \alpha^2\oplus\alpha \\ \alpha^2\oplus 1 & \alpha^2\oplus 1 & \alpha^2\oplus\alpha\oplus 1 & \alpha \\ \alpha\oplus 1 & 1 & \alpha\oplus 1 & \alpha \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&1&0&0 \\ 0&0&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&1&0 \\ 1&0&1&0 \\ 0&1&0&0 \\ 0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 0&0&1&0 \\ 1&0&1&0 \\ 0&\alpha\oplus 1&0&\alpha \\ 0&1&0&0 \end{pmatrix}$

3) если $p_i(\alpha) \neq 0$, $i = 1, 2, 3, 6$, то $\mathcal{M}^4_{(\alpha,\alpha)}$ является MDS-матрицей над P.

Доказательство. Справедливость теоремы показывается аналогично доказательству теоремы 1. ■

Вычисление образа $\mathbf{a} \cdot \mathcal{M}^4_{(\alpha,0)}$ вектора $\mathbf{a} = (a_0, a_1, a_2, a_3)$, как показано на рис. 4, осуществляется в результате четырёхкратной итерации обобщённой сети Фейстеля, которая может быть эффективно реализована, поскольку использует умножение на *фиксированный* элемент поля. Аналогичное представление имеет место для других двух матриц теоремы 7.

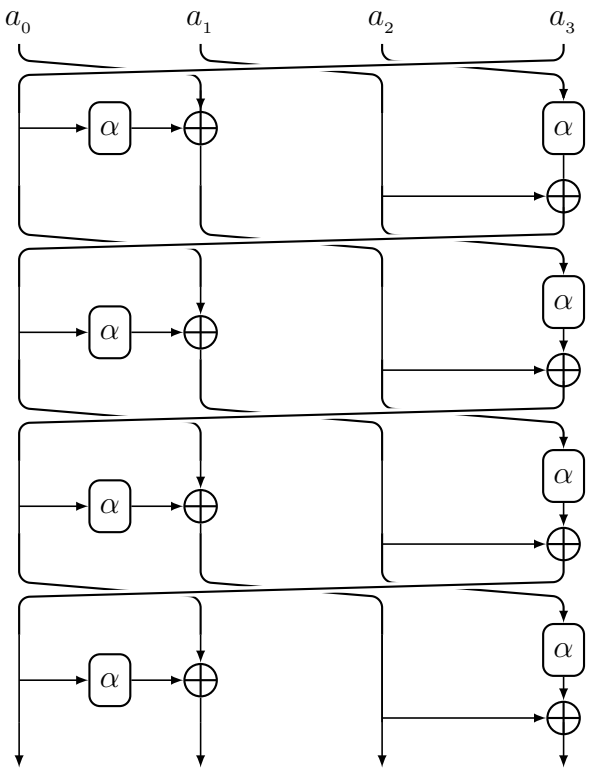


Рис. 4. Действие отображения $\mathbf{a} \cdot \mathcal{M}^4_{(\alpha,0)}$

5. XOR-сложность некоторых MDS-отображений

Пусть $P = \mathrm{GF}(2^8) = \mathrm{GF}(2)[x]/(x^8 + x^7 + x^6 + x + 1)$, α — примитивный элемент поля P и $f(x) = x^t + f_{t-1}x^{t-1} + \ldots + f_1 x + f_0 \in P[x]$, где для любого $i \in \{0, \ldots, t-1\}$ коэффициент $f_i \in \{0, 1, \alpha, \alpha + 1, \alpha^2\}$.

XOR-сложность элементов поля P представлена в табл. 3. Записи соответствуют XOR-сложности элемента, полученного в результате конкатенации соответствующего шестнадцатеричного значения строки и столбца. Например, $\mathsf{XOR}(\mathsf{0x27}) = 28$.

Таблица 3

XOR-сложность элементов поля P

XOR	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	0	0	3	9	5	11	10	**14**	7	11	12	18	14	20	13	21
1.	12	18	11	19	13	17	18	**24**	17	23	22	26	12	20	23	29
2.	**16**	**22**	**21**	**25**	**11**	**19**	**22**	28	**17**	**23**	**16**	**24**	**18**	**22**	**23**	**29**
3.	20	24	25	31	27	33	26	**34**	11	19	22	28	24	30	29	33
4.	20	24	23	29	25	31	26	**34**	11	19	20	26	22	28	29	33
5.	18	24	25	29	15	23	24	**30**	19	25	20	28	22	26	25	31
6.	24	30	25	33	27	31	30	**36**	29	35	36	40	26	34	35	41
7.	10	18	19	25	21	27	28	**32**	25	29	28	34	30	36	31	39
8.	25	21	26	20	24	22	31	**27**	30	26	33	31	27	21	36	32
9.	11	5	20	16	22	18	25	**23**	22	20	29	25	31	27	32	26
a.	19	17	26	22	28	24	29	**23**	14	8	23	19	25	21	28	26
b.	21	17	24	22	18	12	27	**23**	22	18	23	17	21	19	28	24
c.	27	23	32	30	26	20	33	**29**	28	24	31	25	29	27	34	30
d.	31	29	36	32	38	34	41	**35**	26	20	33	29	35	31	40	38
e.	9	3	16	12	18	14	23	**21**	20	18	25	21	27	23	30	24
f.	25	21	28	22	26	24	31	**27**	30	26	35	33	29	23	36	32

Используя результаты из табл. 3, приведём следующие рассуждения. Пусть $f(x) = x^4 + x + \beta$. Для сопровождающей матрицы данного многочлена справедливо равенство $\mathsf{XOR}(S_f) = (2-1) \cdot 8 + \mathsf{XOR}(\beta) = \mathsf{XOR}(\beta) + 8$. Найдём $\mathsf{XOR}(L_{f,\beta})$ отображения $L_{f,\beta}$, действие которого представлено на рис. 1. Пусть $\mathbf{z}_{(1)}, \mathbf{z}_{(2)}$ — выход последних двух регистров сдвига данной схемы, $\mathbf{w}_{(1)}, \mathbf{w}_{(2)}$ — выход отображений ψ_β и ψ'_β соответственно. Нетрудно заметить, что $\mathsf{XOR}(L_{f,\beta}) = 4XOR(S_f) + \mathsf{XOR}(\psi_\beta) + \mathsf{XOR}(\mathbf{z}_{(1)} \oplus \oplus \mathbf{w}_{(1)}) + \mathsf{XOR}(\psi'_\beta) + \mathsf{XOR}(\mathbf{z}_{(2)} \oplus \mathbf{w}_{(2)})$. Пусть $\alpha \in P$ такой, что $L_{f,\beta}$ — MDS-отображение и $\mathsf{XOR}(L_{f,\alpha})$ достигает минимального значения среди всех $\beta \in P$, таких, что $L_{f,\beta}$ — MDS-отображение. Тогда при $\beta = \alpha = \mathsf{0x02}$ получим $\mathsf{XOR}(L_{f,\beta}) = 64 + 4\mathsf{XOR}(S_f) + + 4(\mathsf{XOR}(\beta) + \mathsf{XOR}(\beta^4 + \beta)) = 64 + 4 \cdot 11 + 4(3 + 11) = 164$. По формуле (2) получим, что $\mathsf{XOR}(S^{22}) = 22 \cdot 11 = 242$. Из приведённого анализа можно заключить, что лучше использовать для программной и аппаратной реализации схему, представленную на рис. 1, поскольку она менее стоимостная, чем умножение вектора $\mathbf{a} \in P^4$ на матрицу S^{22} из теоремы 4.

Рассмотрим теперь отображения из теорем 5 и 6. По рис. 2 и 3 заметим, что для любого $i \in \{1, 2, 3\}$ при соответствующих значениях k верны равенства $\mathcal{L}^4_{\lambda_i,k}(\mathbf{a}) = = \mathbf{a}((S^{k_1}_{\lambda_i})^{\mathrm{T}} \oplus E)((S^{k_2}_{\lambda_i})^{\mathrm{T}} \oplus E)$ и $\mathcal{L}^6_{\phi_i,k}(\mathbf{a}) = \mathbf{a}((S^8_{\phi_i})^{\mathrm{T}} \oplus E)((S^{k'}_{\phi_i})^{\mathrm{T}} \oplus E)$ соответственно. Для вычисления $(\mathbf{a}(S^{k_1}_{\lambda_i})^{\mathrm{T}} \oplus \mathbf{a}) = \mathbf{z}$ и $(\mathbf{z}(S^{k_2}_{\lambda_i})^{\mathrm{T}} \oplus \mathbf{z}) = \mathbf{b}$ необходимо $2(4 \cdot 8) = = 64$ операции XOR. Аналогичным образом получим, что для реализации операций $(\mathbf{a}(S^8_{\phi_i})^{\mathrm{T}} \oplus \mathbf{a}) = \mathbf{z}$ и $(\mathbf{z}(S^{k'}_{\phi_i})^{\mathrm{T}} \oplus \mathbf{z}) = \mathbf{b}$ необходимо $2(6 \cdot 8) = 96$ операций XOR. Тогда из равенств (1) и (2) следует

$$\mathsf{XOR}(\mathcal{L}^4_{\lambda_i,k}) = k \cdot \mathsf{XOR}(S_{\lambda_i}) + 64, \ \mathsf{XOR}(\mathcal{L}^6_{\phi_i,k}) = k \cdot \mathsf{XOR}(S_{\phi_i}) + 96.$$

Проведём аналогичные рассуждения для матрицы из п. 4. Из табл. 2 можно определить, что количество побитовых операций XOR, необходимых для реализации матриц $\mathcal{M}^4_{(\alpha,0)}$, $\mathcal{M}^4_{(\alpha^2,\alpha)}$ и $\mathcal{M}^4_{(\alpha,\alpha)}$, равно $4 \cdot 22$, $4 \cdot 24$ и $4 \cdot 28$ соответственно.

Пусть $h_1(x) = x^4 + \beta^2 x^3 + x^2 + \beta x + 1$, $h_2(x) = x^4 + (\beta + 1)x^3 + x^2 + \beta x + 1$, $h_3(x) = x^4 + \beta^2 x^3 + x^2 + x + \beta \in \mathrm{GF}(2^n)[x]$. В [4] показано, что при некоторых $\beta \in \mathrm{GF}(2^n)$ матрица $S^4_{h_i}$ является MDS-матрицей для любого $i \in \{1, 2, 3\}$.

В [3] авторы используют многочлены $g_1(x) = x^6 + 2x^5 + 8x^4 + 5x^3 + 8x^2 + 2x + 1$ и $g_2(x) = x^6 + 4x^5 + x^4 + 2x^3 + x^2 + 3x + 2$ для построения MDS-матриц размера 6×6, которые реализованы в семействе хэш-функций PHOTON. Получено, что над полем $\mathrm{GF}(2^n) = \mathrm{GF}(2)[x]/(x^8 + x^4 + x^3 + x + 1)$ матрица $S^6_{g_i}$ является MDS-матрицей, где $i \in \{1, 2\}$. Заметим, что многочлены $g_1(x)$ и $g_2(x)$ имеют вид

$$g'_1(x) = x^6 + \beta x^5 + \beta^3 x^4 + (\beta^2 \oplus 1)x^3 + \beta^3 x^2 + \beta x + 1,$$
$$g'_2(x) = x^6 + \beta^2 x^5 + x^4 + \beta x^3 + x^2 + (\beta \oplus 1)x + \beta$$

при некотором $\beta \in \mathrm{GF}(2^n)$. Используя значения из табл. 3 и равенства (1) и (2), получим результаты, представленные в табл. 4 и 5.

Таблица 4

Сравнение параметра XOR-сложность для MDS-отображений множества P^4

MDS-отображения	$S^4_{h_1}$	$S^4_{h_2}$	$S^4_{h_3}$	$L_{f,\alpha}$	$\mathcal{L}^4_{\lambda_1,3}$	$\mathcal{L}^4_{\lambda_2,6}$	$\mathcal{L}^4_{\lambda_3,9}$	$\mathcal{M}^4_{(\alpha,0)}$	$\mathcal{M}^4_{(\alpha^2,\alpha)}$	$\mathcal{M}^4_{(\alpha,\alpha)}$
XOR-сложность	128	144	128	164	136	130	163	88	96	112

Таблица 5

Сравнение параметра XOR-сложность для MDS-отображений множества P^6

MDS-отображения	$S^6_{g'_1}$	$S^6_{g'_2}$	$\mathcal{L}^6_{\phi_1,9}$	$\mathcal{L}^6_{\phi_2,9}$	$\mathcal{L}^6_{\phi_3,10}$
XOR-сложность	366	342	312	312	336

Из табл. 4 и 5 можно сделать следующие выводы:

1) отображения $\mathcal{M}^4_{(\alpha,0)}$, $\mathcal{M}^4_{(\alpha^2,\alpha)}$ и $\mathcal{M}^4_{(\alpha,\alpha)}$ имеют наилучшие значения XOR-сложности среди всех изучаемых отображений, хотя при этом приходится отказаться от использования линейных регистров сдвига для их реализации;
2) аппаратная реализация отображений $\mathcal{L}^6_{\phi_1,9}$, $\mathcal{L}^6_{\phi_2,9}$ и $\mathcal{L}^6_{\phi_3,10}$ менее сложная, чем аппаратная реализация умножения векторов на матрицы ${S_{g'_1}}^6$ и ${S_{g'_2}}^6$.

Заключение

В работе предложены новые методы построения MDS-матриц с помощью рекурсивных процедур. Рассмотренные отображения реализуются с использованием линейных регистров сдвига и обобщённой сети Фейстеля. Они обладают хорошими эксплуатационными характеристиками с точки зрения реализации на вычислительных платформах с ограниченными ресурсами.

ЛИТЕРАТУРА

1. *Shannon C. E.* Communication theory of secrecy systems // Bell System Technical J. 1949. V. 28. No. 4. P. 656–715.

2. *Augot D. and Finiasz M.* Direct construction of recursive MDS diffusion layers using shortened BCH codes // Intern. Workshop on Fast Software Encryption. Springer, 2014. P. 3–17.
3. *Guo J., Peyrin T., and Poschmann A.* The PHOTON family of lightweight hash functions // Ann. Cryptology Conf. Springer, 2011. P. 222–239.
4. *Gupta K. C. and Ray I. G.* On constructions of MDS matrices from companion matrices for lightweight cryptography // Intern. Conf. Availability, Reliability, and Security. Springer, 2013. P. 29–43.
5. *Sarkar S. and Sim S. M.* A deeper understanding of the XOR count distribution in the context of lightweight cryptography // Intern. Conf. Cryptology in Africa. Springer, 2016. P. 167–182.
6. *Toh D., Teo J., Khoo K., and Sim S. M.* Lightweight MDS serial-type matrices with minimal fixed XOR count // Intern. Conf. Cryptology in Africa. Springer, 2018. P. 51–71.
7. *Burov D. A. and Pogorelov B. A.* The influence of linear mapping reducibility on the choice of round constants // Математические вопросы криптографии. 2017. Т. 8. № 2. С. 51–64.
8. *Глухов М. М., Елизаров В. П., Нечаев А. А.* Алгебра: 2-е изд., испр. и доп. Санкт-Петербург; Москва; Краснодар: Лань, 2015.

REFERENCES

1. *Shannon C. E.* Communication theory of secrecy systems. Bell System Technical J., 1949, vol. 28, no. 4, pp. 656–715.
2. *Augot D. and Finiasz M.* Direct construction of recursive MDS diffusion layers using shortened BCH codes. Intern. Workshop on Fast Software Encryption, Springer, 2014, pp. 3–17.
3. *Guo J., Peyrin T., and Poschmann A.* The PHOTON family of lightweight hash functions. Ann. Cryptology Conf., Springer, 2011, pp. 222–239.
4. *Gupta K. C. and Ray I. G.* On constructions of MDS matrices from companion matrices for lightweight cryptography. Intern. Conf. Availability, Reliability, and Security, Springer, 2013, pp. 29–43.
5. *Sarkar S. and Sim S. M.* A deeper understanding of the XOR count distribution in the context of lightweight cryptography. Intern. Conf. Cryptology in Africa, Springer, 2016, pp. 167–182.
6. *Toh D., Teo J., Khoo K., and Sim S. M.* Lightweight MDS serial-type matrices with minimal fixed XOR count. Intern. Conf. Cryptology in Africa, Springer, 2018, pp. 51–71.
7. *Burov D. A. and Pogorelov B. A.* The influence of linear mapping reducibility on the choice of round constants. Matematicheskie Voprosy Kriptografii, 2017, vol. 8, no. 2, pp. 51–64.
8. *Glukhov M. M., Elizarov V. P., and Nechaev A. A.* Algebra [Algebra]. St. Petersburg; Moscow; Krasnodar, Lan Publ., 2015. (in Russian)

УДК 621.391:519.7+621.391.1:004.7

ПОКАЗАТЕЛЬ 2-ТРАНЗИТИВНОСТИ ОДНОГО КЛАССА ПОДСТАНОВОК КОНЕЧНОГО ПОЛЯ

Д. У. Эрнандес Пилото

ООО «Центр сертификационных исследований», г. Москва, Россия

А. В. Аборневым предложен класс подстановок простого поля, построенный с помощью разрядных функций над кольцом вычетов по модулю p^2. В данной работе рассматривается более широкий класс подстановок произвольного конечного поля, полученный заменой разрядных функций на произвольные отображения. Приводится оценка снизу показателя 2-транзитивности множества Σh, где Σ — регулярная группа подстановок, а h — подстановка из нового класса. Получены достаточные условия достижимости указанной оценки.

Ключевые слова: *подстановки конечных полей, транзитивные группы подстановок, показатель 2-транзитивности.*

DOI 10.17223/20710410/46/2

2-TRANSITIVITY DEGREE FOR ONE CLASS OF SUBSTITUTIONS OVER FINITE FIELDS

D. H. Hernández Piloto

Certification Research Center, Moscow, Russia

E-mail: dhhernandez2410@gmail.com

The paper deals with the class of substitutions proposed by A. V. Abornev, constructed using digit functions γ_1 over the ring $\mathbb{Z}_{p^2}$ of the form $h(\mathbf{x}) = \mathbf{z}$, where $\mathbf{z} = \mathbf{z}_1 + p\mathbf{z}_2$, $(\mathbf{z}_1|\mathbf{z}_2) = \gamma_1(\mathbf{x}K)$ and K is a matrix of dimensions $m \times 2m$. We consider a generalization of this class of substitutions using arbitrary functions $F : P^m \to P^m$ over finite field P in the place of the digit functions γ_1. A set Σ is called 2-transitive if for any pairs $\alpha = (a_1, a_2)$, $\beta = (b_1, b_2)$ in Σ there exists a substitution g, such that $g(a_i) = b_i$, $i \in \{1, 2\}$. We are interested in the degree of 2-transitivity of a group Σ, denoted by $d_2(\Sigma)$, which is equal to the smallest natural value k, such that $(\Sigma)^k$ is a 2-transitive group. The main goal is to find groups of substitutions with the minimum of this parameter. Using our construction, it is demonstrated that the degree of 2-transitivity is lower bounded by 4. When $F(x + a) - F(x)$ is a substitution for any $a \in P^m \backslash \{\mathbf{0}\}$, the degree of 2-transitivity of the composition Σh is equal to 4. In other papers these functions were called planar. Notice that in a field with characteristic 2 planar functions do not exist. If the characteristic is not 2, then these functions exist. Indeed, if Q is an extension of degree m of P, $\hat{F}(x) = x^2$ for all $x \in Q$, and $\alpha_1, \ldots, \alpha_m$ is the base of the vector space Q_P, then the function $F(x_1, \ldots, x_m) = \hat{F}(\alpha_1 x_1 + \ldots + \alpha_m x_m)$, $x_1, \ldots, x_m \in P$, is planar.

Keywords: *transitivity, degree of 2-transitivity, digit function, regular group, substitution.*

Введение

Пусть p — произвольное простое число и $R = \mathbb{Z}_{p^2}$ — кольцо вычетов по модулю p^2.

Определение 1. Подмножество $\Gamma(R) = \{a \in R : a^p = a\}$ называют p-адическим разрядным множеством кольца R.

Каждый элемент $a \in R$ однозначно представляется в виде $a = a_0 + pa_1$, $a_i \in \Gamma(R)$, $i \in \{0, 1\}$, называемом p-адическим разложением элемента a.

Определение 2. Отображения $\gamma_i : R \to \Gamma(R)$, $\gamma_i(a) = a_i$, $i \in \{0, 1\}$, будем называть p-адическими разрядными функциями, а элементы $a_i = \gamma_i(a)$ — p-адическими разрядами элемента a.

Для каждого вектора $\mathbf{a} = (a_1, \ldots, a_t) \in R^t$ определим вектор $\gamma_1(\mathbf{a}) = (\gamma_1(a_1), \ldots, \gamma_1(a_t))$. Обозначим через $R^*_{m,m}$ множество всех обратимых матриц порядка m над кольцом R.

Определение 3. Назовём матрицу K размера $m \times n$ над R разрядно-инъективной, если любая ненулевая строка $\mathbf{a} \in R^m$ однозначно восстанавливается по строке $\gamma_1(\mathbf{a}K) \in R^n$.

Теорема 1 [1]. Пусть $G \in R^*_{m,m}, U \in R^*_{m,m}$. Тогда матрица

$$K = U(E|E + pG)_{m \times 2m}$$

является разрядно-инъективной и отображение $h : R^m \to R^m$, действующее на произвольной строке $\mathbf{x} \in R^m$ по правилу

$$h(\mathbf{x}) = \mathbf{z}, \text{ где } \mathbf{z} = \mathbf{z}_1 + p\mathbf{z}_2 \in R^m, (\mathbf{z}_1|\mathbf{z}_2) = \gamma_1(\mathbf{x}K) \in R^{2m},$$

является подстановкой.

Используя p-адическое разложение, запишем матрицу U в виде $U = U_0 + pU_1$. Тогда

$$\begin{gathered}\gamma_1(\mathbf{x}K) = (\gamma_1((\mathbf{x_0} + p\mathbf{x_1})(U_0 + pU_1)), \gamma_1((\mathbf{x_0} + p\mathbf{x_1})(U_0 + p(U_1 + U_0G)))) = \\ = (\gamma_1(\mathbf{x_0}U_0) + \mathbf{x_0}U_1 + \mathbf{x_1}U_0, \gamma_1(\mathbf{x_0}U_0) + \mathbf{x_0}U_1 + \mathbf{x_1}U_0 + \mathbf{x_0}U_0G),\end{gathered}$$

где в правой части равенства сложение векторов осуществляется покоординатно в поле $\mathbb{Z}_p$. Следовательно, отображение h действует по правилу

$$(\mathbf{x_0}, \mathbf{x_1}) \mapsto (\gamma_1(\mathbf{x_0}U_0) + \mathbf{x_0}U_1 + \mathbf{x_1}U_0, \gamma_1(\mathbf{x_0}U_0) + \mathbf{x_0}U_1 + \mathbf{x_1}U_0 + \mathbf{x_0}U_0G). \tag{1}$$

Схематично отображение h представлено на рис. 1.

Таким образом, имеем следующую систему:

$$\begin{cases}\gamma_1(\mathbf{x_0}U_0) + \mathbf{x_0}U_1 + \mathbf{x_1}U_0 = \mathbf{a_0}, \\ \mathbf{a_0} + \mathbf{x_0}U_0G = \mathbf{a_1}.\end{cases}$$

Ввиду указанного строения стало очевидным, что вычисление прообраза подстановки сводится к решению системы линейных уравнений и имеет сложность $\mathrm{O}(m^3)$ при $m \to \infty$.

В настоящей работе предлагается более общая конструкция, полученная заменой на рис. 1 блока, отмеченного пунктирными линиями, на произвольное отображение F. Построенные подстановки рассматриваются над произвольным конечным полем P.

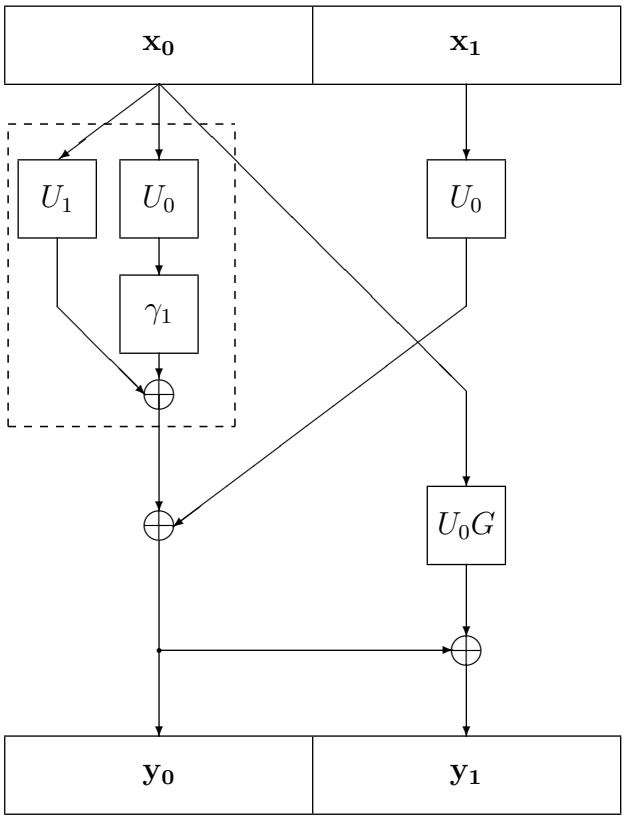


Рис. 1

1. Новая конструкция и ее свойства

Пусть P — произвольное конечное поле. Рассмотрим отображение $h : P^{2m} \to P^{2m}$, действующее по правилу

$$h(\mathbf{x_0}, \mathbf{x_1}) = (F(\mathbf{x_0}) + \mathbf{x_1}U, F(\mathbf{x_0}) + \mathbf{x_1}U + \mathbf{x_0}G), \tag{2}$$

где $\mathbf{x_0}, \mathbf{x_1} \in P^m$; $U, G \in P^*_{m,m}$; $F : P^m \to P^m$ — произвольное отображение. Схематично отображение h представлено на рис. 2.

Теорема 2. Отображение $h : P^{2m} \to P^{2m}$, определённое равенством (2), является подстановкой.

Доказательство. Достаточно показать, что отображение h инъективно. Пусть $\mathbf{a}, \mathbf{b} \in P^{2m}$ такие, что $h(\mathbf{a_0}, \mathbf{a_1}) = h(\mathbf{b_0}, \mathbf{b_1})$. Тогда $h(\mathbf{a_0}, \mathbf{a_1}) = (F(\mathbf{a_0}) + \mathbf{a_1}U, F(\mathbf{a_0}) + \mathbf{a_1}U + \mathbf{a_0}G)$, $h(\mathbf{b_0}, \mathbf{b_1}) = (F(\mathbf{b_0}) + \mathbf{b_1}U, F(\mathbf{b_0}) + \mathbf{b_1}U + \mathbf{b_0}G)$ и справедливы равенства

$$\begin{cases} F(\mathbf{a_0}) + \mathbf{a_1}U = \mathbf{c_0}, \\ \mathbf{c_0} + \mathbf{a_0}G = \mathbf{c_1}, \\ F(\mathbf{b_0}) + \mathbf{b_1}U = \mathbf{c_0}, \\ \mathbf{c_0} + \mathbf{b_0}G = \mathbf{c_1}. \end{cases}$$

Отсюда $\mathbf{c_0} + \mathbf{a_0}G = \mathbf{c_0} + \mathbf{b_0}G$, т. е. $\mathbf{a_0}G = \mathbf{b_0}G$, а значит, $\mathbf{a_0} = \mathbf{b_0}$. Теперь $F(\mathbf{a_0}) + \mathbf{a_1}U = F(\mathbf{b_0}) + \mathbf{b_1}U$, а так как $\mathbf{a_0} = \mathbf{b_0}$, то $\mathbf{a_1}U = \mathbf{b_1}U$ и, следовательно, $\mathbf{a_1} = \mathbf{b_1}$. ■

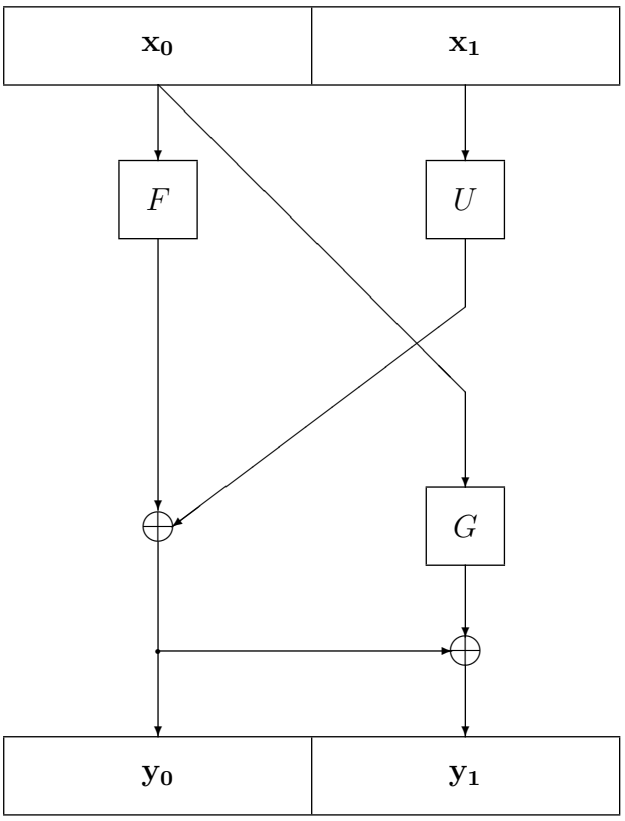


Рис. 2

Обозначим через $S(\Omega)$ симметрическую группу подстановок на множестве Ω.

Определение 4. Группа подстановок $\Sigma < S(\Omega)$ называется регулярной, если для любых $a, b \in \Omega$ в Σ существует единственная подстановка σ, удовлетворяющая условию $\sigma(a) = b$.

Рассмотрим произведение регулярной группы Σ на подстановку h.

Определение 5. Множество Σh называется 2-транзитивным, если для любых двух наборов $\alpha = (a_1, a_2), \beta = (b_1, b_2)$ из P^{2m} в Σh существует подстановка g, переводящая α в β, т. е. удовлетворяющая условию $g(a_i) = b_i$, $i \in \{1, 2\}$.

Определение 6. Показателем 2-транзитивности множества подстановок Σh называется число $d_2(\Sigma h)$, равное минимальному натуральному значению k, такому, что множество $(\Sigma h)^k$ является 2-транзитивным.

Интерес представляют подстановки с минимальным значением данного параметра. Согласно [2], достаточно рассмотреть матрицу Q_h переходов ненулевых разностей биграмм вида

$$(Q_h)_{(N-1)\times(N-1)} = \frac{1}{N}(\nu_{\mathbf{ab}})_{\mathbf{a},\mathbf{b}\in P^{2m}\setminus\theta},$$

где $\nu_{\mathbf{ab}} = |\{\mathbf{x} \in P^{2m} : h(\mathbf{x}+\mathbf{a}) - h(\mathbf{x}) = \mathbf{b}\}|$ и $N = |P^{2m}|$. Заметим, что для нахождения показателя 2-транзитивности достаточно описать степени матрицы Q_h, поскольку $d_2(\Sigma h) = k$ тогда и только тогда, когда $Q_h^{k-1} > 0$, а матрицы $Q_h^1, \ldots, Q_h^{k-2}$ содержат нулевые элементы.

Приведём некоторые факты из работы [1].

Утверждение 1. Элемент $\nu_{\mathbf{ab}}$ для матрицы Q_h равен числу решений системы уравнений

$$\begin{cases} \mathbf{b_0} = \gamma_1(\mathbf{a}U) + \gamma_1(\mathbf{x_0} + \gamma_0(\mathbf{a_0}U)), \\ \mathbf{b_1} = \mathbf{b_0} + \gamma_0(\mathbf{a}U)G - \gamma_1(\mathbf{x_1} + \mathbf{b_0}) \end{cases}$$

относительно $\mathbf{x_0}, \mathbf{x_1} \in \mathbb{Z}_p^m$.

Теорема 3. Пусть h — подстановка на $\mathbb{Z}_p^{2m}$, определённая равенством (1). Тогда $d_2(\Sigma h) \geqslant 4$.

Теорема 4. Пусть $p = 2$, $m \in \mathbb{N}$, подстановка h вида (1) выбрана так, что все миноры матрицы U_0 отличны от нуля. Тогда $d_2(\Sigma h) = 4$.

Получим аналоги этих утверждений применительно к новому классу подстановок. Пусть h — подстановка вида (2). Для оценки элементов $\nu_{\mathbf{ab}}$ матрицы Q_h используем

Утверждение 2. Элемент $\nu_{\mathbf{ab}}$ матрицы Q_h равен числу решений системы уравнений

$$\begin{cases} \mathbf{b_0} = F(\mathbf{x_0} + \mathbf{a_0}) - F(\mathbf{x_0}) + \mathbf{a_1}U, \\ \mathbf{b_1} = \mathbf{b_0} + \mathbf{a_0}G \end{cases}$$

относительно $\mathbf{x_0} \in P^m$.

Доказательство. Для произвольных обратимых матриц U, G, по определению, элемент $\nu_{\mathbf{ab}}$ есть число решений уравнения

$$\mathbf{b} = h(\mathbf{x} + \mathbf{a}) - h(\mathbf{x}) \tag{3}$$

относительно $\mathbf{x} \in P^{2m}$. Пользуясь (2), из (3) для строки $\mathbf{b} = (\mathbf{b_0}, \mathbf{b_1})$ получим равенство

$$\begin{gathered} h(\mathbf{x} + \mathbf{a}) - h(\mathbf{x}) = (F(\mathbf{x_0} + \mathbf{a_0}) + (\mathbf{x_1} + \mathbf{a_1})U, F(\mathbf{x_0} + \mathbf{a_0}) + (\mathbf{x_1} + \mathbf{a_1})U + (\mathbf{x_0} + \mathbf{a_0})G) - \\ -(F(\mathbf{x_0}) + \mathbf{x_1}U, F(\mathbf{x_0}) + \mathbf{x_1}U + \mathbf{x_0}G) = (F(\mathbf{x_0} + \mathbf{a_0}) + \mathbf{x_1}U + \mathbf{a_1}U, F(\mathbf{x_0} + \mathbf{a_0}) + \mathbf{x_1}U + \\ + \mathbf{a_1}U + \mathbf{x_0}G + \mathbf{a_0}G) - (F(\mathbf{x_0}) + \mathbf{x_1}U, F(\mathbf{x_0}) + \mathbf{x_1}U + \mathbf{x_0}G) = \\ = (F(\mathbf{x_0} + \mathbf{a_0}) - F(\mathbf{x_0}) + \mathbf{a_1}U, F(\mathbf{x_0} + \mathbf{a_0}) - F(\mathbf{x_0}) + \mathbf{a_1}U + \mathbf{a_0}G). \end{gathered}$$

Значит,

$$\begin{cases} \mathbf{b_0} = F(\mathbf{x_0} + \mathbf{a_0}) - F(\mathbf{x_0}) + \mathbf{a_1}U, \\ \mathbf{b_1} = \mathbf{b_0} + \mathbf{a_0}G. \end{cases}$$

Утверждение доказано. ■

Теорема 5. Пусть $h-$подстановка на P^{2m}, определённая равенством (2). Тогда $d_2(\Sigma h) \geqslant 4$.

Доказательство. Достаточно показать, что матрица Q_h^2 содержит нулевые элементы. Элементы $\nu_{\mathbf{ac}}^{(2)}$ матрицы Q_h^2 опишем, используя утверждение 2. Заметим, что

$$\nu_{\mathbf{ac}}^{(2)} = \frac{1}{N^2} \sum_{\mathbf{b} \in P^m} \nu_{\mathbf{ab}} \nu_{\mathbf{bc}}$$

и, согласно утверждению 2, произведение $\nu_{\mathbf{ab}}\nu_{\mathbf{bc}}$ равно числу решений $(\mathbf{x_0}, \mathbf{y_0}) \in (P^m)^2$ системы

$$\begin{cases} \mathbf{b_0} = F(\mathbf{x_0} + \mathbf{a_0}) - F(\mathbf{x_0}) + \mathbf{a_1}U, \\ \mathbf{b_1} = \mathbf{b_0} + \mathbf{a_0}G, \\ \mathbf{c_0} = F(\mathbf{y_0} + \mathbf{b_0}) - F(\mathbf{y_0}) + \mathbf{b_1}U, \\ \mathbf{c_1} = \mathbf{c_0} + \mathbf{b_0}G. \end{cases} \tag{4}$$

Элемент $\nu^{(2)}_{\mathbf{ac}}$ пропорционален числу решений системы (4) относительно системы независимых переменных $(\mathbf{x_0}, \mathbf{y_0}, \mathbf{b_0}, \mathbf{b_1}) \in (P^m)^4$. При этом условие $\nu^{(2)}_{\mathbf{ac}} > 0$ равносильно совместности системы (4) относительно последнего набора неизвестных. Покажем, что в матрице Q^2_h всегда есть нулевые элементы. Пусть $\mathbf{a_0} = \mathbf{c_0} = \mathbf{0}$. Первое и последнее уравнения системы (4) принимают следующий вид:

$$\begin{cases} \mathbf{b_0} = \mathbf{a_1}U, \\ \mathbf{c_1} = \mathbf{b_0}G. \end{cases}$$

Следствием данной системы является уравнение $\mathbf{c_1} = \mathbf{a_1}UG$. Пусть $\mathbf{a_1}, \mathbf{c_1} \in P^m$ выбраны так, что последнее равенство не выполняется. Тогда для $\mathbf{a_0} = \mathbf{c_0} = \mathbf{0}$ элемент $\nu^{(2)}_{\mathbf{ac}}$ равен нулю. ■

2. Условия для транзитивности и 2-транзитивности множества Σh

Пусть $\Sigma < S(\Omega)$ — регулярная группа подстановок и h — произвольная подстановка из $S(\Omega)$. Тогда несложно заметить, что произведение Σh является регулярным множеством подстановок, и, как следствие, оно транзитивно.

Теорема 6. Пусть $F(\mathbf{x}+\mathbf{f}) - F(\mathbf{x})$ для любого $\mathbf{f} \in P^m \backslash \{\mathbf{0}\}$ является подстановкой на P^m. Тогда для любой подстановки h вида (2) выполнено равенство $d_2(\Sigma h) = 4$.

Доказательство. Достаточно доказать, что $Q^3_h > 0$. Опишем элементы $\nu^{(3)}_{\mathbf{ad}}$ матрицы Q^3_h, используя утверждение 2. Заметим, что

$$\nu^{(3)}_{\mathbf{ad}} = \frac{1}{N^3} \sum_{\mathbf{b},\mathbf{c} \in P^m} \nu_{\mathbf{ab}} \nu_{\mathbf{bc}} \nu_{\mathbf{cd}}$$

и, согласно утверждению 2, произведение $\nu_{\mathbf{ab}}\nu_{\mathbf{bc}}\nu_{\mathbf{cd}}$ равно числу решений $(\mathbf{x_0}, \mathbf{y_0}, \mathbf{z_0}) \in \in (P^m)^3$ системы

$$\begin{cases} \mathbf{b_0} = F(\mathbf{x_0} + \mathbf{a_0}) - F(\mathbf{x_0}) + \mathbf{a_1}U, \\ \mathbf{b_1} = \mathbf{b_0} + \mathbf{a_0}G, \\ \mathbf{c_0} = F(\mathbf{y_0} + \mathbf{b_0}) - F(\mathbf{y_0}) + \mathbf{b_1}U, \\ \mathbf{c_1} = \mathbf{c_0} + \mathbf{b_0}G, \\ \mathbf{d_0} = F(\mathbf{z_0} + \mathbf{c_0}) - F(\mathbf{z_0}) + \mathbf{c_1}U, \\ \mathbf{d_1} = \mathbf{d_0} + \mathbf{c_0}G, \end{cases} \tag{5}$$

а $\nu^{(3)}_{\mathbf{ad}}$ пропорционально числу решений системы (5) относительно системы независимых переменных $(\mathbf{x_0}, \mathbf{y_0}, \mathbf{z_0}, \mathbf{b_0}, \mathbf{b_1}, \mathbf{c_0}, \mathbf{c_1}) \in (P^m)^7$. При этом условие $\nu^{(3)}_{\mathbf{ad}} > 0$ равносильно совместности системы (5) относительно последнего набора неизвестных. Покажем, что в условиях теоремы неравенство $\nu^{(3)}_{\mathbf{ad}} > 0$ выполняется при всех $\mathbf{a}, \mathbf{d} \in P^m \backslash \mathbf{0}$. Из пятого и шестого уравнений системы (5) можно выразить переменные $\mathbf{c_0}, \mathbf{c_1}$ соответственно через остальные переменные следующим образом:

$$\mathbf{c_0} = (\mathbf{d_1} - \mathbf{d_0})G^{-1},$$
$$\mathbf{c_1} = (\mathbf{d_0} - (F(\mathbf{z_0} + \mathbf{c_0}) - F(\mathbf{z_0})))U^{-1}.$$

Подставив в третье и пятое уравнения выражения для $\mathbf{b_1}, \mathbf{c_1}$ из второго и четвертого уравнений соответственно, получим

$$F(\mathbf{y_0} + \mathbf{b_0}) - F(\mathbf{y_0}) = \mathbf{c_0} - \mathbf{b_0}U - \mathbf{a_0}GU,$$
$$F(\mathbf{z_0} + \mathbf{c_0}) - F(\mathbf{z_0}) = \mathbf{d_0} - \mathbf{c_0}U - \mathbf{b_0}GU.$$

Из приведённых рассуждений следует, что система (5) равносильна системе уравнений

$$\begin{cases} F(\mathbf{y_0}+\mathbf{b_0})-F(\mathbf{y_0})=\mathbf{c_0}-\mathbf{b_0}U-\mathbf{a_0}GU, \\ F(\mathbf{z_0}+\mathbf{c_0})-F(\mathbf{z_0})=\mathbf{d_0}-\mathbf{c_0}U-\mathbf{b_0}GU, \\ \mathbf{b_0}=F(\mathbf{x_0}+\mathbf{a_0})-F(\mathbf{x_0})+\mathbf{a_1}U, \\ \mathbf{c_0}=(\mathbf{d_1}-\mathbf{d_0})G^{-1}, \\ \mathbf{b_1}=\mathbf{b_0}+\mathbf{a_0}G, \\ \mathbf{c_1}=(\mathbf{d_0}-(F(\mathbf{z_0}+\mathbf{c_0})-F(\mathbf{z_0})))U^{-1} \end{cases} \tag{6}$$

и совместна тогда и только тогда, когда совместна система из первых четырёх уравнений системы (6). Пусть $\mathbf{a}\in P^m\backslash\mathbf{0}$ — произвольный вектор. Покажем, что в уравнении

$$\mathbf{b_0}=F(\mathbf{x_0}+\mathbf{a_0})-F(\mathbf{x_0})+\mathbf{a_1}U$$

найдётся $\mathbf{x_0}\in P^m$, при котором $\mathbf{b_0}\neq\mathbf{0}$. Для доказательства рассмотрим два случая:

1) Пусть $\mathbf{a_0}=\mathbf{0}$, тогда $\mathbf{b_0}=\mathbf{a_1}U\neq\mathbf{0}$.

2) Если $\mathbf{a_0}\neq\mathbf{0}$, то существование искомого $\mathbf{x_0}$ следует из того, что $F(\mathbf{x_0}+\mathbf{a_0})-F(\mathbf{x_0})$ не является константой. Зафиксируем произвольные $\mathbf{a},\mathbf{d}\in P^m\backslash\mathbf{0}$. Элемент $\mathbf{x_0}\in P^m$ выберем так, чтобы выполнялось условие $\mathbf{b_0}\neq\mathbf{0}$. Если $\mathbf{c_0}\neq\mathbf{0}$, то получим систему

$$\begin{cases} F(\mathbf{y_0}+\mathbf{b_0})-F(\mathbf{y_0})=\mathbf{c_0}-\mathbf{b_0}U-\mathbf{a_0}GU, \\ F(\mathbf{z_0}+\mathbf{c_0})-F(\mathbf{z_0})=\mathbf{d_0}-\mathbf{c_0}U-\mathbf{b_0}GU, \end{cases} \tag{7}$$

совместную по выбору F. Если $\mathbf{c_0}=\mathbf{0}$, то

$$\begin{cases} F(\mathbf{y_0}+\mathbf{b_0})-F(\mathbf{y_0})=-\mathbf{b_0}U-\mathbf{a_0}GU, \\ \mathbf{d_0}=\mathbf{b_0}GU. \end{cases} \tag{8}$$

Преобразовав второе уравнение с использованием уравнений из системы (5)

$$\mathbf{d_0}=(\mathbf{b_1}-\mathbf{a_0}G)GU=\mathbf{b_1}GU-\mathbf{a_0}G^2U=(-(F(\mathbf{y_0}+\mathbf{b_0})-F(\mathbf{y_0}))U^{-1})GU-\mathbf{a_0}G^2U,$$

получим

$$\mathbf{d_0}=-(F(\mathbf{y_0}+\mathbf{b_0})-F(\mathbf{y_0}))U^{-1}GU-\mathbf{a_0}G^2U.$$

Тогда

$$F(\mathbf{y_0}+\mathbf{b_0})-F(\mathbf{y_0})=-d_0U^{-1}G^{-1}U-\mathbf{a_0}GU=-(\mathbf{b_0}GU)U^{-1}G^{-1}U-\mathbf{a_0}GU=-\mathbf{b_0}U-\mathbf{a_0}GU.$$

Получили первое уравнение из (8), и, аналогично (7), система совместна. Таким образом, доказано, что все элементы матрицы Q_h^3 положительны. ■

Отметим, что в случае поля характеристики 2 не существует отображений F, удовлетворяющих условию теоремы 6. Если характеристика поля отлична от 2, то такие отображения существуют. Действительно, пусть Q — расширение поля P степени m, $\hat{F}(x)=x^2$ для всех $x\in Q$. Тогда если $\alpha_1,\ldots,\alpha_m$ — базис векторного пространства Q_P, то условиям теоремы 6 удовлетворяет отображение $F(x_1,\ldots,x_m)=$ $=\hat{F}(\alpha_1x_1+\ldots+\alpha_mx_m)$, $x_1,\ldots,x_m\in P$. Функции, удовлетворяющие условию теоремы 6, названы в работе [3] планарными. В этой работе приводится обзор известных результатов о планарных функциях и указаны широкие классы рассматриваемых функций.

Заключение

Построен новый класс подстановок, обобщающий подстановки, предложенные в [1]. Удалось оценить снизу показатель 2-транзитивности для произведения регулярной группы подстановок на произвольную подстановку нового класса. Приведены достаточные условия обращения доказанной оценки в равенство.

Автор выражает благодарность И. В. Чередникy за постановку задачи и внимание к работе.

ЛИТЕРАТУРА

1. *Аборнев А. В.* Подстановки, индуцированные разрядно-инъективными преобразованиями модуля над кольцом Галуа // Прикладная дискретная математика. 2013. № 4. С. 5–15.
2. *Глухов М. М.* О 2-транзитивности произведения регулярных групп подстановок // Труды по дискретной математике. М.: Физматлит, 2000. С. 37–52.
3. *Глухов М. М.* О приближении дискретных функций линейными функциями // Математические вопросы криптографии. 2016. Т. 7. № 4. С. 29–50.

REFERENCES

1. *Abornev A. V.* Podstanovki, indutsirovannye razryadno-inektivnymi preobrazovaniyami modulya nad koltsom Galua [Substitutions induced by digit-injective transformations of a module over a Galois ring]. Prikladnaya Diskretnaya Matematika, 2013, no. 4, pp. 5–15. (in Russian)
2. *Glukhov M. M.* O 2-tranzitivnosti proizvedeniya regulyarnykh grupp podstanovok [On 2-transitivity of the composition of regular substitution groups]. Trudy po Diskretnoy Matematike. Moscow, Fizmatlit Publ., 2000, pp. 37–52. (in Russian)
3. *Glukhov M. M.* O priblizhenii diskretnykh funktsiy lineynymi funktsiyami [On the approximation of discrete functions by linear functions]. Matematicheskie Voprosy Kriptografii, 2016, vol. 7, no. 4, pp. 29–50. (in Russian)

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

UDC 519.7 DOI 10.17223/20710410/46/3

CRYPTANALYTICAL FINITE AUTOMATON INVERTIBILITY WITH FINITE DELAY[1]

G. P. Agibalov

National Research Tomsk State University, Tomsk, Russia

E-mail: agibalov@mail.tsu.ru

The paper continues an investigation of the cryptanalytical invertibility concept with a finite delay introduced by the author for finite automata. Here, we expound an algorithmic test for an automaton A to be cryptanalytically invertible with a finite delay, that is, to have a recovering function f which allows to calculate a prefix of a length m in an input sequence of the automaton A by using its output sequence of a length $m + \tau$ and some additional information about A defining a type of its invertibility and known to cryptanalysts. The test finds out whether the automaton A has a recovering function f or not and if it has, determines some or, may be, all of such functions. The test algorithm simulates a backtracking method for searching a possibility to transform a binary relation to a function by shortening its domain to a set corresponding to the invertibility type under consideration.

Keywords: *finite automata, information-lossless automata, automata invertibility, cryptanalytical invertibility, cryptanalytical invertibility test.*

Introduction

To continue the research we have begun in [1], we first present the problem under consideration, namely the automaton cryptanalytical invertibility, and connected with it basic concepts and terms.

An arbitrary finite automaton is represented by a 5-tuple $A = (X, Q, Y, \psi, \varphi)$, where X, Q, and Y are the input alphabet, the set of states and the output alphabet respectively, $\psi : X \times Q \to Q$ and $\varphi : X \times Q \to Y$. The last functions, being defined for pairs $xq \in \in X \times Q$, are expanded on pairs $\alpha q \in X^* \times Q$ by induction on the length $|\alpha|$ of a word $\alpha \in X^*$, namely the functions $\psi : X^* \times Q \to Q$ and $\bar{\varphi} : X^* \times Q \to Y^*$ are defined as $\psi(\Lambda, q) = q$, $\psi(\alpha\beta, q) = \psi(\beta, \psi(\alpha, q))$, $\bar{\varphi}(\Lambda, q) = \Lambda$, $\bar{\varphi}(x, q) = \varphi(x, q)$ and $\bar{\varphi}(\alpha\beta, q) = = \bar{\varphi}(\alpha, q)\bar{\varphi}(\beta, \psi(\alpha, q))$. The symbol Λ here denotes the empty word in any alphabet. Thus, $\psi(\alpha, q)$ is a state to which the automaton A goes from the state q under the action of the input word α, and $\bar{\varphi}(\alpha, q)$ is a word which it outputs under this action.

Everywhere further, τ means a natural number and is called a finite delay, and without another note, it is supposed that $\alpha \in X^m$ for $m = |\alpha\delta| - \tau$, $\delta \in X^\tau$, $q \in Q$. In dependence on context, the last symbols are considered as elements of the pointed sets respectively or as variables with these sets as their ranges.

In connection with the automaton A, we believe that q, α, and δ are the variables with values from Q, X^m, and X^τ denoting, respectively, an initial state, an information word, and

[1]The author is supported by the RFBR-grant no. 17-01-00354.

a delay word in an input sequence $\alpha\delta$ of the automaton A, and $K = \{\forall q, \forall\alpha, \forall\delta, \exists q, \exists\delta\}$ is a set of universal and existential quantifiers that binds these variables. Note that in K there is no the quantifier $\exists\alpha$. This is explained with the following argument: for a cryptanalyst, an information word α in an input word of the automaton A is supposed to be unknown and not some one, but any one. As for the length $m = |\alpha|$ of the word α, it is proposed to be known since it can be calculated as it is shown above where $|\alpha\delta| = |\bar{\varphi}(\alpha\delta, q)|$ and $\bar{\varphi}(\alpha\delta, q)$ is a sequence supervised on the output of A by a cryptanalyst. Also, let $V_0 = \{q, \delta, \psi(\alpha, q), \psi(\alpha\delta, q)\}$ where q, $\psi(\alpha, q)$, and $\psi(\alpha\delta, q)$ are, respectively, the initial, intermediate and final states of the automaton A and δ is a delay word. For any subset $\upsilon \subseteq V_0$, let $\upsilon(q, \alpha, \delta)$ be the system of functions (or vector function) represented by the formulas in υ and depending on variables q, α, δ denoting respectively an initial state, an input word, and a delay word in the automaton A. Denote D_υ the range of the function $\upsilon(q, \alpha, \delta)$, that is, the set of its possible values.

1. Automata cryptanalitical invertibility problem

The automaton A is called (cryptanalitically) invertible with a delay τ if there exist quantifiers K_1, K_2, K_3 in K with the different variables from $\{q, \alpha, \delta\}$, a subset $\upsilon \subseteq V_0$ and a function $f : Y^{m+\tau} \times D_\upsilon \to X^m$ such that

$$K_1K_2K_3(f(\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) = \alpha); \tag{1}$$

in this case f is called a recovering function (it recovers α using $\bar{\varphi}(\alpha\delta, q)$ and $\upsilon(q, \alpha, \delta)$), the 4-tuple $IT = (K_1K_2K_3, \upsilon)$, the triple $ID = (K_1K_2K_3)$, and $IO = \upsilon$ are respectively called a type, a degree, and an order of (cryptanalytical) invertibility of the automaton A.

In this definition, $K_i = Q_ix_i$ for each $i = 1, 2, 3$, a quantifier symbol $Q_i \in \{\forall, \exists\}$, and a variable $x_i \in \{q, \alpha, \delta\}$. Therefore, at the same time in the future, we equally use (1) and the expression

$$Q_1x_1Q_2x_2Q_3x_3(f(\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) = \alpha), \tag{2}$$

where $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$ and $Q_1x_1Q_2x_2Q_3x_3 = ID$.

The main problem that we consider in the paper, the problem of automata cryptanalytical invertibility — ACI, is the following decision one: given a finite automaton $A = (X, Q, Y, \psi, \varphi)$, an invertibility type $IT = (K_1K_2K_3, \upsilon) = (Q_1x_1Q_2x_2Q_3x_3, \upsilon)$, and a natural number τ, find out whether the automaton A is invertible of type IT with the delay τ and if so, construct a proper recovering function f satisfying the any of conditions (1) or (2).

2. Function cryptanalytical invertibility problem

To decide the problem ACI, we first try to decide the following auxiliary abstract mathematical problem of function invertibility — FI: given a function $g(x_1, \ldots, x_n)$, a quantifier word $Q_1x_1 \ldots Q_nx_n$, and a number $k_0 \in \{1, \ldots, n\}$ where $Q_{k_0} = \forall$, find out if there exist functions f such that the formula

$$Q_1x_1Q_2x_2 \ldots Q_nx_n(f(g(x_1, x_2, \ldots, x_n)) = x_{k_0}) \tag{3}$$

is true, and if exist, construct some of them.

Using the terms related to the cryptanalytical invertibility of an automaton, we can say that in this problem the question is about the invertibility of type $(Q_1x_1 \ldots Q_nx_n)$ for the function $g(x_1, \ldots, x_n)$ with respect to a variable x_{k_0} and with a recovering function $f : D_g \to D_{k_0}$ where D_g and D_{k_0} are the ranges of the function g and of the variable x_{k_0} respectively.

Clearly, the main problem (ACI) is obtained from the auxiliary one (FI) as the following particular case: $n = 3$, $k_0 \in \{1, 2, 3\}$, $\{x_1, x_2, x_3\} = \{q, \alpha, \delta\}$, $x_{k_0} = \alpha$, $g(x_1, x_2, x_3) = (\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta))$. Thus, any method deciding the auxiliary problem also decides the main one, and, so, our problem ACI reduces to our problem FI.

Every of the predicate logic formulas under consideration in the paper including (1)–(3) is written in a normal form $Q_1x_1 \ldots Q_nx_nP(x_1, \ldots, x_n)$, that is, with a quantifier prefix $Q_1x_1 \ldots Q_nx_n$ and its scope being an underlying predicate expression $P(x_1, \ldots, x_n)$ without quantifiers and, moreover, of the special kind $(f(g(x_1, \ldots, x_n)) = x_{k_0})$. We consider the quantifier prefix $Q_1x_1 \ldots Q_nx_n$ in it as a way to define a domain of values of subject variables $x_1, \ldots, x_n$ that the underlying function $P(x_1, \ldots, x_n)$ depends on. In fact, the quantifier prefix in it generates some n-tuples $a = a_1 \ldots a_n$ of values for the variable $x = x_1 \ldots x_n$, and the underlying expression calculates the values $g(a)$ and determines f by the equalities $f(g(a)) = a_{k_0}$. According to the quantifier logic [2], the quantifier $\forall x_k$ generates all the possible values a_k of the variable x_k from its range D_k and the quantifier $\exists x_k$ generates a one of the possible values a_k taken from D_k in dependence on the values $a_1, \ldots, a_{k-1}$ of the previous variables $x_1, \ldots, x_{k-1}$ respectively. From the cryptanalytical point of view, we suppose that the value a_k provided by the quantifier $\exists x_k$ as well as the rule of its generating, the function $h(x_1, \ldots, x_n) = f(g(x_1, \ldots, x_n))$ and, in general, the function of $P(x_1, \ldots, x_n)$ are a priori unknown to a cryptanalyst.

Note that under suppositions named above, we are forced to decide the FI problem by trying different values allegedly generated by an existential quantifier what many times complicates the deciding algorithm. The same effect results from determining a function f by the equations $f(g(a)) = a_{k_0}$, because the last very often (for example, when $g(a) = g(b)$ and $a_{k_0} \neq b_{k_0}$) determines not function f but a binary relation f which is not a function.

Consider (3) taking into account that has been just said in relation to the FI problem. Let $n = r + s$, $r \geqslant 1$, $s \geqslant 0$, $i_1 < \ldots < i_r$, $j_1 < \ldots < j_s$, $\{i_1 \ldots, i_r, j_1, \ldots, j_s\} = \{1, \ldots, n\}$, $Q_{i_1} = \ldots = Q_{i_r} = \forall$, $Q_{j_1} = \ldots = Q_{j_s} = \exists$, $D_1, \ldots, D_n$ and D_g are the ranges of variables x_1, $\ldots$, x_n and the function g respectively. So $k_0 \in \{i_1, \ldots, i_r\}$, $Q_{k_0} = \forall$, $g : D_1 \times \ldots \times D_n \to D_g$, $f : D_g \to D_{k_0}$. In the case $s = 0$ it is supposed that $\{j_1, \ldots, j_s\} = \varnothing$. Also, let for $k \in \{j_1, \ldots, j_s\}$, $\varepsilon_k : D_1 \times \ldots \times D_{k-1} \to D_k$ and $\varepsilon_k(a_1, \ldots, a_{k-1})$ denotes a value of the variable x_k, the existence of which is implied by a quantifier Q_kx_k with $Q_k = \exists$ in dependence on the values $a_1, \ldots, a_{k-1}$ chosen before by the quantifiers $Q_1x_1, \ldots, Q_{k-1}x_{k-1}$ for the variables $x_1, \ldots, x_{k-1}$ respectively. Further, in order to address or refer to functions $\varepsilon_k(a_1, \ldots, a_{k-1})$, we call them existential ones for their relation to quantifiers of the similar name. A function $\varepsilon_k(a_1, \ldots, a_{k-1})$ isn't obliged to essentially depend on each of its arguments. In this case we exclude inessential arguments from the list under the sign of the function. At last, if $s = 0$, that is, in the quantifier prefix under consideration there are no existential quantifiers and hence $D_{j_1} \times \ldots \times D_{j_s} = \varnothing$, then we have $\varepsilon_1 \ldots \varepsilon_s = \Lambda$.

Believing the value $\varepsilon_k(a_1, \ldots, a_{k-1})$ be unknown, to find out it we can try different elements a_k in D_k as the real value for ε_k and to pick out that of them, for which the equations $f(g(a_1, \ldots, a_n)) = a_{k_0}$ determine f as a function. In the case when no element in D_k satisfies this condition, we can change the value a_{k-1} of the previous variable x_{k-1} like in the method of backtracking search tree traversal [3–5].

For the quantifier prefix $Q_1x_1 \ldots Q_nx_n$ in (3), define a subset $M_n \subseteq D_1 \times \ldots \times D_n$ by induction on $k = 1, 2, \ldots, n$, namely let $M_0 = \{\Lambda\}$ and for each $k \in \{1, \ldots, n\}$, if $k \in \{i_1, \ldots, i_r\}$, then $M_k = \{a_1 \ldots a_{k-1}a_k : a_1 \ldots a_{k-1} \in M_{k-1}, a_k \in D_k\} = M_{k-1} \times D_k$, otherwise if $k \in \{j_1, \ldots, j_s\}$, then $M_k = \{a_1 \ldots a_{k-1}a_k : a_1 \ldots a_{k-1} \in M_{k-1}, a_k = \varepsilon_k(a_1, \ldots, a_{k-1})\} = M_{k-1} \times \{\varepsilon_k(a_1, \ldots, a_{k-1})\}$. By this definition, M_n is uniquelly

defined by the existential functions $\varepsilon_k(a_1, \ldots, a_{k-1})$, $k \in \{j_1, \ldots, j_s\}$. Therefore, we denote it M_ε where $\varepsilon = \varepsilon_{j_1} \ldots \varepsilon_{j_s}$ is the vector existential function of $Q_1x_1 \ldots Q_nx_n$, say that M_ε corresponds to these functions or shorter to ε and call M_ε the existential domain of the predicate word $Q_1x_1 \ldots Q_nx_n$ corresponding to existential functions in ε.

Notice that by the definition,

$$a_1 \ldots a_n \in M_\varepsilon \Leftrightarrow$$
$$\Leftrightarrow (a_1 \ldots a_n \in D_1 \times \ldots \times D_n) \;\&\; (a_{j_1} \ldots a_{j_s} = \varepsilon_{j_1}(a_1, \ldots, a_{j_1-1}) \ldots \varepsilon_{j_s}(a_1, \ldots, a_{j_s-1})),$$

that is, M_ε consists of those vectors $a_1 \ldots a_n$ in $D_1 \times \ldots \times D_n$ which are generated by the quantifier prefix by means of existential functions $\varepsilon_{j_1}, \ldots, \varepsilon_{j_s}$ (independently of the underlying expression) in such a way that a_k is any element in D_k if $Q_k = \forall$ or it is $\varepsilon_k(a_1, \ldots, a_{k-1})$ otherwise, $k \in \{1, \ldots, n\}$.

Also, please pay attention to the following property of the set M_ε, resulting from the functionality of mappings ε_k in its definition: for all $a_1a_2 \ldots a_n$ and $b_1b_2 \ldots b_n$ in M_ε and for any $k \in \{j_1, \ldots, j_s\}$ if $a_1 \ldots a_{k-1} = b_1 \ldots b_{k-1}$, then $a_k = \varepsilon_k(a_1, \ldots, a_{k-1}) = \varepsilon_k(b_1, \ldots, b_{k-1}) = b_k$.

Further, in dependence on context, we use the terms of existential function $\varepsilon_k(a_1, \ldots, a_{k-1})$ and of existential domain M_ε in connection not only with a quantifier prefix $Q_1x_1 \ldots Q_nx_n$ but with an automaton invertibility degree being denoted in the same way.

Now, we give some examples demonstrating what we have just discussed.

3. Examples of existential functions and domains of a predicate prefix

Example 1. Let $n = 3$, $x = x_1x_2x_3$, $g(x) = g(x_1, x_2, x_3) = (x_1x_2 + x_3) \bmod 3$, $D_1 = D_2 = D_3 = D_g = \{0, 1, 2\}$, $k_0 = 1$ and $x_{k_0} = x_1$, $f : D_g \to D_1$, $Q_1x_1Q_2x_2Q_3x_3(f(g(x_1, x_2, x_3)) = x_{k_0}) = \forall x_1 \forall x_2 \exists x_3 (f((x_1x_2 + x_3) \bmod 3) = x_1)$, the function $\varepsilon_3 : D_1 \times D_2 \to D_3$ is given in the Table 1.

Table 1

x_1x_2	00	01	02	10	11	12	20	21	22
$\varepsilon_3(x_1, x_2)$	2	2	2	1	0	2	0	1	2

Then $M_\varepsilon = M_{\varepsilon_3} = \{002, 012, 022, 101, 110, 122, 200, 211, 222\}$, the values $g(x)$ and $f(g(x))$ for $x \in M_\varepsilon$ are presented in the Table 2.

Table 2

$x \in M_\varepsilon$	002	012	022	101	110	122	200	211	222
$g(x)$	2	2	2	1	1	1	0	0	0
$f(g(x))$	0	0	0	1	1	1	2	2	2

It is immediately seen that M_ε is really generated by the quantifier prefix $\forall x_1 \forall x_2 \exists x_3$ by means of the existential function ε_3, and f is a function on D_g with the values in D_1, satisfying the underlying predicate equation for vectors in M_ε and hence proving the invertibility of type $(\forall x_1, \forall x_2, \exists x_3)$ of the function g with respect to the variable x_1 and with the recovering function f. We can add that in fact there are yet at least five other existential functions and five other recovering functions f, with which the function g in the example is invertible of the type $\forall x_1 \forall x_2 \exists x_3$ with respect to the variable x_1.

Example 2. This example only differs from the first one in the range D_3 which now is $D_3 = \{0, 1\}$ and in the existential function $\varepsilon_3 : D_1 \times D_2 \to D_3$ (Table 3).

Table 3

x_1x_2	00	01	02	10	11	12	20	21	22
$\varepsilon_3(x_1, x_2)$	0	0	0	1	0	1	1	0	1

In this case we obtain the following set

$$M_\varepsilon = M_{\varepsilon_3} = \{000, 010, 020, 101, 110, 121, 201, 210, 221\},$$

and the following functions $g(x)$ and $f(g(x))$ defined on it (Table 4).

Table 4

$x \in M_\varepsilon$	000	010	020	101	110	121	201	210	221
$g(x)$	0	0	0	1	1	0	1	2	2
$f(g(x))$	0	0	0	1	1	0	1	2	2

From the Table 4, it is seen that f is a function on D_g but it doesn't satisfy the equation $f(g(x_1, x_2, x_3)) = x_1$ on M_ε and therefore g is not invertible of the type $\forall x_1 \forall x_2 \exists x_3$ with existential function ε and with respect to the variable x_1. There is a suspicion that it is not invertible of this type with any existential function ε for $\exists$ and with respect to the same variable.

4. Existential functions and domains of automaton invertibility degrees

In [1], all the possible automaton cryptanalytical invertibility types were defined. In the section 1 of this paper, we have repeated the definition. Each type IT is characterised by an invertibility degree ID and invertibility order IO. Here, for each of all thirteen possible IDs $Q_1x_1Q_2x_2Q_3x_3$ of an automaton $A = (X, Q, Y, \psi, \varphi)$, we give the general description of ranges and domains for arbitrary existential functions $\varepsilon_1, \varepsilon_2, \varepsilon_3$ in it and, for any $\varepsilon \subseteq \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$, the general description of existential domain M_ε in the form of algorithm for computing vectors from $D_1 \times D_2 \times D_3$ in it.

In order to make the text of this section to be nearer to the automata theory language which we keep to in our research, instead of typical symbols x_1, x_2, and x_3 of abstract mathematical variables, we use the symbols q, α, and δ usually denoting in automata theory an initial state of the automaton A, its input information and its input delay words respectively. Besides, m is the length of α and τ is the length of δ.

1) $ID = \forall q \forall \alpha \forall \delta$, $\varepsilon = \Lambda$, $M_\varepsilon = \{q\alpha\delta : q \in Q, \alpha \in X^m, \delta \in X^\tau\}$;
2) $ID = \forall q \forall \alpha \exists \delta$, $\varepsilon_3 : Q \times X^m \to X^\tau$, $M_\varepsilon = M_{\varepsilon_3} = \{q\alpha\varepsilon_3(q, \alpha) : q \in Q, \alpha \in X^m\}$;
3) $ID = \forall q \exists \delta \forall \alpha$, $\varepsilon_2 : Q \to X^\tau$, $M_\varepsilon = M_{\varepsilon_2} = \{q\varepsilon_2(q)\alpha : q \in Q, \alpha \in X^m\}$;
4) $ID = \exists q \forall \alpha \forall \delta$, $\varepsilon_1 \in Q$, $M_\varepsilon = M_{\varepsilon_1} = \{\varepsilon_1\alpha\delta : \alpha \in X^m, \delta \in X^\tau\}$;
5) $ID = \exists q \forall \alpha \exists \delta$, $\varepsilon_1 \in Q$, $\varepsilon_3 : Q \times X^m \to X^\tau$, $M_\varepsilon = M_{\varepsilon_1\varepsilon_3} = \{\varepsilon_1\alpha\varepsilon_3(\varepsilon_1, \alpha) : \alpha \in X^m\}$;
6) $ID = \exists q \exists \delta \forall \alpha$, $\varepsilon_1 \in Q$, $\varepsilon_2 : Q \to X^\tau$, $M_\varepsilon = M_{\varepsilon_1\varepsilon_2} = \{\varepsilon_1\varepsilon_2(\varepsilon_1)\alpha : \alpha \in X^m\}$;
7) $ID = \forall \alpha \exists q \forall \delta$, $\varepsilon_2 : X^m \to Q$, $M_\varepsilon = M_{\varepsilon_2} = \{\alpha\varepsilon_2(\alpha)\delta : \alpha \in X^m, \delta \in X^\tau\}$;
8) $ID = \forall \alpha \exists q \exists \delta$, $\varepsilon_2 : X^m \to Q$, $\varepsilon_3 : X^m \times Q \to X^\tau$, $M_\varepsilon = M_{\varepsilon_2\varepsilon_3} =$
$= \{\alpha\varepsilon_2(\alpha)\varepsilon_3(\alpha, \varepsilon_2(\alpha)) : \alpha \in X^m\}$;
9) $ID = \forall \alpha \forall \delta \exists q$, $\varepsilon_3 : X^m \times X^\tau \to Q$, $M_\varepsilon = M_{\varepsilon_3} = \{\alpha\delta\varepsilon_3(\alpha, \delta) : \alpha \in X^m, \delta \in X^\tau\}$;
10) $ID = \forall \alpha \exists \delta \forall q$, $\varepsilon_2 : X^m \to X^\tau$, $M_\varepsilon = M_{\varepsilon_2} = \{\alpha\varepsilon_2(\alpha)q : \alpha \in X^m, q \in Q\}$;

11) $ID = \forall\delta\exists q\forall\alpha$, $\varepsilon_2 : X^\tau \to Q$, $M_\varepsilon = M_{\varepsilon_2} = \{\delta\varepsilon_2(\delta)\alpha : \alpha \in X^m, \delta \in X^\tau\}$;
12) $ID = \exists\delta\forall q\forall\alpha$, $\varepsilon_1 \in X^\tau$, $M_\varepsilon = M_{\varepsilon_1} = \{\varepsilon_1 q\alpha : q \in Q, \alpha \in X^m\}$;
13) $ID = \exists\delta\forall\alpha\exists q$, $\varepsilon_1 \in X^\tau$, $\varepsilon_3 : X^\tau \times X^m \to Q$, $M_\varepsilon = M_{\varepsilon_1\varepsilon_3} = \{\varepsilon_1\alpha\varepsilon_3(\varepsilon_1, \alpha) : \alpha \in X^m\}$.

From the given expressions for the sets M_ε, we can see the expressions for the size $|M_\varepsilon|$ of these sets. The Table 5 contains them for all numbers of ID. In it $k = |X|$, $h = |Q|$.

Table 5

$N.ID$	1	2	3	4	5	6	7	8	9	10	11	12	13
$\lvert M_\varepsilon\rvert$	$hk^{m+\tau}$	hk^m	hk^m	$k^{m+\tau}$	k^m	k^m	$k^{m+\tau}$	k^m	$k^{m+\tau}$	hk^m	$k^{m+\tau}$	hk^m	k^m

5. Test for function cryptanalytical invertibility

Lemma 1. For a function $g(x_1, \ldots, x_n)$, there exists a function $f : D_g \to D_{k_0}$ with the true formula (3), if and only if for each $k \in \{j_1, \ldots, j_s\}$ there exists an existential function $\varepsilon_k : D_1 \times \ldots \times D_{k-1} \to D_k$ such that the set M_ε corresponding to $\varepsilon = \varepsilon_{j_1} \ldots \varepsilon_{j_s}$ satisfies the following condition:

$$\forall a = a_1 \ldots a_n \in M_\varepsilon \, \forall b = b_1 \ldots b_n \in M_\varepsilon \, (a_{k_0} \neq b_{k_0} \Rightarrow g(a) \neq g(b)). \tag{4}$$

Proof. Necessity: given $\exists f((3))$, prove $\exists\varepsilon((4))$. We have:

$$\begin{gathered}
\exists f((3)) \Rightarrow \exists f(Q_1 x_1 \ldots Q_n x_n (f(g(x_1, \ldots, x_n)) = x_{k_0})) \Rightarrow \\
\Rightarrow \exists f \exists\varepsilon(\forall a = a_1 \ldots a_n \in M_\varepsilon (f(g(a)) = a_{k_0})) \Rightarrow \\
\Rightarrow \exists f \exists\varepsilon(\forall a \in M_\varepsilon (f(g(a)) = a_{k_0}) \;\&\; \forall b \in M_\varepsilon (f(g(b)) = b_{k_0})) \Rightarrow \\
\Rightarrow \exists f \exists\varepsilon(\forall a \in M_\varepsilon \, \forall b \in M_\varepsilon (f(g(a)) = a_{k_0}) \;\&\; (f(g(b)) = b_{k_0})) \Rightarrow \\
\Rightarrow \exists f \exists\varepsilon(\forall a \in M_\varepsilon \, \forall b \in M_\varepsilon (a_{k_0} \neq b_{k_0} \Rightarrow f(g(a)) \neq f(g(b)))) \Rightarrow \\
\Rightarrow \exists\varepsilon(\forall a \in M_\varepsilon \, \forall b \in M_\varepsilon (a_{k_0} \neq b_{k_0} \Rightarrow g(a) \neq g(b))) = \exists\varepsilon((4)).
\end{gathered}$$

Sufficiency: given $\exists\varepsilon((4))$, prove $\exists f((3))$. Define $f : D_g \to D_{k_0}$ as $f(g(a)) = a_{k_0}$, $a \in M_\varepsilon$. We have:

$$\exists\varepsilon((4)) = \exists\varepsilon(\forall a \in M_\varepsilon \, \forall b \in M_\varepsilon (a_{k_0} \neq b_{k_0} \Rightarrow g(a) \neq g(b)).$$

Therefore, $g(a) = g(b) \Rightarrow a_{k_0} = b_{k_0} \Rightarrow f(g(a)) = f(g(b))$ for any a and b from M_ε what means that f is a function on $\{g(a) : a \in M_\varepsilon\}$. So, $\forall a \in M_\varepsilon (f(g(a)) = a_{k_0})$ that is equivalent to $((3))$. ■

So, by trying the different existential functions ε on satisfying the existential domains M_ε to the condition (4), we can find out whether there exists a function f recovering a certain variable of a given function g or not.

Corollary 1. A function $g(x_1, \ldots, x_n)$ is invertible of a type $Q_1 x_1 \ldots Q_n x_n$ with respect to a variable x_{k_0}, $k_0 \in \{i_1, \ldots, i_r\}$, if and only if there exist existential functions $\varepsilon_k : D_1 \times \ldots \times D_{k-1} \to D_k$, $k = j_1, \ldots, j_s$, the corresponding to which set M_ε satisfies the condition (4).

So, by trying the different existential functions ε on satisfying the existential domains M_ε to the condition (4), we can find out whether a function g is invertible of a certain type with respect to some its variable or not.

Example 3. This is the end of Example 1. We see here that $\forall x_1 \forall x_2 \exists x_3 (f(g(x)) = x_1)$, that is, the state (3) is true as well as $\forall a \in M_\varepsilon \, \forall b \in M_\varepsilon (a_1 \neq b_1 \Rightarrow g(a) \neq g(b))$, that is,

the state (4) is true too. For instance, if $(x_1, x_2) = (1, 2)$, then $x_3 = 2$, $g(x_1, x_2, x_3) = (x_1x_2 + x_3) \bmod 3 = 1$ and $f(g(x_1, x_2, x_3)) = f(1) = 1 = x_1$, and also if $a = 020$ and $b = 101$, then $a_1 = 0 \neq 1 = b_1$ and $g(a) = g(020) = 0 \neq 1 = g(101) = g(b)$.

Example 4. This is the end of example 2. Here, both conditions (3) and (4) are false because, for example, $f(g(x)) \neq x_1$ for $x = 121$ and $x = 201$, $a_1 \neq b_1$ and $g(a) = g(b)$ for $a = 000$ and $b = 121$, for $a = 020$ and $b = 121$. Moreover, immediately from the Table 4, it is seen that, for this g, there doesn't exist f with the property $f(g(x)) = x_1$ and it isn't possible to recover the value x_1 from the value $g(x)$. Also, it's impossible to make the condition (4) to become true in the way of choosing other values for x_3 in points $x \in M_\varepsilon$, since for any values a_3, b_3 of variable x_3, there exist some values a_1, a_2 and b_1, b_2 of the variables x_1, x_2 such that a_1, a_2 are arbitrary, b_1 is invertible modulo 3 and $b_2 = b_1^{-1}(a_1a_2 + a_3 - b_3) \bmod 3$ and then we will have what we need, namely: $a_1 \neq b_1$ and $g(a) = a_1a_2 + a_3 = b_1b_2 + b_3 = g(b)$. So, if for x we take the value $b' = 120$ instead of $b = 121$, then, from one side, for $a = 020$ and $b' = 120$, we will have what we want, namely: $a_1 = 0 \neq 1 = b'_1$ and $g(a) = 0 \neq 2 = g(b')$, and from another one, — unwanted fact, namely: $a_1 = 2 \neq 1 = b'_1$ and $g(a) = 2 = 2 = g(b')$ for $a = 210$ and $b' = 120$, etc.

6. Test for automaton cryptanalytical invertibility

Let q and s, α and β, δ and γ be the values from Q, X^*, X^τ respectively and $a_1a_2a_3$, $b_1b_2b_3 \in M_\varepsilon$ where a_1, a_2, a_3 and b_1, b_2, b_3 are the different values from $\{q, \alpha, \delta\}$ and $\{s, \beta, \gamma\}$ respectively such that if a_k is q, α or δ, then b_k is s, β or γ respectively, $k \in \{1, 2, 3\}$.

Theorem 1. The automaton A is cryptanalytically invertible of a type $(Q_1x_1Q_2x_2Q_3x_3, \upsilon(q, \alpha, \delta))$, that is, there exists a function f such that (2) is true, if and only if for $Q_1x_1Q_2x_2Q_3x_3$ there is an existential vector function ε such that the following formula is true:

$$\forall a_1a_2a_3 \in M_\varepsilon \, \forall b_1b_2b_3 \in M_\varepsilon(\alpha \neq \beta \Rightarrow ((\bar{\varphi}(\alpha\delta, q), \upsilon(q, \alpha, \delta)) \neq (\bar{\varphi}(\beta\gamma, s), \upsilon(s, \beta, \gamma)))). \quad (5)$$

Proof. The proposition under proof is a particular case of Lemma 1. ■

The theorem is the base for deciding by the following exhaustive search method if an automaton A is cryptanalytically invertible of a given type $(Q_1x_1Q_2x_2Q_3x_3, \upsilon(q, \alpha, \delta))$ or not:

1) For every possible existential vector function ε of the $ID = Q_1x_1Q_2x_2Q_3x_3$, generate the existential domain M_ε.
2) Apply Theorem 1 to M_ε, that is, verify whether (5) is true.
3) If for some ε, (5) is true for M_ε, the automaton A is cryptanalytically invertible of the type $(Q_1x_1Q_2x_2Q_3x_3, \upsilon(q, \alpha, \delta))$. Otherwise, that is, if for all existential functions ε of the ID, the condition (5) is false for M_ε, then the automaton A is not cryptanalytically invertible of this type.

7. Decision methods for function cryptanalytical invertibility problem

7.1. Exhaustive search

1) For every possible existential function $\varepsilon = \varepsilon_{j_1} \ldots \varepsilon_{j_s}$ where $\varepsilon_k : D_1 \times \ldots \times D_{k-1} \to D_k$, $k \in \{j_1, \ldots j_s\}$, generate the existential domain M_ε.
2) Apply Lemma 1 to M_ε, that is, verify whether (4) is true.
3) If for some ε, (4) is true for M_ε, the invertibility problem under consideration is positively solvable. Otherwise, that is, if for all ε, (4) is false for M_ε, then the invertibility problem has the negative solution.

7.2. Search by collision elimination

A pair (a, b) of words a and b in $D_1 \times \ldots \times D_n$ is called a collision if $a_{k_0} \neq b_{k_0}$ and $g(a) = g(b)$. We call the collision (a, b) a collision in a subset $U \subseteq D_1 \times \ldots \times D_n$ if $a, b \in U$. Also, we say that U has no collisions, or is free of collisions, if for every a and b in U the pair (a, b) isn't a collision. Further, collisions in an existential domain M_ε as depending on ε are called ε-collisions.

Theorem 2. There exists a function f satisfying (3) if and only if for some ε, the existential domain M_ε has no ε-collisions.

Proof. According to lemma 1,
$\exists f((3)) \Leftrightarrow \exists\varepsilon((4)) \Leftrightarrow \forall a, b \in M_\varepsilon\,(a_{k_0} \neq b_{k_0} \Rightarrow g(a) \neq g(b)) \Leftrightarrow \forall a, b \in M_\varepsilon\,(a_{k_0} = b_{k_0} \vee$
$\vee g(a) \neq g(b)) \Leftrightarrow \forall a, b \in M_\varepsilon \neg(a_{k_0} \neq b_{k_0}\ \&\ g(a) = g(b)) \Leftrightarrow (M_\varepsilon$ is free of ε-collisions). ■

Corollary 2. A function $g(x_1, \ldots, x_n)$ is invertible of a type $Q_1 x_1 \ldots Q_n x_n$ with respect to a variable x_{k_0}, $k_0 \in \{i_1, \ldots, i_r\}$, if and only if for some ε, the existential domain M_ε is free of ε-collisions.

For a and b in M_ε, we say as well that the pair (a, b) is a non-ε-collision if it is not a ε-collision, that is, if $a_{k_0} = b_{k_0}$ or $g(a) \neq g(b)$. The following operations are introduced in order to eliminate the ε-collisions from an existential domain M_ε and to get a new domain $M_{\varepsilon'}$ without ε'-collisions (if it is possible) or with other ε'-collisions (otherwise), so witnessing that the function g under consideration is respectively invertible or uninvertible of a given type with respect to a given variable.

Let $a = a_1 \ldots a_{j_1} \ldots a_{j_s} \ldots a_n$, $A = a_{j_1} a_{j_2} \ldots a_{j_s}$, $A' = a'_{j_1} a'_{j_2} \ldots a'_{j_s}$, and $b = b_1 \ldots b_{j_1}$ $\ldots b_{j_s} \ldots b_n$, $B = b_{j_1} b_{j_2} \ldots b_{j_s}$, $B' = b'_{j_1} b'_{j_2} \ldots b'_{j_s}$. Define $a' = a_1 \ldots a_{j_1 - 1} a'_{j_1} \ldots a'_{j_s} a_{j_s+1} \ldots a_n$ and $b' = b_1 \ldots b_{j_1 - 1} b'_{j_1} \ldots b'_{j_s} b_{j_s+1} \ldots b_n$. We say that a' and b' are obtained by substituting A by A' and B by B', or A' for A and B' for B, and write $a' = a(A' \to A)$ and $b' = b(B' \to B)$ respectively. Now we can transform the ε-collision (a, b) in M_ε to a non-ε'-collision (a', b) in $M_{\varepsilon'}$ where $a' = a'_1 \ldots a'_n = a(A' \to A)$, $A' \neq A$, $g(a') \neq g(a)$, $\varepsilon' = \varepsilon'_{j_1} \ldots \varepsilon'_{j_s}$, and $\varepsilon'_k(a'_1 \ldots a'_{k-1}) = a'_k$ for each $k \in \{j_1, \ldots, j_s\}$. Analogously, ε-collision (a, b) in M_ε can be transformed to a non-ε'-collision (a, b') in $M_{\varepsilon'}$.

So, in the Example 2, we have ε-collisions (a, b) with $a = 121$ and $b \in \{000, 010, 020\}$ and (a, b) with $a = 201$ and $b \in \{101, 110\}$. In the case $D_3 = \{0, 1\}$ that we have it seems impossible to eliminate these ε-collisions without creating others. But if we correct this example and allow $D_3 = \{0, 1, 2\}$ like in the Example 1, we get a possibility to eliminate them at all by taking, for instance, $\varepsilon'_3(12) = \varepsilon'_3(20) = 2$. In this case ε-collisions $(a, b) =$ $= (121, b)$ and $(a, b) = (201, b)$ in M_ε are transformed to non-ε'-collisions $(122, b)$ and $(202, b)$ respectively in $M_{\varepsilon'}$. At the same time we can note that an elimination of a ε-collision by correcting an existential function ε can produce other collisions and complicate the process of recognizing whether there is an existential function ε without collisions in M_ε. Really, in our example we could eliminate the collisions $(121, b)$ for $b \in \{000, 010, 020\}$ by taking $\varepsilon'(12) = 0$ and obtain the new ε'-collisions $(120, b')$ where $b' \in \{210, 221\}$.

Nevertheless, the notion of ε-collision is very important in the cryptanalytical invertibility theory in many ways. It is enough to say that the exhaustive method above remains strong after changing the need of true condition (4) in it by the request for collision absence (Corollary 2). The requirements of collisions lack in M_ε follow from the need to have a recovering function f (Theorem 2) or invertibility property of g (Corollary 2).

7.3. Search by forward and back tracking

Further, we believe that on any finite set M under consideration a linear ordering relation $\leqslant$ (not greater than) is supposed to be given, and for any $a, b \in M$ we write $a < b$ (a less than b) if $a \leqslant b$ and $a \neq b$. This relation extends to Cartesian products of linearly ordered sets, for instance, as lexicographical ordering in the following way: $a_1a_2 \dots a_n < < b_1b_2 \dots b_n \Leftrightarrow a_i < b_i$ where i is determined from the conditions $a_1 = b_1, \dots, a_{i-1} = b_{i-1}$, $a_i < b_i$ and $i \in \{1, 2, \dots, n\}$, that is, i is the least number in $\{1, 2, \dots, n\}$ such that $a_i \neq b_i$ and $a_i < b_i$.

Now we introduce some additional notation and notions, namely $I = \{i_1, \dots, i_r\}$, $i_1 < < \dots < i_r$, $J = \{j_1, \dots, j_s\}$, $j_1 < \dots < j_s$, $n = r + s$, $I \cap J = \varnothing$, $I \cup J = \{1, 2, \dots, n\}$, $D(I) = \{d_1, \dots, d_{m_r}\} = D_{i_1} \times \dots \times D_{i_r}$, $D(J) = \{e_1, \dots, e_{m_s}\} = D_{j_1} \times \dots \times D_{j_s}$, $u_{ij} = = d_i \otimes e_j = a_1a_2 \dots a_n$ where $a_{i_1} \dots a_{i_r} = d_i$ and $a_{j_1} \dots a_{j_s} = e_j$, $i = 1, \dots, m_r$, $j = 1, \dots, m_s$, $m = m_r \cdot m_s$, $\{u_1, u_2, \dots, u_m\} = \{u_{ij} : i = 1, \dots, m_r, j = 1, \dots, m_s\}$, $U_t = \{u_1, \dots, u_t\}$, $1 \leqslant t \leqslant m$.

So, here we consider each vector $a = a_1a_2 \dots a_n \in D_1 \times \dots \times D_n$ as a blend $d \otimes e$ of a vector $d = a_{i_1}a_{i_2} \dots a_{i_r} \in D(I)$ and a vector $e = a_{j_1}a_{j_2} \dots a_{j_s} \in D(J)$ and write $a = d \otimes e$.

For every $a = a_1 \dots a_n \in D_1 \times \dots \times D_n$ and $b = b_1 \dots b_n \in D_1 \times \dots \times D_n$ we say that a and b are equivalent if for each $k \in J$ we have $a_1 \dots a_{k-1} = b_1 \dots b_{k-1} \Rightarrow a_k = b_k$. It is clear that this notion here comes from the functionality of the coordinates ε_k of the existential vector function ε. When $a_1 \dots a_{k-1} = b_1 \dots b_{k-1}$ and $a_k \neq b_k$ for some $k \in J$, we call the pair (a, b) inequality, and the replacement in a and b the elements a_k and b_k by one and the same element from $D(J)$ is called an inequality elimination. We also say that a subset $U \subseteq D_1 \times \dots \times D_n$, particularly M_ε, is an equivalence class if all the elements in it are equivalent each other. It is not difficult to see that any such subset is quite simply transformed into an equivalence class by applying, possibly repeatedly, the inequality elimination to pairs of elements in it.

Here in reality, we consider the problem to determine an existential function $\varepsilon : D(I) \to \to D(J)$, that is, for every $d \in D(I)$ to choose an element $\varepsilon(d) \in D(J)$ so that the set $U_\varepsilon = \{d \otimes \varepsilon(d) : d \in D(I)\}$ is namely an equivalence class without collisions (further shortly called ECwC) or to show that such a function ε doesn't exist. The first outcome means that the function $g(x_1, \dots, x_n)$ is invertible of a given type $Q_1x_1 \dots Q_nx_n$ with respect to a given variable x_{k_0}, the second one – that g isn't invertible of this type. The correctness of this decision of the problem is provided by a correct searching an ECwC U_ε with the help of so called forwardtracking (FT) and backtracking (BT) operations correctly defined below and used on the space $D(I) \times D(J)$.

FT: given ECwC $U_t = \{u_1, \dots, u_t\}$, $1 \leqslant t < m_r$; take $e \in D(J)$ and $u_{t+1} = d_{t+1} \otimes e$ so that u_{t+1} is equivalent to each of $u_1, \dots, u_t$ and is not in collision with any of them; define $\mathrm{FT}(U_t) = U_{t+1} = \{u_1, \dots, u_t, u_{t+1}\}$. It is clear that if such an e exists, then FT transforms ECwC U_t into ECwC U_{t+1}. Otherwise, the forwardtracking from ECwC U_t into ECwC U_{t+1} is impossible and backtracking from U_t can be accomplished according to the following general or particular definitions.

BT (general): given ECwC $U_t = \{u_1, \dots, u_t\}$, $t \geqslant 1$, $d_{t+1} \in D(I)$ and for each $j \in J$ there is $t_j \in \{1, \dots, t\}$ such that $d_{t+1} \otimes e_j$ isn't equivalent to $u_{t_j} = d_{t_j} \otimes e_{t_j}$ or is in a collision with it. This means that given U_t is impossible to transform by FT into U_{t+1} with given $d_{t+1} \in D(I)$ and any $e_j \in D(J)$ in $u_{t+1} = d_{t+1} \otimes e_j$. In application to these data the backtracking generally consists in taking a specific e from $D(J)$ for $u_{t+1} = d_{t+1} \otimes e$ as well as some $j \in J$ and replacing in U_t the points $u_{t_j} = d_{t_j} \otimes e_{t_j}$ by some other ones $u'_{t_j} = d_{t_j} \otimes e'_{t_j}$ which are equivalent to u_{t+1}, to each other u'_{t_j} and to the rest of U_t and

aren't in collision with them. The set $U_{t+1} = U'_t \cup \{u_{t+1}\}$, where u_{t+1} and U'_t are obtained in the described way in the place of d_{t+1} and U_t respectively, is defined as a result of the backtracking from U_t and d_{t+1}, namely: $U_{t+1} = \mathrm{BT}(U_t \cup \{d_{t+1}\})$.

For instance, in Example 1 let $t = 3$, $U_t = \{u_{t-2}, u_{t-1}, u_t\}$, $u_{t-2} = 010$, $u_{t-1} = 101$, $u_t = 120$, $d_{t+1} = 20$, $D(J) = \{e_1, e_2, e_3\}$, $e_1 = 0, e_2 = 1, e_3 = 2$. We can see that every possible value u_{t+1} is in collision with some $u_{t_j} \in \{u_{t-2}, u_{t-1}, u_t\}$. Indeed, if $u_{t+1} = d_{t+1} \otimes e_1$, then $u_{t+1} = 200$ and is in collision with $010 = u_{t-2}$; in analogous way, we can show that if $u_{t+1} = d_{t+1} \otimes e_2 = 201$, then it is in collision with $101 = u_{t-1}$, and if $u_{t+1} = d_{t+1} \otimes e_3 = 202$, then it is in collision with $120 = u_t$. At the same time the set U_t itself is an ECwC, that is, all the points u_{t-2}, u_{t-1}, u_t in U_t are equivalent each other and there are no collisions between them. The aim of backtracking operation is to attach a next data d_{t+1} to a given ECwC U_t as a component of its next member $u_{t+1} = d_{t+1} \otimes e$ admitting the choice of any value e from $D(J)$ as the existential component in u_{t+1} and any correction of existential components in other members of U_t with the only condition — preserving the ECwC properties of the set under backtracking. The choice of the component e in u_{t+1} and the correction of the existential components in members of U_t aren't one-valued and are possible in many different ways. In our instance we have taken $e = 2$ and 0, 1, 2 as existential values in u_{t-2}, u_{t-1}, u_t respectively. So, $U_{t+1} = \mathrm{BT}(U_t \cup \{d_{t+1}\}) = \{u_{t-2}, u_{t-1}, u'_t, u_{t+1}\} = \{010, 101, 122, 202\}$. It is directly verified that this set is ECwC what is required. The following is a result of another variant of backtracking for the same instance: $U_{t+1} = \{u'_{t-2}, u'_{t-1}, u_t, u_{t+1}\} = \{011, 102, 120, 200\}$.

BT (particular): as in general BT, given ECwC $U_t = \{u_1, \ldots, u_t\}$, $d_{t+1} \in D(I)$ and for each $j \in J$ there is $t_j \in \{1, \ldots, t\}$ such that $d_{t+1} \otimes e_j$ isn't equivalent to $u_{t_j} = d_{t_j} \otimes e_{t_j}$ or is in a collision with it. The particular application of the backtracking to these data consists in taking a free (for the first time) existential value e'_t from $D(J)$ for $u'_t = d_t \otimes e'_t$ and e'_{t+1} from $D(J)$ for $u_{t+1} = d_{t+1} \otimes e'_{t+1}$ so that u'_t is equivalent to each of $u_1, \ldots, u_{t-1}$ and without collisions with them, u_{t+1} is equivalent to each of $u_1, \ldots, u_{t-1}, u'_t$ and without collisions with them. The set $U_{t+1} = \{u_1, \ldots, u_{t-1}, u'_t, u_{t+1}\}$ is defined to be the result of the bachtracking from U_t and d_{t+1}, that is, $U_{t+1} = \mathrm{BT}(U_t \cup \{d_{t+1}\})$.

In case, when such an e'_{t+1} doesn't exist, another free e'_t is chosen in $D(J)$ for $u'_t = d_t \otimes e'_t$. If e'_t doesn't exist for choosing a needed e'_{t+1}, then another free e'_{t-1} is chosen in $D(J)$ for $u'_{t-1} = d_{t-1} \otimes e'_{t-1}$ and so on.

After executing BT and constructing U_{t+1} forwardtracking is tried to be applied to U_{t+1}. The computation ends when the beginning point without free existential values is achieved. The analysed function g is adopted to be invertible of a certain type iff an ECwC of the maximal size m_r is demonstrated by this computation.

The author expresses his thanks to I. A. Pankratova for a valuable observation and important correction of the paper.

REFERENCES

1. *Agibalov G. P.* Cryptanalytic concept of finite automaton invertibility with finite delay. Prikladnaya Diskretnaya Matematika. 2019, no. 44, pp. 34–42.
2. *Rasiowa H.* Introduction to Modern Mathematics. Amsterdam; London, North-Holland Publishing Company; Warszawa, PWN, 1973. 339 p.
3. *Agibalov G. P. and Belyaev V. A.* Tehnologiya Resheniya Kombinatorno-logicheskih Zadach Metodom Sokraschyonnogo Obhoda Dereva Poiska [Technology for Solving Combinatorial-Logical Problems by the Method of Shortened Search Tree Traversal]. Tomsk, TSU Publ., 1981. 126 p. (in Russian)

4. *Christofides H.* Graph Theory. An algorithmic Approach. New York; London; San Francisco, Academic Press, 1975.
5. *Zakrevskij A., Pottosin Yu., and Cheremisinova L.* Combinatorial Algorithms of Discrete Mathematics. Tallinn, TUT Press, 2008. 193 p.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ НАДЁЖНОСТИ ВЫЧИСЛИТЕЛЬНЫХ И УПРАВЛЯЮЩИХ СИСТЕМ

УДК 519.718.7

МЕТОД ПОСТРОЕНИЯ ЛЕГКО ДИАГНОСТИРУЕМЫХ СХЕМ ИЗ ФУНКЦИОНАЛЬНЫХ ЭЛЕМЕНТОВ ОТНОСИТЕЛЬНО ЕДИНИЧНЫХ НЕИСПРАВНОСТЕЙ[1]

К. А. Попков

Институт прикладной математики им. М. В. Келдыша РАН, г. Москва, Россия

Предложен метод синтеза схем из функциональных элементов в произвольном функционально полном базисе, реализующих заданные булевы функции и допускающих единичные диагностические тесты малой длины относительно константных и/или инверсных неисправностей на входах и/или выходах элементов при выполнении определённых начальных условий, связанных с существованием коротких единичных проверяющих тестов для схем в том же базисе при таких же неисправностях. На основании этого метода получены новые верхние оценки длин минимальных единичных диагностических тестов для схем из функциональных элементов в некоторых базисах при некоторых неисправностях элементов.

Ключевые слова: *схема из функциональных элементов, булева функция, константная неисправность, инверсная неисправность, единичный проверяющий тест, единичный диагностический тест.*

DOI 10.17223/20710410/46/4

A METHOD FOR CONSTRUCTING LOGIC NETWORKS ALLOWING SHORT SINGLE DIAGNOSTIC TESTS

K. A. Popkov

Keldysh Institute of Applied Mathematics, Moscow, Russia

E-mail: kirill-formulist@mail.ru

Let $D^B(f)$ be the least length of a single diagnostic test for irredundant logic networks consisting of logic gates in a functionally complete basis B, implementing given Boolean function f, and having at most one fixed type fault at inputs or outputs of gates. Let $D^B(n) = \max D^B(f)$, where the maximum is taken over all Boolean functions f in n variables. Consider the bases $B_{(1)} = \{x_1x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3, \overline{x}\}$, $B_{(2)} = \{x\&y, x \oplus y, 1\}$, $B_{(3)} = \{\eta(\tilde{x}^4), x_1 \sim x_2, \overline{x}, 0\}$, where $\eta(\tilde{x}^4)$ is an arbitrary non-self-dual Boolean function taking the value α on the tuple $(\alpha, \alpha, \alpha, \alpha)$ and the value $\overline{\alpha}$ on all 4-tuples adjacent with it, for each $\alpha \in \{0, 1\}$; $B_{(4)} = \{x\&y, \overline{x}, x \oplus y \oplus z\}$. The following inequalities are obtained:
1) $D^{B_{(1)}}(n) \leqslant 3$ for each $n \geqslant 0$ under stuck-at-0 faults at inputs and outputs of gates;
2) $D^{B_{(2)}}(n) \leqslant 3$ for each $n \geqslant 0$ under stuck-at-1 faults at outputs of gates;

[1]Работа выполнена при поддержке гранта РНФ, проект № 19-71-30004.

3) $D^{B_{(3)}}(n) \leqslant 4$ for each $n \geqslant 0$ under stuck-at-0 and stuck-at-1 faults at inputs and outputs of gates;
4) $D^{B_{(4)}}(n) \leqslant 4$ for each $n \geqslant 1$ under stuck-at-0 and stuck-at-1 faults at outputs of gates;
5) $D^{B_{(2)}}(n) \leqslant 3$ for each $n \geqslant 0$ under inverse faults at inputs and outputs of gates.
All inequalities are proved by the method of synthesis of logic networks implementing given Boolean functions and allowing short single diagnostic tests, based on the existence of short single fault detection tests for networks in the same basis under the same faults.

Keywords: *logic network, Boolean function, stuck-at fault, inverse fault, single fault detection test, single diagnostic test.*

Введение

В работе рассматривается задача синтеза легкотестируемых схем, реализующих заданные булевы функции. Логический подход к тестированию электрических схем предложен И. А. Чегис и С. В. Яблонским в [1]; этот подход также применим к тестированию схем из функциональных элементов [2 – 4]. Пусть имеется схема из функциональных элементов S с одним выходом, реализующая булеву функцию $f(\tilde{x}^n)$, где $\tilde{x}^n = (x_1, \ldots, x_n)$. Представим, что под воздействием некоторого источника неисправностей один или несколько входов и/или выходов элементов схемы S могут перейти в неисправное состояние. В результате данная схема вместо исходной функции $f(\tilde{x}^n)$ будет реализовывать некоторую булеву функцию $g(\tilde{x}^n)$, вообще говоря, отличную от f. Все такие функции $g(\tilde{x}^n)$, получающиеся при всевозможных допустимых для рассматриваемой задачи неисправностях элементов схемы S, называются *функциями неисправности* данной схемы.

Введём следующие определения [2 – 4]. *Проверяющим тестом* для схемы S называется такое множество T наборов значений переменных $x_1, \ldots, x_n$, что для любой отличной от $f(\tilde{x}^n)$ функции неисправности $g(\tilde{x}^n)$ схемы S в T найдётся набор $\tilde{\sigma}$, на котором $f(\tilde{\sigma}) \neq g(\tilde{\sigma})$. *Диагностическим тестом* для схемы S называется такое множество T наборов значений переменных $x_1, \ldots, x_n$, что T является проверяющим тестом и, кроме того, для любых двух различных функций неисправности $g_1(\tilde{x}^n)$ и $g_2(\tilde{x}^n)$ схемы S в T найдётся набор $\tilde{\sigma}$, на котором $g_1(\tilde{\sigma}) \neq g_2(\tilde{\sigma})$. Число наборов в T называется *длиной* теста. В качестве тривиального диагностического (и проверяющего) теста длины 2^n для схемы S всегда можно взять множество, состоящее из всех двоичных наборов длины n. Тест называется *полным*, если в схеме могут быть неисправны сколько угодно входов/выходов элементов, и *единичным*, если в схеме может быть неисправно не более одного входа/выхода элемента. Единичные тесты обычно рассматривают для *неизбыточных схем* [4, с. 110–111], т. е. для таких схем, в которых любая допустимая неисправность любого одного элемента приводит к функции неисправности, отличной от исходной функции, реализуемой данной схемой.

Любое множество булевых функций будем называть *базисом*.

Пусть зафиксирован вид неисправностей элементов, B — произвольный функционально полный базис и T — единичный диагностический тест (ЕДТ) для некоторой схемы S в базисе B. Обозначим: $D^B(T)$ — длина теста T; $D^B(S) = \min D^B(T)$, где минимум берётся по всем ЕДТ T для схемы S; $D^B(f) = \min D^B(S)$, где минимум берётся по всем неизбыточным схемам S в базисе B, реализующим функцию f; $D^B(n) = \max D^B(f)$, где максимум берётся по всем булевым функциям f от n перемен-

ных, для которых определено значение $D^B(f)$. Функция $D^B(n)$ называется *функцией Шеннона* длины ЕДТ.

Перечислим основные результаты, касающиеся ЕДТ для схем из функциональных элементов. Класс допустимых неисправностей функциональных элементов ограничим константными и/или инверсными неисправностями на входах и/или выходах элементов. Константная неисправность на входе (выходе) функционального элемента означает, что значение на этом его входе (на его выходе) становится равно некоторой булевой константе. Неисправности на входах и/или выходах элементов называются однотипными константными типа p, если эта константа одна и та же для каждого неисправного входа/выхода элемента и равна p, и произвольными константными, если эта константа может быть равна как 0, так и 1 для каждого неисправного входа/выхода элемента независимо от неисправностей других входов/выходов элементов. Инверсная неисправность на входе (выходе) функционального элемента означает, что значение на этом его входе (на его выходе) становится противоположным значению на этом же его входе (на его выходе) в случае, когда данный элемент исправен.

Для удобства под буквой D будем ставить символы «$0,1$», «0», «1» или «Inv» в случаях, когда в схемах допускаются соответственно произвольные константные неисправности, однотипные константные неисправности типа 0, однотипные константные неисправности типа 1 или инверсные неисправности элементов, а после них — символы «(IO)», «(I)» или «(O)» в случаях, когда в схемах допускаются неисправности соответственно на входах и выходах, только на входах или только на выходах элементов. Вполне разумно предполагать, что если в базисе содержится булева константа α, то у элемента, её реализующего, нет входов и не может быть неисправности типа α на его выходе.

Сначала перечислим результаты для ЕДТ при константных неисправностях на выходах элементов, затем — при инверсных неисправностях на выходах элементов, а в конце — при неисправностях на входах и выходах либо только на входах элементов.

В [4, с. 113, теорема 9] с использованием идей С. В. Яблонского установлено, что для любого полного базиса B функция $D^B_{0,1\,(\mathrm{O})}(n)$ асимптотически не превосходит $2^{n+1}/n$; аналогично можно показать, что $D^B_{p\,(\mathrm{O})}(n) \lesssim 2^n/n$, $p = 0, 1$. Для стандартного базиса $B_1 = \{\&, \vee, \neg\}$ Н. П. Редькин в [5] получил оценку $D^{B_1}_{p\,(\mathrm{O})}(n) \leqslant 2n + 1$, где $p = 0$ или 1. Впоследствии эта оценка была улучшена в [6], где, в частности, доказано, что $D^{B_1}_{p\,(\mathrm{O})}(n) = 2$ при $n \geqslant 2$. В работе [7] для базиса Жегалкина $B_2 = \{\&, \oplus, 1\}$ установлено равенство $D^{B_2}_{0\,(\mathrm{O})}(n) = 2$ при $n \geqslant 2$. В [8], в частности, для любого полного конечного базиса B при n, большем максимального числа существенных переменных у функций из B, доказаны неравенства $D^B_{p\,(\mathrm{O})}(n) \geqslant 2$, где $p = 0$ или 1, и $D^B_{0,1\,(\mathrm{O})}(n) \geqslant 3$. Д. С. Романов и Е. Ю. Романова в [9] получили оценку $D^{B_2}_{0,1\,(\mathrm{O})}(n) \leqslant 22$, а также доказали существование базиса B_3, состоящего из булевых функций от не более чем девяти переменных, с условием $D^{B_3}_{0,1\,(\mathrm{O})}(n) \leqslant 6$. Одним из результатов работы [10] является доказательство существования базиса B_4, состоящего из одной булевой функции от шести переменных, с условием $D^{B_4}_{0,1\,(\mathrm{O})}(n) = 3$ при $n \geqslant 2$.

С. В. Коваценко установил, что $D^{B_2}_{\mathrm{Inv}\,(\mathrm{O})}(n) \leqslant n + 1$ [11]. Д. С. Романов в [12, 13] получил равенства $D^{B_1}_{\mathrm{Inv}\,(\mathrm{O})}(n) = 2$ при $n \geqslant 2$ и $D^{B_2}_{\mathrm{Inv}\,(\mathrm{O})}(n) = 1$ соответственно, второе из которых улучшает результат работы [11]. И. Г. Любич совместно с Д. С. Романовым в [14] получили оценку $D^B_{\mathrm{Inv}\,(\mathrm{O})}(n) \leqslant 4$, где B — либо произвольный полный базис, содержащий хотя бы одну из функций $x\&y$, $x \vee y$, либо один из базисов $\{\overline{x\&y}\}$, $\{\overline{x \vee y}\}$.

Н. П. Редькин в [15] установил неравенство $D^{B_1}_{p\,(\mathrm{I})}(n) \lesssim 4\left(2^{\lfloor n/2\rfloor}+2^{\lceil n/2\rceil-1}\right)$ для $p=0,1$. В работе [16], в частности, доказано существование такой булевой функции ψ от четырёх переменных, что для базиса $B_5=\{\psi,\overline{\psi}\}$ выполнены соотношения $D^{B_5}_{0\,(\mathrm{IO})}(n)=2$ и $D^{B_5}_{0\,(\mathrm{I})}(n)=1$, а для базиса $B^*_5=\{\psi^*,\overline{\psi^*}\}$ — соотношения $D^{B^*_5}_{1\,(\mathrm{IO})}(n)=2$ и $D^{B^*_5}_{1\,(\mathrm{I})}(n)=1$, где ψ^* — двойственная к ψ булева функция. В [17], в том числе, доказано существование базиса B_6, состоящего из булевых функций от не более чем шести переменных, в котором $D^{B_6}_{0,1\,(\mathrm{IO})}(n)=4$ при $n\geqslant 3$ и $D^{B_6}_{0,1\,(\mathrm{I})}(n)=2$.

В данной работе предложен метод построения схем из функциональных элементов в произвольном функционально полном базисе, реализующих заданные булевы функции и допускающих короткие ЕДТ, основанный на существовании схем в том же базисе, допускающих короткие единичные проверяющие тесты (ЕПТ) при таких же неисправностях элементов (теорема 1). С использованием этого метода получен ряд константных верхних оценок функций Шеннона длины ЕДТ для схем в разных базисах при различных видах неисправностей элементов (теоремы 2–6).

Введём обозначения $\tilde{0}^d=\underbrace{0,\ldots,0}_{d}$, $\tilde{1}^d=\underbrace{1,\ldots,1}_{d}$, где $d\in\mathbb{N}\cup\{0\}$. В случае $d=0$ они обозначают пустую строку: например, $(\tilde{1}^n,\tilde{0}^0)=(\tilde{1}^n)$.

1. Описание метода

Пусть M — произвольное множество двоичных наборов длины n. Через $I_M(\tilde{x}^n)$ будем обозначать булеву функцию, принимающую значение 1 на наборах из множества M и значение 0 на всех остальных наборах. Два двоичных набора одинаковой длины называются *соседними*, если они различаются ровно в одной компоненте.

Пусть $f(\tilde{x}^n)$ — булева функция, T — множество (некоторых) двоичных наборов длины n, $\alpha\in\{0,1\}$. Будем говорить, что функция f *обладает* (T,α)*-свойством*, если существует двоичный набор длины n, не принадлежащий множеству T, на котором данная функция принимает значение α.

Пусть зафиксирован вид неисправностей функциональных элементов: константные (однотипные типа $p\in\{0,1\}$ либо произвольные) и/или инверсные неисправности на входах и/или выходах элементов.

Теорема 1. Пусть для неконстантной булевой функции $f(\tilde{x}^n)$ и функционально полного базиса B существуют такие $m\in\mathbb{N}$, двоичные наборы $\tilde{c}_1=(c_{1,1},\ldots,c_{1,m})$, $\tilde{c}_0=(c_{0,1},\ldots,c_{0,m})$, булева функция $\varphi(\tilde{x}^m)$, множество $T=\{\tilde{\sigma}_1,\ldots,\tilde{\sigma}_l\}$ двоичных наборов длины n, где $|T|<2^n$; подмножества $M_1,\ldots,M_m$ множества T и двоичные наборы $\tilde{\pi}_1,\ldots,\tilde{\pi}_l$ длины m, что выполнены следующие условия:

1) для любого $i\in\{1,\ldots,m\}$ функцию $f_i(\tilde{x}^n)=(c_{1,i}f\oplus c_{0,i}\overline{f}\oplus I_{M_i})(\tilde{x}^n)$ можно реализовать неизбыточной схемой S_i в базисе B, для которой множество T является ЕПТ;
2) $\tilde{\pi}_j=(f_1(\tilde{\sigma}_j),\ldots,f_m(\tilde{\sigma}_j))$ для любого $j\in\{1,\ldots,l\}$;
3) функцию $\varphi(\tilde{x}^m)$ можно реализовать схемой S_φ в базисе B, неизбыточной и допускающей ЕПТ T_φ относительно неисправностей рассматриваемого вида, кроме, быть может, константных неисправностей на выходе выходного элемента схемы, где

$$T_\varphi=\begin{cases}\{\tilde{\pi}_1,\ldots,\tilde{\pi}_l,\tilde{c}_a\}, & \text{если функция } f \text{ не обладает } (T,\overline{a})\text{-свойством}\\ & \text{для некоторого } a\in\{0,1\},\\ \{\tilde{\pi}_1,\ldots,\tilde{\pi}_l,\tilde{c}_1,\tilde{c}_0\} & \text{иначе;}\end{cases}$$

4) функция $\varphi(\tilde{x}^m)$ принимает значение α на наборе $\tilde{c}_\alpha$ и всех соседних с ним наборах для любого $\alpha \in \{0,1\}$, такого, что функция f обладает (T,α)-свойством;
5) функция $\varphi(\tilde{x}^m)$ принимает значение $f(\tilde{\sigma}_j)$ на наборе $\tilde{\pi}_j$ и значение $\overline{f}(\tilde{\sigma}_j)$ на всех наборах, соседних с набором $\tilde{\pi}_j$, для любого $j \in \{1,\dots,l\}$.

Тогда

$$D^B(f) \leqslant \begin{cases} |T|+1, & \text{если функция } f \text{ не обладает } (T,\overline{a})\text{-свойством} \\ & \text{для некоторого } a \in \{0,1\}, \\ |T|+2 & \text{иначе.} \end{cases}$$

Доказательство. Входы схемы S_φ соединим с выходами схем $S_1,\dots,S_m$ (первый вход — с выходом схемы S_1, …, m-й вход — с выходом схемы S_m). Полученную схему с n входами, на которые подаются переменные $x_1,\dots,x_n$, и выходом, совпадающим с выходом схемы S_φ, обозначим через S (рис. 1); очевидно, что она является схемой в базисе B.

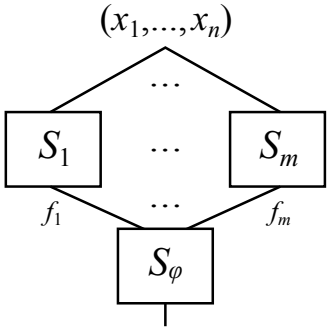


Рис. 1. Схема S

Докажем, что данная схема при отсутствии в ней неисправностей реализует функцию $f(\tilde{x}^n)$. На выходах подсхем $S_1,\dots,S_m$ по условию 1 реализуются функции $f_1(\tilde{x}^n),\dots,f_m(\tilde{x}^n)$ соответственно. На любом таком двоичном наборе $\tilde{\tau}_\alpha$ длины n, не принадлежащем множеству T (а значит, и множеству M_i), что $f(\tilde{\tau}_\alpha)=\alpha$, где $\alpha \in \{0,1\}$, на выходе подсхемы S_i, $i=1,\dots,m$, возникнет значение

$$f_i(\tilde{\tau}_\alpha) = c_{1,i} f(\tilde{\tau}_\alpha) \oplus c_{0,i}\overline{f}(\tilde{\tau}_\alpha) \oplus I_{M_i}(\tilde{\tau}_\alpha) = c_{1,i}\alpha \oplus c_{0,i}\overline{\alpha} \oplus 0 = c_{\alpha,i}.$$

Тогда на входы подсхемы S_φ поступит набор $\tilde{c}_\alpha$, а на её выходе, т. е. на выходе схемы S, возникнет значение $\alpha = f(\tilde{\tau}_\alpha)$ в силу условий 3 и 4. Далее, для любого $j \in \{1,\dots,l\}$ на наборе $\tilde{\sigma}_j$ на выходах подсхем $S_1,\dots,S_m$ возникнут значения $f_1(\tilde{\sigma}_j),\dots,f_m(\tilde{\sigma}_j)$ соответственно, поэтому на входы подсхемы S_φ поступит набор $\tilde{\pi}_j$, а на выходе схемы S возникнет значение $f(\tilde{\sigma}_j)$ в силу условий 2, 3, 5. Тем самым показано, что на любом двоичном наборе длины n схема S выдаёт такое же значение, как и функция $f(\tilde{x}^n)$, т. е. реализует эту функцию, что и требовалось доказать.

Найдём все возможные функции неисправности схемы S при неисправности в ней ровно одного входа/выхода элемента и докажем, что она неизбыточна. Пусть сначала неисправный элемент содержится в какой-то подсхеме S_i, $i \in \{1,\dots,m\}$. На любом входном наборе схемы S среди всех значений на входах подсхемы S_φ, очевидно, может измениться только значение на её i-м входе. Поэтому на любом таком наборе $\tilde{\tau}_\alpha$ длины n, не принадлежащем множеству T, что $f(\tilde{\tau}_\alpha)=\alpha$, где $\alpha \in \{0,1\}$, на входы подсхемы S_φ поступит либо набор $\tilde{c}_\alpha$, либо набор, отличающийся от указанного

только в i-й компоненте. Тогда на выходе схемы S возникнет значение $\alpha = f(\tilde{\tau}_\alpha)$ в силу условий 3 и 4. Далее, множество T является ЕПТ для неизбыточной схемы S_i по условию 1, поэтому рассматриваемую неисправность можно обнаружить на каком-то наборе $\tilde{\sigma}_j \in T$, где $j \in \{1, \ldots, l\}$. Тогда на этом наборе на входы подсхемы S_φ поступит набор, отличающийся от набора $\tilde{\pi}_j$ только в i-й компоненте, а на выходе схемы S возникнет значение $\overline{f}(\tilde{\sigma}_j)$ в силу условий 3 и 5.

Тем самым показано, что любая функция неисправности схемы S, возникающая при неисправности входа/выхода какого-то элемента в одной из подсхем $S_1, \ldots, S_m$, на всех наборах длины n, не принадлежащих множеству T, принимает такое же значение, как и функция f, а хотя бы на одном наборе из множества T принимает значение, отличное от значения функции f на этом наборе. Нетрудно заметить, что любую такую функцию неисправности можно представить в виде $(f \oplus I_{T'})(\tilde{x}^n)$, где T' — некоторое непустое подмножество множества T. Этот вид является частным случаем (при $b_1 = 1$, $b_0 = 0$ и $T' \neq \varnothing$) более общего вида

$$(b_1 f \oplus b_0 \overline{f} \oplus I_{T'})(\tilde{x}^n), \qquad (*)$$

в котором $b_1, b_0 \in \{0, 1\}$ и $T' \subseteq T$.

Пусть теперь неисправен какой-то элемент в подсхеме S_φ. Если имеет место константная неисправность на выходе её выходного элемента, то на этом выходе, а значит, и на выходе схемы S, реализуется некоторая булева константа β, которая при этом отлична от функции f по условию теоремы и имеет вид $(*)$: действительно,

$$\beta \equiv \beta f \oplus \beta \overline{f} \oplus I_\varnothing.$$

Рассмотрим любую другую неисправность одного входа/выхода элемента в подсхеме S_φ. Пусть данная подсхема при этом реализует булеву функцию $h(\tilde{x}^m)$ от своих входов. Все элементы в подсхемах $S_1, \ldots, S_m$ исправны, поэтому на любом входном наборе схемы S на входы подсхемы S_φ поступят «правильные» значения. Тогда на любом таком наборе $\tilde{\tau}_\alpha$ длины n, не принадлежащем множеству T, что $f(\tilde{\tau}_\alpha) = \alpha$, где $\alpha \in \{0, 1\}$, на входы указанной подсхемы поступит набор $\tilde{c}_\alpha$, а на выходе схемы S возникнет значение $h(\tilde{c}_\alpha)$, которое обозначим через b_α (по крайней мере, один двоичный набор длины n не принадлежит множеству T в силу условия $|T| < 2^n$). Далее, при подаче на входы схемы S наборов $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_l$ на входы подсхемы S_φ поступят наборы $\tilde{\pi}_1, \ldots, \tilde{\pi}_l$ соответственно. Таким образом, при подаче на входы схемы S некоторых наборов на входы подсхемы S_φ поступят все наборы из множества T_φ в силу определения этого множества. Функцию h можно отличить от функции φ на наборах из множества T_φ по условию 3, поэтому рассматриваемая неисправность схемы S будет обнаружена хотя бы на одном её входном наборе.

Тем самым показано, что при рассматриваемой неисправности одного элемента в подсхеме S_φ функция неисправности $g(\tilde{x}^n)$ схемы S на любом таком наборе $\tilde{\tau}_\alpha$ длины n, не принадлежащем множеству T, что $f(\tilde{\tau}_\alpha) = \alpha$, где $\alpha \in \{0, 1\}$, принимает значение b_α. Нетрудно заметить, что функцию g можно представить в виде $(*)$, где

$$T' = \{\tilde{\delta} \in T : g(\tilde{\delta}) = b_1 f(\tilde{\delta}) \oplus b_0 \overline{f}(\tilde{\delta}) \oplus 1\}.$$

В итоге получаем, что любая функция неисправности схемы S представима в виде $(*)$, где $b_1, b_0 \in \{0, 1\}$ и T' — некоторое подмножество множества T. Кроме того, из приведённых рассуждений вытекает, что значения функции f и любой функции неисправности схемы S различаются хотя бы на одном наборе, поэтому данная схема неизбыточна.

Докажем, что она допускает ЕДТ длины не более

$$\begin{cases} |T|+1, & \text{если функция } f \text{ не обладает } (T,\overline{a})\text{-свойством} \\ & \text{для некоторого } a \in \{0,1\}, \\ |T|+2 & \text{иначе;} \end{cases}$$

отсюда будет следовать справедливость теоремы 1.

Заметим, что функция f также имеет вид $(*)$ при $b_1 = 1$, $b_0 = 0$ и $T' = \varnothing$. Пусть g и $\hat{g}$ — две произвольные различные булевы функции вида $(*)$, причём $g(\tilde{x}^n) = (b_1 f \oplus b_0\overline{f} \oplus I_{T'})(\tilde{x}^n)$, $\hat{g}(\tilde{x}^n) = (\hat{b}_1 f \oplus \hat{b}_0\overline{f} \oplus I_{\hat{T}'})(\tilde{x}^n)$, где $b_1, b_0, \hat{b}_1, \hat{b}_0 \in \{0,1\}$ и $T', \hat{T}' \subseteq T$. В силу условия $|T| < 2^n$ существует хотя бы один двоичный набор длины n, не принадлежащий множеству T. Пусть значение функции f на нём равно a; обозначим указанный набор через $\tilde{\tau}_a$, в результате имеем

$$g(\tilde{\tau}_a) = b_1 f(\tilde{\tau}_a) \oplus b_0\overline{f}(\tilde{\tau}_a) \oplus I_{T'}(\tilde{\tau}_a) = b_1 a \oplus b_0\overline{a} \oplus 0 = b_a; \tag{1}$$

$$\hat{g}(\tilde{\tau}_a) = \hat{b}_1 f(\tilde{\tau}_a) \oplus \hat{b}_0\overline{f}(\tilde{\tau}_a) \oplus I_{\hat{T}'}(\tilde{\tau}_a) = \hat{b}_1 a \oplus \hat{b}_0\overline{a} \oplus 0 = \hat{b}_a. \tag{2}$$

Рассмотрим два случая:

1. На любом наборе длины n, не принадлежащем множеству T, функция f принимает значение a. Тогда эта функция не обладает $(T,\overline{a})$-свойством, поэтому надо доказать неравенство $D^B(f) \leqslant |T|+1$. Если $b_a \neq \hat{b}_a$, то $g(\tilde{\tau}_a) \neq \hat{g}(\tilde{\tau}_a)$ в силу (1), (2) и функции g и $\hat{g}$ можно отличить друг от друга на наборе $\tilde{\tau}_a$. Если же $b_a = \hat{b}_a$, то по аналогии с (1), (2) можно доказать равенства $g(\tilde{\tau}'_a) = b_a$ и $\hat{g}(\tilde{\tau}'_a) = \hat{b}_a$, а значит, и равенство $g(\tilde{\tau}'_a) = \hat{g}(\tilde{\tau}'_a)$ для любого набора $\tilde{\tau}'_a$ длины n, не принадлежащего множеству T. Таким образом, функции $g(\tilde{x}^n)$ и $\hat{g}(\tilde{x}^n)$ могут различаться только на наборах из множества T и обязаны различаться хотя бы на одном таком наборе, поскольку $g \not\equiv \hat{g}$. Получаем, что любые две различные функции g и $\hat{g}$ вида $(*)$ можно отличить друг от друга на наборах из множества $\{\tilde{\tau}_a\} \cup T$, откуда вытекает, что данное множество является ЕДТ длины $|T|+1$ для схемы S и требуемое утверждение доказано.

2. Существует такой набор $\tilde{\tau}_{\overline{a}}$ длины n, не принадлежащий множеству T, на котором функция f принимает значение $\overline{a}$. Тогда эта функция обладает (T,a)-свойством и $(T,\overline{a})$-свойством, поэтому надо доказать неравенство $D^B(f) \leqslant |T|+2$. Имеем

$$g(\tilde{\tau}_{\overline{a}}) = b_1 f(\tilde{\tau}_{\overline{a}}) \oplus b_0\overline{f}(\tilde{\tau}_{\overline{a}}) \oplus I_{T'}(\tilde{\tau}_{\overline{a}}) = b_1\overline{a} \oplus b_0 a \oplus 0 = b_{\overline{a}},$$
$$\hat{g}(\tilde{\tau}_{\overline{a}}) = \hat{b}_1 f(\tilde{\tau}_{\overline{a}}) \oplus \hat{b}_0\overline{f}(\tilde{\tau}_{\overline{a}}) \oplus I_{\hat{T}'}(\tilde{\tau}_{\overline{a}}) = \hat{b}_1\overline{a} \oplus \hat{b}_0 a \oplus 0 = \hat{b}_{\overline{a}}.$$

Если $b_a \neq \hat{b}_a$ или $b_{\overline{a}} \neq \hat{b}_{\overline{a}}$, то $g(\tilde{\tau}_a) \neq \hat{g}(\tilde{\tau}_a)$ или $g(\tilde{\tau}_{\overline{a}}) \neq \hat{g}(\tilde{\tau}_{\overline{a}})$ в силу (1), (2) и выписанных равенств, поэтому функции g и $\hat{g}$ можно отличить друг от друга хотя бы на одном из наборов $\tilde{\tau}_a$, $\tilde{\tau}_{\overline{a}}$. Если же $b_a = \hat{b}_a$ и $b_{\overline{a}} = \hat{b}_{\overline{a}}$, т. е. $b_1 = \hat{b}_1$ и $b_0 = \hat{b}_0$, то из определения функций g, $\hat{g}$ и соотношения $g \not\equiv \hat{g}$ следует, что $T' \neq \hat{T}'$; в таком случае легко видеть, что функции g и $\hat{g}$ можно отличить друг от друга на наборах из множества $T' \triangle \hat{T}' \subseteq T$. Получаем, что любые две различные функции g и $\hat{g}$ вида $(*)$ можно отличить друг от друга на наборах из множества $\{\tilde{\tau}_a, \tilde{\tau}_{\overline{a}}\} \cup T$, откуда вытекает, что данное множество является ЕДТ длины $|T|+2$ для схемы S и требуемое утверждение, а вместе с ним и теорема 1 доказаны. ■

Далее рассмотрим различные приложения теоремы 1.

2. Однотипные константные неисправности на входах и выходах элементов

Докажем вспомогательное утверждение, которое используется в доказательствах теорем 2 и 6. Пусть зафиксированы функционально полный базис B и вид неисправностей функциональных элементов: константные (однотипные типа $p \in \{0,1\}$ либо произвольные) и/или инверсные неисправности на входах и/или выходах элементов.

Лемма 1. Пусть выполнены следующие условия:

а) любую булеву функцию от n переменных, принимающую на наборе $(\tilde{1}^n)$ значение 1, можно реализовать неизбыточной схемой в базисе B, для которой множество $\{(\tilde{1}^n)\}$ является ЕПТ;

б) любую неконстантную булеву функцию от n переменных, принимающую на наборе $(\tilde{1}^n)$ значение 0, можно реализовать схемой в базисе B, неизбыточной и допускающей ЕПТ $\{(\tilde{1}^n)\}$ относительно неисправностей рассматриваемого вида, кроме, быть может, константных неисправностей на выходе выходного элемента схемы.

Тогда для любой неконстантной булевой функции $f(\tilde{x}^n)$ справедливо неравенство $D^B(f) \leqslant 3$.

Доказательство. Введём обозначение $a = f(\tilde{1}^n)$. Положим $m = 4$, $l = 1$,

$$\tilde{c}_a = (c_{a,1}, c_{a,2}, c_{a,3}, c_{a,4}) = (1,0,0,0),\ \tilde{c}_{\overline{a}} = (c_{\overline{a},1}, c_{\overline{a},2}, c_{\overline{a},3}, c_{\overline{a},4}) = (0,0,1,1),$$
$$T = \{(\tilde{1}^n)\},\ \tilde{\pi}_1 = (\tilde{1}^4),\ M_1 = \varnothing,\ M_2 = M_3 = M_4 = T.$$

Пусть $\varphi(\tilde{x}^4)$ — булева функция, удовлетворяющая следующим условиям:

(i) на наборе $\tilde{c}_a$ и всех соседних с ним наборах функция φ принимает значение a;
(ii) на наборе $\tilde{c}_{\overline{a}}$ и всех соседних с ним наборах функция φ принимает значение $\overline{a}$;
(iii) $\varphi(\tilde{\pi}_1) = a$;
(iv) на всех наборах, соседних с набором $\tilde{\pi}_1$, функция φ принимает значение $\overline{a}$.

На всех остальных двоичных наборах длины 4 функция φ может принимать произвольные значения.

Покажем, что данная функция определена корректно, т. е. множества наборов, на которых она принимает значения a и $\overline{a}$, не пересекаются. Заметим, что набор $\tilde{c}_a$ отличается от каждого из наборов $\tilde{c}_{\overline{a}}$, $\tilde{\pi}_1$ в трёх компонентах, поэтому никакой набор, соседний с набором $\tilde{c}_a$, не может быть соседним ни с одним из наборов $\tilde{\pi}_1$, $\tilde{c}_{\overline{a}}$. Набор $\tilde{\pi}_1$ отличается от набора $\tilde{c}_{\overline{a}}$ в двух компонентах, поэтому не является соседним с этим набором. Тем самым показано, что функция $\varphi(\tilde{x}^4)$ определена корректно.

Проверим выполнение условий 1–5 из формулировки теоремы 1. Для любого $i \in \{1,2,3,4\}$ имеем

$$f_i(\tilde{1}^n) = c_{1,i} f(\tilde{1}^n) \oplus c_{0,i} \overline{f}(\tilde{1}^n) \oplus I_{M_i}(\tilde{1}^n) = c_{1,i} a \oplus c_{0,i} \overline{a} \oplus I_{M_i}(\tilde{1}^n) = c_{a,i} \oplus I_{M_i}(\tilde{1}^n). \qquad (3)$$

При $i = 1$ указанное соотношение принимает вид

$$f_1(\tilde{1}^n) = c_{a,1} \oplus I_{M_1}(\tilde{1}^n) = 1 \oplus I_{\varnothing}(\tilde{1}^n) = 1 \oplus 0 = 1,$$

а при $i = 2,3,4$ — вид

$$f_i(\tilde{1}^n) = c_{a,i} \oplus I_{M_i}(\tilde{1}^n) = 0 \oplus I_T(\tilde{1}^n) = 0 \oplus 1 = 1.$$

Тем самым для любого $i \in \{1,2,3,4\}$ доказано равенство $f_i(\tilde{1}^n) = 1$. Тогда по условию *а* функцию $f_i(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B, для которой множество T является ЕПТ, поэтому условие 1 выполнено. Из этого же равенства следует выполнение условия 2, поскольку $\tilde{\pi}_1 = (\tilde{1}^4)$. В силу условий *а*, *б* функцию $\varphi(\tilde{x}^4)$ можно реализовать схемой в базисе B, неизбыточной и допускающей ЕПТ $\{\tilde{\pi}_1\} \subset T_\varphi$ относительно неисправностей рассматриваемого вида, кроме, быть может, константных неисправностей на выходе выходного элемента схемы, поэтому условие 3 также выполнено. Условие 4 следует из (i) и (ii), а условие 5 — из (iii) и (iv). Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^B(f) \leqslant |T| + 2 = 3$. ■

Рассмотрим в качестве базиса множество $B_7 = \{x_1x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3, \overline{x}\}$, а в качестве неисправностей функциональных элементов — однотипные константные неисправности типа 0 на входах и выходах элементов.

Лемма 2. Любую булеву функцию от n переменных, принимающую на наборе $(\tilde{1}^n)$ значение 1, можно реализовать неизбыточной схемой в базисе B_8, допускающей ЕПТ $\{(\tilde{1}^n)\}$.

Лемма 2 следует из лемм 3, 4 работы [16] при $k = 1$.

Лемма 3. Любую булеву функцию от n переменных, принимающую на наборе $(\tilde{1}^n)$ значение 0, можно реализовать неизбыточной схемой в базисе B_8, допускающей ЕПТ $\{(\tilde{1}^n)\}$ относительно неисправностей на входах и выходах элементов, при которых выход выходного элемента схемы исправен.

Лемма 3 следует из леммы 4 работы [16] при $k = 1$.

Теорема 2. Для любого $n \geqslant 0$ справедливо неравенство $D^{B_7}_{0\,(\mathrm{IO})}(n) \leqslant 3$.

Доказательство. Для базиса $B = B_7$ и однотипных константных неисправностей типа 0 на входах и выходах элементов условие *а* леммы 1 выполнено в силу леммы 2, а условие *б* леммы 1 — в силу леммы 3. Из леммы 1 следует, что $D^{B_7}_{0\,(\mathrm{IO})}(f) \leqslant 3$ для любой неконстантной булевой функции $f(\tilde{x}^n)$. В случае $f \equiv 1$ функцию f можно реализовать схемой, состоящей из одного трёхвходового элемента, реализующего функцию вида $x_1x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3$, на все входы которого подаётся переменная x_1; действительно, $x_1x_1x_1 \vee \overline{x}_1\overline{x}_1\overline{x}_1 \equiv 1$. Легко видеть, что при неисправности на выходе этого элемента схема будет реализовывать константу 0, а при неисправности на любом его входе — функцию $\overline{x}_1$, поскольку $0x_1x_1 \vee 1\overline{x}_1\overline{x}_1 = \overline{x}_1$. Любые две из функций $1, 0, \overline{x}_1$ можно отличить друг от друга на множестве $\{(\tilde{1}^n), (\tilde{0}^n)\}$, поэтому $D^{B_7}_{0\,(\mathrm{IO})}(f) \leqslant 2$. Наконец, если $f \equiv 0$, то значение $D^{B_7}_{0\,(\mathrm{IO})}(f)$ не определено в силу леммы 2 из [16] при $k = 1$. Из приведённых рассуждений следует, что $D^{B_7}_{0\,(\mathrm{IO})}(n) \leqslant 3$ для любого $n \geqslant 0$. ■

Используя теорему 2 и принцип двойственности [18, с. 24], а именно, рассматривая схемы, получающиеся заменой всех элементов в схемах из доказательства теоремы 2 на двойственные, нетрудно получить неравенство

$$D^{B_7^*}_{1\,(\mathrm{IO})}(n) \leqslant 3 \tag{4}$$

при $n \geqslant 0$, где $B_7^* = \{\overline{x_1x_2x_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3}, \overline{x}\}$.

Из теоремы 2 и соотношения (4) для любого $n \geqslant 0$ вытекают неравенства $D^{B_7}_{0\,(\mathrm{I})}(n) \leqslant 3$, $D^{B_7^*}_{1\,(\mathrm{I})}(n) \leqslant 3$, $D^{B_7}_{0\,(\mathrm{O})}(n) \leqslant 3$ и $D^{B_7^*}_{1\,(\mathrm{O})}(n) \leqslant 3$, последние два из которых можно отнести и к следующему пункту.

3. Однотипные константные неисправности на выходах элементов

Рассмотрим базис Жегалкина $B_2 = \{\&, \oplus, 1\}$, а в качестве неисправностей функциональных элементов — однотипные константные неисправности типа 1 на выходах элементов. Любой функциональный элемент, реализующий функцию вида $x\&y$ (вида $x \oplus y$, 1, 0), будем называть *конъюнктором* (соответственно *сумматором*, *элементом «константа* 1*»*, *элементом «константа* 0*»*). Вполне разумно считать, что элемент «константа 1» не может быть неисправен.

Двоичный набор $\tilde{\sigma}$ длины n называется *нулевым набором* булевой функции $f(\tilde{x}^n)$, если $f(\tilde{\sigma}) = 0$.

Лемма 4. Любую булеву функцию от n переменных, отличную от константы 1, можно реализовать неизбыточной схемой в базисе B_2, допускающей ЕПТ $\{\tilde{\sigma}\}$, где $\tilde{\sigma}$ — нулевой набор этой функции, содержащий максимальное число единиц.

Доказательство. В случае $f \equiv 0$ реализуем функцию f схемой в базисе B_2, состоящей из одного сумматора, на оба входа которого подаётся константа 1. У такой схемы, очевидно, есть только одна функция неисправности — константа 1, поэтому множество $\{(\tilde{1}^n)\}$ является для неё ЕПТ, что и требовалось доказать. В случае $f \not\equiv 0$ лемма 4 доказывается аналогично основной теореме в [19], где рассматривается базис $\{\&, \oplus, 1, 0\}$, но элемент «константа 0» используется только при построении схемы, реализующей тождественный нуль. ■

Теорема 3. Для любого $n \geqslant 0$ справедливо неравенство $D^{B_2}_{1\,(\mathrm{O})}(n) \leqslant 3$.

Доказательство. Пусть f — произвольная булева функция от n переменных. Докажем неравенство

$$D^{B_2}_{1\,(\mathrm{O})}(f) \leqslant 3, \tag{5}$$

из которого следует справедливость теоремы 3. Если $f \equiv 1$ ($f \equiv 0$), то функцию f можно реализовать схемой в базисе B_2, состоящей из одного элемента «константа 1» (соответственно, состоящей из одного сумматора, на оба входа которого подаётся константа 1). У такой схемы нет ни одной функции неисправности (соответственно, есть только одна функция неисправности — константа 1), поэтому множество, состоящее из любого одного набора длины n, является для неё ЕДТ длины 1 и неравенство (5) выполнено.

Далее будем считать, что функция f отлична от констант. Тогда существует хотя бы один её нулевой набор. Рассмотрим два случая:

1. Единственным нулевым набором функции $f(\tilde{x}^n)$ является набор $(\tilde{1}^n)$. Положим $m = 2$, $l = 1$,

$$\tilde{c}_1 = (c_{1,1}, c_{1,2}) = (1,1),\ \tilde{c}_0 = (c_{0,1}, c_{0,2}) = (0,0),$$
$$T = \{(\tilde{1}^n)\},\ \tilde{\pi}_1 = (0,0),\ M_1 = M_2 = \varnothing,\ \varphi(x_1, x_2) = x_1 \vee x_2.$$

Проверим выполнение условий 1–5 теоремы 1. Для любого $i \in \{1, 2\}$ имеем

$$f_i(\tilde{1}^n) = c_{1,i} f(\tilde{1}^n) \oplus c_{0,i}\overline{f}(\tilde{1}^n) \oplus I_{M_i}(\tilde{1}^n) = 1 \cdot 0 \oplus 0 \cdot 1 \oplus I_{\varnothing}(\tilde{1}^n) = 0.$$

Из равенств $f_i(\tilde{1}^n) = 0$, где $i = 1, 2$, и леммы 4 следует, что функцию $f_i(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B_2, для которой множество T является ЕПТ, и условие 1 выполнено. Из этих же равенств следует выполнение условия 2, поскольку $\tilde{\pi}_1 = (0,0)$. Набор $\tilde{\pi}_1$ является нулевым набором функции $\varphi(x_1, x_2) = x_1 \vee x_2$, содержащим максимальное число единиц, поэтому в силу леммы 4 данную функцию

можно реализовать неизбыточной схемой в базисе B_2, допускающей ЕПТ $\{\tilde{\pi}_1\} \subset T_\varphi$, и условие 3 также выполнено. Условия 4 и 5 следуют из свойств функции $x_1 \vee x_2$ и того факта, что функция f не обладает $(T,0)$-свойством. Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^{B_2}_{1\,(\mathrm{O})}(f) \leqslant |T| + 1 < 3$.

2. Существует нулевой набор функции $f(\tilde{x}^n)$, отличный от набора $(\tilde{1}^n)$. Введём обозначение $a = f(\tilde{1}^n)$. Положим $m = 6$, $l = 2$,

$$\tilde{c}_1 = (c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{1,5}, c_{1,6}) = (1,1,1,0,0,0),\ \tilde{c}_0 = (c_{0,1}, c_{0,2}, c_{0,3}, c_{0,4}, c_{0,5}, c_{0,6}) = (\tilde{1}^6),$$

$$T = \{(\tilde{1}^n)\},\ \tilde{\pi}_1 = (\tilde{0}^6),\ M_1 = M_2 = M_3 = T,\ M_4 = M_5 = M_6 = \begin{cases} T, & \text{если } a = 0, \\ \varnothing, & \text{если } a = 1. \end{cases}$$

Пусть $\varphi(\tilde{x}^6)$ — произвольная булева функция, удовлетворяющая условиям (i)–(iv) из доказательства леммы 1. Любые два из наборов $\tilde{c}_1, \tilde{c}_0, \tilde{\pi}_1$ отличаются друг от друга хотя бы в трёх компонентах, поэтому данная функция определена корректно.

Проверим выполнение условий 1–5 теоремы 1. Для любого $i \in \{1,2,3,4,5,6\}$ выполнено соотношение (3). При $i = 1,2,3$ оно принимает вид

$$f_i(\tilde{1}^n) = c_{a,i} \oplus I_T(\tilde{1}^n) = 1 \oplus 1 = 0;$$

при $a = 0$, $i = 4,5,6$ — вид

$$f_i(\tilde{1}^n) = c_{0,i} \oplus I_T(\tilde{1}^n) = 1 \oplus 1 = 0;$$

при $a = 1$, $i = 4,5,6$ — вид

$$f_i(\tilde{1}^n) = c_{1,i} \oplus I_\varnothing(\tilde{1}^n) = 0 \oplus 0 = 0.$$

Тем самым для любых $a \in \{0,1\}$, $i \in \{1,2,3,4,5,6\}$ доказано равенство $f_i(\tilde{1}^n) = 0$. Тогда по лемме 4 функцию $f_i(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B_2, для которой множество T является ЕПТ, и условие 1 выполнено. Из этого же равенства следует выполнение условия 2, поскольку $\tilde{\pi}_1 = (\tilde{0}^6)$. Набор $\tilde{c}_0$ в силу условия (i) является нулевым набором функции $\varphi(\tilde{x}_6)$, содержащим максимальное число единиц, поэтому по лемме 4 данную функцию можно реализовать неизбыточной схемой в базисе B_2, допускающей ЕПТ $\{\tilde{c}_0\} \subset T_\varphi$, и условие 3 также выполнено (включение $\{\tilde{c}_0\} \subset T_\varphi$ верно в силу определения множества T_φ в формулировке теоремы 1 и того факта, что функция f обладает $(T,0)$-свойством по предположению случая 2). Условие 4 следует из (i) и (ii), а условие 5 — из (iii) и (iv). Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^{B_2}_{1\,(\mathrm{O})}(f) \leqslant |T| + 2 = 3$. ■

Используя теорему 3 и принцип двойственности, нетрудно получить неравенство

$$D^{B_2^*}_{0\,(\mathrm{O})}(n) \leqslant 3 \tag{6}$$

при $n \geqslant 0$, где $B_2^* = \{\vee, \sim, 0\}$.

4. Произвольные константные неисправности на входах и выходах элементов

Рассмотрим в качестве неисправностей функциональных элементов произвольные константные неисправности на входах и выходах элементов, а в качестве базиса — множество $B_8 = \{\eta(\tilde{x}^4), x_1 \sim x_2, \overline{x}, 0\}$, где $\eta(\tilde{x}^4)$ — произвольная несамодвойственная

булева функция, принимающая значение α на наборе $(\tilde{\alpha}^4)$ и значение $\overline{\alpha}$ на всех соседних с ним наборах для любого $\alpha \in \{0,1\}$ (определение самодвойственной булевой функции можно найти, например, в [18, с. 23, с. 34]).

Два двоичных набора одинаковой длины называются *противоположными*, если они различаются во всех компонентах.

Лемма 5. Любую булеву функцию от n переменных, принимающую разные значения на противоположных наборах $\tilde{\sigma}_1$ и $\tilde{\sigma}_2$, можно реализовать неизбыточной схемой в базисе B_8, допускающей ЕПТ $\{\tilde{\sigma}_1, \tilde{\sigma}_2\}$.

Лемма 5 следует из леммы 4 работы [17] при $k = 1$ и $\varphi(\tilde{x}^{2k+2}) = \eta(\tilde{x}_4)$.

Лемма 6. Любую булеву функцию от n переменных, принимающую значение 1 на противоположных наборах $\tilde{\sigma}_1$ и $\tilde{\sigma}_2$, можно реализовать схемой в базисе B_8, неизбыточной и допускающей ЕПТ $\{\tilde{\sigma}_1, \tilde{\sigma}_2\}$ относительно неисправностей на входах и выходах элементов, кроме неисправности типа 1 на выходе выходного элемента схемы.

Лемма 6 доказывается по аналогии с доказательством леммы 5 работы [17] при $k = 1$.

Теорема 4. Для любого $n \geqslant 0$ справедливо неравенство $D^{B_8}_{0,1\,(\mathrm{IO})}(n) \leqslant 4$.

Доказательство. Пусть f — произвольная булева функция от n переменных. Если $f \equiv 1$, то значение $D^{B_8}_{0,1\,(\mathrm{IO})}(f)$ не определено в силу леммы 6 из [17] при $k = 1$. Для любой другой функции f докажем неравенство

$$D^{B_8}_{0,1\,(\mathrm{IO})}(f) \leqslant 4, \tag{7}$$

из которого следует справедливость теоремы 4. В случае $f \equiv 0$ функцию f можно реализовать схемой, состоящей из одного элемента «константа 0». Он не имеет входов, поэтому единственной возможной неисправностью данной схемы является неисправность типа 1 на его выходе, при которой схема реализует константу 1. Указанная неисправность обнаруживается на любом двоичном наборе длины n, поэтому $D^{B_8}_{0,1\,(\mathrm{IO})}(f) \leqslant 1$ и неравенство (7) выполнено.

Далее будем считать, что функция f отлична от констант. Тогда существует такой набор $\tilde{\sigma}_1$ длины n, что $f(\tilde{\sigma}_1) = 1$. Пусть $\tilde{\sigma}_2$ — набор, противоположный набору $\tilde{\sigma}_1$. Введём обозначение $a = f(\tilde{\sigma}_2)$. Положим $m = 6$, $l = 2$,

$$\tilde{c}_1 = (c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{1,5}, c_{1,6}) = (1,1,1,0,0,0),$$
$$\tilde{c}_0 = (c_{0,1}, c_{0,2}, c_{0,3}, c_{0,4}, c_{0,5}, c_{0,6}) = (0,0,0,1,1,1),$$
$$T = \{\tilde{\sigma}_1, \tilde{\sigma}_2\},\ \tilde{\pi}_1 = (\tilde{1}^6),\ \tilde{\pi}_2 = (\tilde{0}^6),$$
$$M_1 = M_2 = M_3 = \begin{cases} \varnothing, & \text{если } a = 0, \\ \{\tilde{\sigma}_2\}, & \text{если } a = 1, \end{cases}$$
$$M_4 = M_5 = M_6 = \begin{cases} T, & \text{если } a = 0, \\ \{\tilde{\sigma}_1\}, & \text{если } a = 1. \end{cases}$$

Пусть $\varphi(\tilde{x}^6)$ — булева функция, удовлетворяющая следующим условиям:

(iii′) $\varphi(\tilde{\pi}_1) = 1$;

(iv′) на всех наборах, соседних с набором $\tilde{\pi}_1$, функция φ принимает значение 0;

(v) $\varphi(\tilde{\pi}_2) = a$;

(vi) на всех наборах, соседних с набором $\tilde{\pi}_2$, функция φ принимает значение $\overline{a}$,

а также условиям (i), (ii). На всех остальных двоичных наборах длины 6 функция φ может принимать произвольные значения.

Любые два из наборов $\tilde{c}_1, \tilde{c}_0, \tilde{\pi}_1, \tilde{\pi}_2$ отличаются друг от друга хотя бы в трёх компонентах, поэтому функция определена корректно.

Проверим выполнение условий 1–5 теоремы 1. Для любого $i \in \{1,2,3,4,5,6\}$ имеем

$$f_i(\tilde{\sigma}_1) = c_{1,i} f(\tilde{\sigma}_1) \oplus c_{0,i}\overline{f}(\tilde{\sigma}_1) \oplus I_{M_i}(\tilde{\sigma}_1) = c_{1,i} \oplus I_{M_i}(\tilde{\sigma}_1),$$
$$f_i(\tilde{\sigma}_2) = c_{1,i} f(\tilde{\sigma}_2) \oplus c_{0,i}\overline{f}(\tilde{\sigma}_2) \oplus I_{M_i}(\tilde{\sigma}_2) = c_{1,i} a \oplus c_{0,i}\overline{a} \oplus I_{M_i}(\tilde{\sigma}_2) = c_{a,i} \oplus I_{M_i}(\tilde{\sigma}_2).$$

При $a = 0$, $i = 1,2,3$ указанные соотношения принимают вид

$$f_i(\tilde{\sigma}_1) = c_{1,i} \oplus I_{\varnothing}(\tilde{\sigma}_1) = 1 \oplus 0 = 1,\ \ f_i(\tilde{\sigma}_2) = c_{0,i} \oplus I_{\varnothing}(\tilde{\sigma}_2) = 0 \oplus 0 = 0;$$

при $a = 0$, $i = 4,5,6$ — вид

$$f_i(\tilde{\sigma}_1) = c_{1,i} \oplus I_T(\tilde{\sigma}_1) = 0 \oplus 1 = 1,\ \ f_i(\tilde{\sigma}_2) = c_{0,i} \oplus I_T(\tilde{\sigma}_2) = 1 \oplus 1 = 0;$$

при $a = 1$, $i = 1,2,3$ — вид

$$f_i(\tilde{\sigma}_1) = c_{1,i} \oplus I_{\{\tilde{\sigma}_2\}}(\tilde{\sigma}_1) = 1 \oplus 0 = 1,\ \ f_i(\tilde{\sigma}_2) = c_{1,i} \oplus I_{\{\tilde{\sigma}_2\}}(\tilde{\sigma}_2) = 1 \oplus 1 = 0;$$

при $a = 1$, $i = 4,5,6$ — вид

$$f_i(\tilde{\sigma}_1) = c_{1,i} \oplus I_{\{\tilde{\sigma}_1\}}(\tilde{\sigma}_1) = 0 \oplus 1 = 1,\ \ f_i(\tilde{\sigma}_2) = c_{1,i} \oplus I_{\{\tilde{\sigma}_1\}}(\tilde{\sigma}_2) = 0 \oplus 0 = 0.$$

Тем самым для любых $a \in \{0,1\}$, $i \in \{1,2,3,4,5,6\}$ доказаны равенства $f_i(\tilde{\sigma}_1) = 1$, $f_i(\tilde{\sigma}_2) = 0$. Тогда по лемме 5 функцию $f_i(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B_8, для которой множество T является ЕПТ, поэтому условие 1 выполнено. Из этих же равенств следует выполнение условия 2, поскольку $\tilde{\pi}_1 = (\tilde{1}^6)$, $\tilde{\pi}_2 = (\tilde{0}^6)$.

В случае $a = 0$ в силу условий (iii′), (v) и леммы 5 функцию $\varphi(\tilde{x}^6)$ можно реализовать неизбыточной схемой в базисе B_8, допускающей ЕПТ $\{\tilde{\pi}_1, \tilde{\pi}_2\} \subset T_\varphi$. Если $a = 1$, то из условий (iii′) и (v) получаем, что $\varphi(\tilde{\pi}_1) = \varphi(\tilde{\pi}_2) = 1$; тогда в силу леммы 6 функцию $\varphi(\tilde{x}^6)$ можно реализовать схемой в базисе B_8, неизбыточной и допускающей ЕПТ $\{\tilde{\pi}_1, \tilde{\pi}_2\} \subset T_\varphi$ относительно неисправностей на входах и выходах элементов, кроме неисправности типа 1 на выходе её выходного элемента. В каждом из случаев $a = 0$, $a = 1$ получаем, что условие 3 выполнено.

Условие 4 следует из (i) и (ii), а условие 5 — из (iii′), (iv′), (v) и (vi). Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^{B_8}_{0,1\,(\mathrm{IO})}(f) \leqslant |T| + 2 = 4$. ■

Из теоремы 4 для любого $n \geqslant 0$ следуют неравенства $D^{B_8}_{0,1\,(\mathrm{I})}(n) \leqslant 4$ и $D^{B_8}_{0,1\,(\mathrm{O})}(n) \leqslant 4$, второе из которых можно отнести и к следующему пункту.

5. Произвольные константные неисправности на выходах элементов

Рассмотрим в качестве базиса множество $B_9 = \{x\&y, \overline{x}, x \oplus y \oplus z\}$, а в качестве неисправностей функциональных элементов — произвольные константные неисправности на выходах элементов.

Лемма 7. Любую булеву функцию от n переменных, принимающую разные значения на наборах $(\tilde{1}^n)$ и $(\tilde{0}^n)$, можно реализовать неизбыточной схемой в базисе B_9, допускающей ЕПТ $\{(\tilde{1}^n), (\tilde{0}^n)\}$.

Лемма 7 следует из рассмотрения случая А в доказательстве теоремы 1 из [10].

Лемма 8. Любую неконстантную булеву функцию от n переменных, принимающую одинаковые значения на наборах $(\tilde{1}^n)$ и $(\tilde{0}^n)$, самая короткая конъюнкция в полиноме Жегалкина которой равна $x_1\& \ldots \& x_k$, можно реализовать неизбыточной схемой в базисе B_9, допускающей ЕПТ $\{(\tilde{1}^n), (\tilde{1}^k, \tilde{0}^{n-k})\}$.

Лемма 8 следует из рассмотрения случая Б в доказательстве теоремы 1 из [10].

Теорема 5. Для любого $n \geqslant 1$ справедливо неравенство $D^{B_9}_{0,1\,(\mathrm{O})}(n) \leqslant 4$.

Доказательство. Пусть f — произвольная булева функция от n переменных. Если $f \equiv 0$ ($f \equiv 1$), то в любой схеме, реализующей функцию f, должен содержаться выходной элемент; тогда при неисправности типа 0 (типа 1) этого элемента схема по-прежнему реализует константу 0 (соответственно 1), т. е. исходная схема избыточна. Получаем, что неизбыточных схем, реализующих функцию f, не существует и значение $D^{B_9}_{0,1\,(\mathrm{O})}(f)$ не определено. Для любой другой (т. е. отличной от констант) функции f докажем неравенство

$$D^{B_9}_{0,1\,(\mathrm{O})}(f) \leqslant 4,$$

из которого следует справедливость теоремы 5. Введём обозначение $a = f(\tilde{1}^n)$. Рассмотрим два случая:

1. Пусть $f(\tilde{0}^n) = \overline{a}$. Положим $m = 6$, $l = 2$,

$$\tilde{c}_1 = (c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{1,5}, c_{1,6}) = (1,1,1,0,0,0),$$
$$\tilde{c}_0 = (c_{0,1}, c_{0,2}, c_{0,3}, c_{0,4}, c_{0,5}, c_{0,6}) = (0,0,0,1,1,1),$$
$$T = \{(\tilde{1}^n), (\tilde{0}^n)\},\ \tilde{\pi}_1 = (\tilde{1}^6),\ \tilde{\pi}_2 = (\tilde{0}^6),$$
$$M_1 = M_2 = M_3 = \begin{cases} T, & \text{если } a = 0, \\ \varnothing, & \text{если } a = 1, \end{cases}$$
$$M_4 = M_5 = M_6 = \begin{cases} \varnothing, & \text{если } a = 0, \\ T, & \text{если } a = 1. \end{cases}$$

Пусть $\varphi(\tilde{x}^6)$ — булева функция, удовлетворяющая следующим условиям:
(v′) $\varphi(\tilde{\pi}_2) = \overline{a}$;
(vi′) на всех наборах, соседних с набором $\tilde{\pi}_2$, функция φ принимает значение a,
а также условиям (i)–(iv). На всех остальных двоичных наборах длины 6 функция φ может принимать произвольные значения.

Проверим выполнение условий 1–5 теоремы 1. Для любого $i \in \{1,2,3,4,5,6\}$ выполнены соотношения (3) и

$$f_i(\tilde{0}^n) = c_{1,i} f(\tilde{0}^n) \oplus c_{0,i} \overline{f}(\tilde{0}^n) \oplus I_{M_i}(\tilde{0}^n) = c_{1,i}\overline{a} \oplus c_{0,i} a \oplus I_{M_i}(\tilde{0}^n) = c_{\overline{a},i} \oplus I_{M_i}(\tilde{0}^n).$$

При $a = 0$, $i = 1,2,3$ они принимают вид

$$f_i(\tilde{1}^n) = c_{0,i} \oplus I_T(\tilde{1}^n) = 0 \oplus 1 = 1,\ f_i(\tilde{0}^n) = c_{1,i} \oplus I_T(\tilde{0}^n) = 1 \oplus 1 = 0;$$

при $a = 0$, $i = 4,5,6$ — вид

$$f_i(\tilde{1}^n) = c_{0,i} \oplus I_\varnothing(\tilde{1}^n) = 1 \oplus 0 = 1,\ f_i(\tilde{0}^n) = c_{1,i} \oplus I_\varnothing(\tilde{0}^n) = 0 \oplus 0 = 0;$$

при $a = 1$, $i = 1,2,3$ — вид

$$f_i(\tilde{1}^n) = c_{1,i} \oplus I_\varnothing(\tilde{1}^n) = 1 \oplus 0 = 1,\ f_i(\tilde{0}^n) = c_{0,i} \oplus I_\varnothing(\tilde{0}^n) = 0 \oplus 0 = 0;$$

при $a = 1$, $i = 4,5,6$ — вид

$$f_i(\tilde{1}^n) = c_{1,i} \oplus I_T(\tilde{1}^n) = 0 \oplus 1 = 1,\ f_i(\tilde{0}^n) = c_{0,i} \oplus I_T(\tilde{0}^n) = 1 \oplus 1 = 0.$$

Тем самым для любых $a \in \{0,1\}$, $i \in \{1,2,3,4,5,6\}$ доказаны равенства $f_i(\tilde{1}^n) = 1$, $f_i(\tilde{0}^n) = 0$, а значит, и неравенство $f_i(\tilde{1}^n) \neq f_i(\tilde{0}^n)$. Тогда по лемме 7 функцию $f_i(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B_9, для которой множество T является ЕПТ, поэтому условие 1 выполнено. Из этих же равенств следует выполнение условия 2, поскольку $\tilde{\pi}_1 = (\tilde{1}^6)$, $\tilde{\pi}_2 = (\tilde{0}^6)$.

В силу условий (iii), (v′) и леммы 7 функцию $\varphi(\tilde{x}^6)$ можно реализовать неизбыточной схемой в базисе B_9, допускающей ЕПТ $\{\tilde{\pi}_1, \tilde{\pi}_2\} \subset T_\varphi$. Получаем, что условие 3 также выполнено. Условие 4 следует из (i) и (ii), а условие 5 — из (iii), (iv), (v′) и (vi′). Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^{B_9}_{0,1\,(\mathrm{O})}(f) \leqslant |T| + 2 = 4$.

2. Пусть $f(\tilde{0}^n) = a$. Положим $m = 6$, $l = 2$,

$$\tilde{c}_a = (c_{a,1}, c_{a,2}, c_{a,3}, c_{a,4}, c_{a,5}, c_{a,6}) = (\tilde{0}^6),\ \tilde{c}_{\overline{a}} = (c_{\overline{a},1}, c_{\overline{a},2}, c_{\overline{a},3}, c_{\overline{a},4}, c_{\overline{a},5}, c_{\overline{a},6}) = (\tilde{1}^6),$$
$$T = \{(\tilde{1}^n), (\tilde{0}^n)\},\ \tilde{\pi}_1 = (1,1,1,0,0,0),\ \tilde{\pi}_2 = (0,0,0,1,1,1),$$
$$M_1 = M_2 = M_3 = \{(\tilde{1}^n)\},\ M_4 = M_5 = M_6 = \{(\tilde{0}^n)\}.$$

Булеву функцию $\varphi(\tilde{x}^6)$ определим позже.

Проверим выполнение условий 1 и 2 из теоремы 1. Для любого $i \in \{1,2,3,4,5,6\}$ имеем

$$f_i(\tilde{1}^n) = c_{1,i} f(\tilde{1}^n) \oplus c_{0,i} \overline{f}(\tilde{1}^n) \oplus I_{M_i}(\tilde{1}^n) = c_{1,i} a \oplus c_{0,i} \overline{a} \oplus I_{M_i}(\tilde{1}^n) = c_{a,i} \oplus I_{M_i}(\tilde{1}^n) = I_{M_i}(\tilde{1}^n),$$
$$f_i(\tilde{0}^n) = c_{1,i} f(\tilde{0}^n) \oplus c_{0,i} \overline{f}(\tilde{0}^n) \oplus I_{M_i}(\tilde{0}^n) = c_{1,i} a \oplus c_{0,i} \overline{a} \oplus I_{M_i}(\tilde{0}^n) = c_{a,i} \oplus I_{M_i}(\tilde{0}^n) = I_{M_i}(\tilde{0}^n).$$

При $i = 1,2,3$ указанные соотношения принимают вид

$$f_i(\tilde{1}^n) = I_{\{(\tilde{1}^n)\}}(\tilde{1}^n) = 1,\ f_i(\tilde{0}^n) = I_{\{(\tilde{1}^n)\}}(\tilde{0}^n) = 0,$$

а при $i = 4,5,6$ — вид

$$f_i(\tilde{1}^n) = I_{\{(\tilde{0}^n)\}}(\tilde{1}^n) = 0,\ f_i(\tilde{0}^n) = I_{\{(\tilde{0}^n)\}}(\tilde{0}^n) = 1.$$

Из последних равенств следует выполнение условия 2, поскольку $\tilde{\pi}_1 = (1,1,1,0,0,0)$, $\tilde{\pi}_2 = (0,0,0,1,1,1)$, и, кроме того, справедливость неравенства $f_i(\tilde{1}^n) \neq f_i(\tilde{0}^n)$ для любого $i \in \{1,2,3,4,5,6\}$. Тогда по лемме 7 функцию $f_i(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B_9, для которой множество T является ЕПТ, поэтому условие 1 также выполнено. Рассмотрим два подслучая:

2.1. Пусть функция $f(\tilde{x}^n)$ на наборах, не принадлежащих множеству T, принимает оба значения a, $\overline{a}$. Возьмём в качестве $\varphi(\tilde{x}^6)$ произвольную булеву функцию, удовлетворяющую условиям (i)–(vi).

Проверим выполнение условий 3–5 теоремы 1. В силу условий (i), (ii) и леммы 7 функцию $\varphi(\tilde{x}^6)$ можно реализовать неизбыточной схемой в базисе B_9, допускающей ЕПТ $\{\tilde{c}_a, \tilde{c}_{\overline{a}}\} \subset T_\varphi$, поэтому условие 3 выполнено (включение $\{\tilde{c}_a, \tilde{c}_{\overline{a}}\} \subset T_\varphi$ верно в силу определения множества T_φ в формулировке теоремы 1 и того факта, что функция f обладает (T,a)-свойством и $(T,\overline{a})$-свойством по предположению подслучая 2.1). Условие 4 следует из (i) и (ii), а условие 5 — из условий (iii)–(vi). Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^{B_9}_{0,1\,(\mathrm{O})}(f) \leqslant |T| + 2 = 4$.

2.2. Пусть функция $f(\tilde{x}^n)$ на любом наборе, не принадлежащем множеству T, принимает одно и то же значение. Это значение равно $\overline{a}$, поскольку $f \not\equiv a$, но

$f(\tilde{1}^n) = f(\tilde{0}^n) = a$. Возьмём в качестве $\varphi(\tilde{x}^6)$ булеву функцию, принимающая значение a на наборах $\tilde{\pi}_1$ и $\tilde{\pi}_2$ и значение $\overline{a}$ на всех остальных наборах. Представим её полиномом Жегалкина

$$\varphi(\tilde{x}^6) = K_1 \oplus \ldots \oplus K_m \oplus c, \tag{8}$$

где $m \geqslant 1$; $c \in \{0,1\}$; K_1 — самая короткая конъюнкция в этом полиноме.

Проверим выполнение условий 3–5 теоремы 1. Пусть $K_1 = x_{j_1}\& \ldots \& x_{j_k}$. Ясно, что на наборе $(\tilde{0}^6)$ функция φ принимает значение $\underbrace{0 \oplus \ldots \oplus 0}_{m} \oplus c = c$, которое по определению этой функции равно $\overline{a}$, а на наборе $\tilde{\pi}$ длины 6, единичными компонентами которого являются в точности компоненты с номерами $j_1, \ldots, j_k$, она принимает значение $1 \oplus \underbrace{0 \oplus \ldots \oplus 0}_{m-1} \oplus c = 1 \oplus c$, которое равно a. Это означает, что $\tilde{\pi} = \tilde{\pi}_1$ или $\tilde{\pi} = \tilde{\pi}_2$, тогда $k = 3$, т. е. ранг самой короткой конъюнкции K_1 в правой части представления (8) равен 3. В эту часть обязательно входит слагаемое $x_1x_2x_3$, поскольку в противном случае на наборе $\tilde{\pi}_1$ все конъюнкции $K_1, \ldots, K_m$ обращаются в нуль и значение $\varphi(\tilde{\pi}_1)$ равно $c = \overline{a}$, что противоречит определению функции φ. Поэтому можно считать, что $K_1 = x_1x_2x_3$. Тогда по лемме 8 функцию $\varphi(\tilde{x}^6)$ можно реализовать неизбыточной схемой в базисе B_9, допускающей ЕПТ $\{(\tilde{1}^6), (1,1,1,0,0,0)\} = \{\tilde{c}_{\overline{a}}, \tilde{\pi}_1\} \subset T_\varphi$, и условие 3 выполнено (включение $\{\tilde{c}_{\overline{a}}, \tilde{\pi}_1\} \subset T_\varphi$ верно в силу определения множества T_φ в формулировке теоремы 1 и того факта, что функция f обладает $(T, \overline{a})$-свойством по предположению подслучая 2.2).

Условия 4 и 5 проверяются непосредственно с учётом определения функции φ и того, что функция f не обладает (T, a)-свойством по предположению подслучая. Таким образом, выполнены все условия теоремы 1, из которой следует, что $D^{B_9}_{0,1\,(\mathrm{O})}(f) \leqslant$ $\leqslant |T| + 1 < 4$. ■

6. Инверсные неисправности

Пусть зафиксированы «места» возможных неисправностей функциональных элементов: входы и выходы элементов, только их входы или только их выходы.

Утверждение 1. Пусть S — схема из функциональных элементов, неизбыточная и допускающая ЕПТ T относительно однотипных константных неисправностей типа p, $p \in \{0,1\}$, элементов, кроме, быть может, неисправности на выходе её выходного элемента. Тогда она неизбыточна и допускает ЕПТ T относительно инверсных неисправностей элементов.

Доказательство. Если в схеме S невозможна константная неисправность типа p ни одного элемента (такое может быть, например, в случае, когда в этой схеме не содержится элементов), то в ней невозможна и инверсная неисправность ни одного элемента и утверждение очевидно. Пусть теперь в схеме S возможна константная неисправность типа p хотя бы одного элемента. Данная неисправность в силу неизбыточности схемы должна быть обнаружена хотя бы на одном наборе из множества T, поэтому $|T| \geqslant 1$. Предположим, что имеет место инверсная неисправность произвольного элемента схемы S. Рассмотрим два случая:

1. Неисправен выход выходного элемента схемы S. Тогда схема реализует отрицание исходной функции и указанную неисправность можно обнаружить на любом наборе из множества T.

2. Имеет место любая другая инверсная неисправность одного элемента схемы S. Неисправный вход/выход элемента обозначим через v. Множество T по условию позволяет обнаружить константную неисправность типа p на входе/выходе v, поэтому

в T найдётся такой набор $\tilde{\sigma}$, что значение a, выдаваемое схемой S на этом наборе при отсутствии в ней неисправностей, отличается от значения на выходе этой схемы на наборе $\tilde{\sigma}$ при неисправности типа p входа/выхода v (которое соответственно равно $\overline{a}$). При этом значение на входе/выходе v в схеме S при его неисправности типа p на наборе $\tilde{\sigma}$ равно p, а при отсутствии неисправностей в схеме S обязано равняться $\overline{p}$ (иначе указанную неисправность нельзя обнаружить на наборе $\tilde{\sigma}$). Тогда при возникновении инверсной неисправности на входе/выходе v значение на нём в схеме S на наборе $\tilde{\sigma}$ меняется с $\overline{p}$ на p, а значение на выходе схемы S — с a на $\overline{a}$. Следовательно, рассматриваемую инверсную неисправность можно обнаружить хотя бы на одном наборе из множества T, откуда вытекает справедливость утверждения 1. ■

Следствие 1. Пусть S — схема из функциональных элементов, неизбыточная и допускающая ЕПТ T относительно произвольных константных неисправностей элементов, кроме, быть может, неисправностей на выходе выходного элемента. Тогда она неизбыточна и допускает ЕПТ T относительно инверсных неисправностей элементов.

С использованием утверждения 1 и следствия 1 из теорем 2–5 и неравенств (4) и (6) можно получить аналогичные оценки величин $D^B(n)$ в случае, когда, помимо константных неисправностей на входах и выходах (только на выходах) элементов, допускаются также инверсные неисправности на входах и выходах (соответственно только на выходах) элементов. Действительно, если выполнены условия 1–5 теоремы 1 при однотипных или произвольных константных неисправностях элементов, то все эти условия справедливы и при дополнительном рассмотрении инверсных неисправностей элементов в тех же местах.

7. Инверсные неисправности на входах и выходах элементов

Рассмотрим базис Жегалкина B_2, а в качестве неисправностей функциональных элементов — инверсные неисправности на входах и выходах элементов.

Утверждение 2. Любую булеву функцию $f(\tilde{x}^n)$ можно реализовать неизбыточной схемой в базисе B_2, для которой множество $\{(\tilde{1}^n)\}$ является ЕПТ.

Доказательство. Если $f \equiv 1$ ($f \equiv 0$), то функцию f можно реализовать схемой в базисе B_2, состоящей из одного элемента «константа 1» (соответственно, состоящей из одного сумматора, на оба входа которого подаётся константа 1). У такой схемы нет ни одной функции неисправности (соответственно, есть только одна функция неисправности — константа 1) и утверждение очевидно. Далее считаем, что функция f отлична от констант. Реализуем её стандартной схемой S в базисе B_2, моделирующей представление этой функции полиномом Жегалкина (см., например, [4, с. 114]; слагаемое c либо отсутствует, либо равно 1). Рассмотрим неисправность произвольного одного входа/выхода элемента схемы S. Если неисправен вход или выход какого-то конъюнктора, то на наборе $(\tilde{1}^n)$ значение на выходе цепочки из конъюнкторов, содержащей данный конъюнктор, изменится с 1 на 0. Это изменение, а также изменение на входе или выходе какого-то сумматора либо на выходе элемента «константа 1», в случае, если указанный вход/выход неисправен, очевидно, пройдёт по цепочке из сумматоров до выхода схемы, и неисправность будет обнаружена на наборе $(\tilde{1}^n)$. ■

Теорема 6. Для любого $n \geqslant 0$ справедливо неравенство $D^{B_2}_{\mathrm{Inv\,(IO)}}(n) \leqslant 3$.

Доказательство. Для базиса $B = B_2$ и инверсных неисправностей на входах и выходах элементов условия *а* и *б* леммы 1 выполнены в силу утверждения 2. Из этой леммы следует, что $D^{B_2}_{\mathrm{Inv\,(IO)}}(f) \leqslant 3$ для любой неконстантной булевой функции $f(\tilde{x}^n)$.

В случаях $f \equiv 1$, $f \equiv 0$ очевидным образом получаем, что $D^{B_2}_{\mathrm{Inv\,(IO)}}(f) \leqslant 1$ — см. начало доказательства утверждения 2. Из приведённых неравенств следует справедливость теоремы 6. ■

Используя теорему 6 и принцип двойственности, нетрудно получить неравенство

$$D^{B_2^*}_{\mathrm{Inv\,(IO)}}(n) \leqslant 3 \tag{9}$$

при $n \geqslant 0$ для базиса $B_2^* = \{\vee, \sim, 0\}$.

Из теоремы 6 и соотношения (9) для любого $n \geqslant 0$ вытекают неравенства $D^{B_2}_{\mathrm{Inv\,(I)}}(n) \leqslant 3$ и $D^{B_2^*}_{\mathrm{Inv\,(I)}}(n) \leqslant 3$.

Заключение

Предложен метод построения схем из функциональных элементов в произвольном функционально полном базисе, допускающих короткие единичные диагностические тесты относительно константных (однотипных или произвольных) либо инверсных неисправностей на входах и/или выходах элементов, основанный на существовании схем в том же базисе, допускающих короткие единичные проверяющие тесты относительно неисправностей такого же типа. С использованием данного метода получен ряд новых константных верхних оценок функций Шеннона длины ЕДТ для схем в различных базисах при разных типах неисправностей элементов.

Метод может иметь большую теоретическую значимость: с его использованием из любых существующих и новых верхних оценок функций Шеннона длины ЕПТ для схем из функциональных элементов можно пытаться получить отличающиеся от них не более чем на 2 верхние оценки функций Шеннона длины ЕДТ для схем в том же базисе при тех же неисправностях. Способы получения таких оценок наглядно продемонстрированы в доказательствах теорем 2–6. Вместе с тем указанный метод, вероятно, не позволяет получить точные значения функций Шеннона длины ЕДТ для схем из функциональных элементов (т. е. верхние оценки, полученные с его помощью, не являются окончательными), в отличие от других методов синтеза схем, использованных, например, в работах автора [6, 7].

ЛИТЕРАТУРА

1. *Чегис И. А., Яблонский С. В.* Логические способы контроля работы электрических схем // Труды МИАН. 1958. Т. 51. С. 270–360.
2. *Яблонский С. В.* Надёжность и контроль управляющих систем // Материалы Всесоюзного семинара по дискретной математике и её приложениям (Москва, 31 января–2 февраля 1984 г.). М.: Изд-во МГУ, 1986. С. 7–12.
3. *Яблонский С. В.* Некоторые вопросы надёжности и контроля управляющих систем // Математические вопросы кибернетики. Вып. 1. М.: Наука, 1988. С. 5–25.
4. *Редькин Н. П.* Надёжность и диагностика схем. М.: Изд-во МГУ, 1992. 192 с.
5. *Редькин Н. П.* О единичных диагностических тестах для однотипных константных неисправностей на выходах функциональных элементов // Вестник Московского университета. Сер. 1. Математика. Механика. 1992. № 5. С. 43–46.
6. *Попков К. А.* О точном значении длины минимального единичного диагностического теста для одного класса схем // Дискретный анализ и исследование операций. 2017. Т. 24. № 3. С. 80–103.
7. *Попков К. А.* О единичных диагностических тестах для схем из функциональных элементов в базисе Жегалкина // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2016. № 3. С. 3–18.

8. *Попков К. А.* Нижние оценки длин единичных тестов для схем из функциональных элементов // Дискретная математика. 2017. Т. 29. Вып. 2. С. 53–69.
9. *Романов Д. С., Романова Е. Ю.* Метод синтеза неизбыточных схем, допускающих короткие единичные диагностические тесты при константных неисправностях на выходах элементов // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2016. №. 2. С. 87–102.
10. *Попков К. А.* Короткие единичные тесты для схем при произвольных константных неисправностях на выходах элементов // Дискретная математика. 2018. Т. 30. Вып. 3. С. 99–116.
11. *Коваценко С. В.* Синтез легкотестируемых схем в базисе Жегалкина для инверсных неисправностей // Вестник Московского университета. Сер. 15. Вычислительная математика и кибернетика. 2000. № 2. С. 45–47.
12. *Романов Д. С.* Метод синтеза неизбыточных схем в стандартном базисе, допускающих единичные диагностические тесты длины два // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2016. № 3. С. 56–72.
13. *Романов Д. С.* Метод синтеза неизбыточных схем в базисе Жегалкина, допускающих единичные диагностические тесты длины один // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2015. № 4. С. 38–54.
14. *Любич И. Г., Романов Д. С.* О единичных диагностических тестах относительно инверсных неисправностей элементов в схемах над некоторыми базисами // Прикладная математика и информатика. Вып. 58. М.: МАКС Пресс, 2018. С. 47–61.
15. *Редькин Н. П.* О схемах, допускающих короткие единичные диагностические тесты // Дискретная математика. 1989. Т. 1. Вып. 3. С. 71–76.
16. *Попков К. А.* Синтез легкотестируемых схем при однотипных константных неисправностях на входах и выходах элементов // Интеллектуальные системы. Теория и приложения. 2018. Т. 23. Вып. 3. С. 131–147.
17. *Попков К. А.* Синтез легкотестируемых схем при произвольных константных неисправностях на входах и выходах элементов // Прикладная дискретная математика. 2019. № 43. С. 78–100.
18. *Яблонский С. В.* Введение в дискретную математику. М.: Наука, 1986. 384 с.
19. *Бородина Ю. В.* О схемах, допускающих единичные тесты длины 1 при константных неисправностях на выходах элементов // Вестник Московского университета. Сер. 1. Математика. Механика. 2008. № 5. С. 49–52.

REFERENCES

1. *Chegis I. A. and Yablonskiy S. V.* Logicheskie sposoby kontrolya raboty elektricheskikh skhem [Logical methods of control of work of electric circuits]. Trudy Mat. Inst. Steklov, 1958, vol. 51, pp. 270–360. (in Russian)
2. *Yablonskiy S. V.* Nadezhnost' i kontrol' upravlyayushchikh sistem [Reliability and verification of control systems]. Materialy Vsesoyuznogo seminara po diskretnoy matematike i ee prilozheniyam (Moscow, 31 Jan.–2 Feb. 1984). Moscow., MSU Publ., 1986, pp. 7–12. (in Russian)
3. *Yablonskiy S. V.* Nekotorye voprosy nadezhnosti i kontrolya upravlyayushchikh sistem [Some questions of reliability and verification of control systems]. Matematicheskie Voprosy Kibernetiki, iss. 1, Moscow, Nauka Publ., 1988, pp. 5–25. (in Russian)
4. *Red'kin N. P.* Nadezhnost' i diagnostika skhem [Circuits Reliability and Diagnostics]. Moscow, MSU Publ., 1992. 192 p. (in Russian)
5. *Red'kin N. P.* O edinichnykh diagnosticheskikh testakh dlya odnotipnykh konstantnykh neispravnostey na vykhodakh funktsional'nykh elementov [On single diagnostic tests for one-

type stuck-at faults at outputs of logic gates]. Vestnik MSU, Ser. 1, 1992, no. 5, pp. 43–46. (in Russian)

6. *Popkov K. A.* On the exact value of the length of the minimal single diagnostic test for a particular class of circuits. J. Appl. Industr. Math., 2017, vol. 11, no. 3, pp. 431–443.
7. *Popkov K. A.* O edinichnykh diagnosticheskikh testakh dlya skhem iz funktsional'nykh elementov v bazise Zhegalkina [On single diagnostic tests for logic networks in the Zhegalkin basis]. Izvestiya Vysshikh Uchebnykh Zavedeniy. Povolzhskiy Region. Fiziko-Matematicheskie Nauki, 2016, no. 3, pp. 3–18. (in Russian)
8. *Popkov K. A.* Lower bounds for lengths of single tests for Boolean circuits. Discrete Math. Appl., 2019, vol. 29, iss. 1, pp. 23–33.
9. *Romanov D. S. and Romanova E. Yu.* Metod sinteza neizbytochnykh skhem, dopuskayuyshchikh korotkie edinichnye diagnosticheskie testy pri konstantnykh neispravnostyakh na vykhodakh elementov [A method of synthesis of irredundant logic networks allowing short single diagnostic tests under stuck-at faults at outputs of gates]. Izvestiya Vysshikh Uchebnykh Zavedeniy. Povolzhskiy Region. Fiziko-Matematicheskie Nauki, 2016, no. 2, pp. 87–102. (in Russian)
10. *Popkov K. A.* Korotkie edinichnye testy dlya skhem pri proizvol'nykh konstantnykh neispravnostyakh na vykhodakh elementov [Short single tests for logic networks under arbitrary stuck-at faults at outputs of gates]. Diskretnaya Matematika, 2018, vol. 30, iss. 3, pp. 99–116. (in Russian)
11. *Kovatsenko S. V.* Sintez legkotestiruemykh skhem v bazise Zhegalkina dlya inversnykh neispravnostey [Synthesis of easily testable logic networks in the Zhegalkin basis for inverse faults]. Vestnik MSU, Ser. 15, 2000, no. 2, pp. 45–47. (in Russian)
12. *Romanov D. S.* Metod sinteza neizbytochnykh skhem v standartnom bazise, dopuskayuyshchikh edinichnye diagnosticheskie testy dliny dva [A method of synthesis of irredundant logic networks in the standard basis allowing single diagnostic tests of the length two]. Izvestiya Vysshikh Uchebnykh Zavedeniy. Povolzhskiy Region. Fiziko-Matematicheskie Nauki, 2016, no. 3, pp. 56–72. (in Russian)
13. *Romanov D. S.* Metod sinteza neizbytochnykh skhem v bazise Zhegalkina, dopuskayuyshchikh edinichnye diagnosticheskie testy dliny odin [A method of synthesis of irredundant logic networks in the Zhegalkin basis allowing single diagnostic tests of the length one]. Izvestiya Vysshikh Uchebnykh Zavedeniy. Povolzhskiy Region. Fiziko-Matematicheskie Nauki, 2015. no. 4, pp. 38–54. (in Russian)
14. *Lyubich I. G. and Romanov D. S.* Single fault diagnostic tests for inversion faults of circuit elements over some bases. Comput. Math. and Modelling, 2019, vol. 30, iss. 1, pp. 36–47.
15. *Red'kin N. P.* On schemes permitting short single diagnostic tests. Discrete Math. Appl., vol. 1, iss. 3, pp. 263–270.
16. *Popkov K. A.* Sintez legkotestiruemykh skhem pri odnotipnykh konstantnykh neispravnostyakh na vkhodakh i vykhodakh elementov [Synthesis of easily testable logic networks under one-type stuck-at faults at inputs and outputs of gates]. Intellektual'nye Sistemy. Teoriya i Prilozheniya, 2018, vol. 23, iss. 3, pp. 131–147. (in Russian)
17. *Popkov K. A.* Sintez legkotestiruemykh skhem pri proizvol'nykh konstantnykh neispravnostyakh na vkhodakh i vykhodakh elementov [Synthesis of easily testable logic networks under arbitrary stuck-at faults at inputs and outputs of gates]. Prikladnaya Diskretnaya Matematika, 2019, no. 43, pp. 78–100. (in Russian)
18. *Yablonskiy S. V.* Vvedenie v diskretnuyu matematiku [Introduction to Discrete Mathematics]. Moscow, Nauka Publ., 1986. 384 p. (in Russian)
19. *Borodina Yu. V.* Circuits admitting single-fault tests of length 1 under constant faults at outputs of elements. Mosc. Univ. Mat. Bull., 2008, vol. 63, iss. 5, pp. 202–204.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 591.68

АССОЦИАТИВНЫЙ ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ДЛЯ ДИНАМИЧЕСКОЙ ОБРАБОТКИ ДЕРЕВА КРАТЧАЙШИХ ПУТЕЙ ПОСЛЕ ДОБАВЛЕНИЯ НОВОЙ ДУГИ

А. Ш. Непомнящая, Т. В. Снытникова

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

Строится ассоциативный параллельный алгоритм для динамической обработки дерева кратчайших путей после добавления новой дуги к ориентированному взвешенному графу. Кратко описывается STAR-машина, которая моделирует работу ассоциативных (контекстно-адресуемых) параллельных систем с простейшими процессорными элементами и вертикальной обработкой информации. Описывается используемая структура данных и её свойства. Ассоциативный параллельный алгоритм представляется в виде процедуры InsertArcSPT, корректность которой доказывается. Показано, что на STAR-машине эта процедура выполняется за время $O(hk)$, где h — число битов, которое требуется для кодирования длины максимального кратчайшего пути; k — число вершин, для которых вычисляются новые кратчайшие пути после добавления новой дуги к исходному графу. Приводятся результаты тестирования реализации алгоритма на графическом ускорителе.

Ключевые слова: *ориентированный взвешенный граф, матрица смежности, инкрементальный алгоритм, аффектная вершина, вертикальная обработка данных, ассоциативный параллельный процессор.*

DOI 10.17223/20710410/46/5

ASSOCIATIVE PARALLEL ALGORITHM FOR DYNAMIC UPDATE OF SHORTEST PATHS TREE AFTER INSERTING AN ARC

A. Sh. Nepomniaschaya, T. V. Snytnikova

The Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia

E-mail: anep@ssd.sscc.ru, snytnikovat@ssd.sscc.ru

The paper proposes an associative parallel algorithm for dynamic update of the shortest paths tree after inserting an arc into a directed weighted graph. First we shortly recall the STAR-machine that simulates the functioning of associative parallel processors. The associative parallel algorithm is given as a procedure InsertArcSPT, correctness of which is proved. Now we give a short description of the associative incremental algorithm. Let the arc (i, j) be added to the graph G. Then we check whether the shortest distance from the root to the vertex j decreases if the shortest

path to the vertex j includes the arc (i, j). If this is true, then the vertex j becomes affected and the new shortest distance to this vertex is written in the matrix of the shortest distances. Then we replace the previous arc in the tree with the arc (i, j). After that, we calculate the distance to those vertexes v for which there is an arc (j, v). Then the all vertexes the distances to which decrease become affected. After that, the new distances are written in the corresponding rows of the shortest distance matrix. Moreover the corresponding arcs are included in the shortest paths tree. We have shown that the time complexity of this procedure is $O(hk)$, while the time complexity of a static associative version of Dijkstra algorithm is $O(hn)$. Here h is the number of bits for coding the maximum of the shortest paths length, n is the number of all graph vertexes and k is the number of affected vertexes for which the new distances are computed. This procedure was tested on the NVIDIA GEFORCE 920M using the implementation of STAR-machine on the GPU. Performance was evaluated on R-MAT graphs which simulate real graphs from social networks and the Internet. Graphs were generated by the GraphHPC-1.0 package with the following parameters: the number of vertexes is defined by a power of two (from 11 to 13), the average degree of vertexes connectivity is 32. We use two modes: zero weighting arcs are added (pessimistic mode) and arcs with random weight are added (realistic mode). In the experiments, we take into account as the runtime of the procedure and the number of affected vertexes. For each test, $\approx 10\% n$ runs were performed. After that, the runs were distributed by the number of affected vertexes. The shortest paths and distances do not change in most runs (more than 50 % in the case of the pessimistic mode and more than 88 % in the case of the realistic mode). In 99 % cases, the number of affected vertexes does not exceed 5 in the first mode and 2 in the second mode. Note that the associative algorithm traverses the graph along the vertexes (outgoing arcs are processed in parallel). While in the sequential version, the graph is traversed along arcs, and the number of arcs that need to be checked can go up to 2000 and 100 accordingly. Also we note that in average the dynamic version runs about 500 times faster in the pessimistic mode and about 1000 times faster in the realistic mode than the static parallel version of Dijkstra's algorithm.

Keywords: *oriented weighted graph, adjacency matrix, incremental algorithm, affected vertex, vertical data processing, associative parallel processor.*

Введение

Ассоциативные (контекстно-адресуемые) параллельные процессоры типа SIMD принадлежат к классу мелкозернистых параллельных систем с последовательно-поразрядной (вертикальной) обработкой информации и простейшими процессорными элементами. В работе [1] такие процессоры называют «системами вертикальной обработки». Перечислим основные достоинства этой архитектуры: параллелизм по данным на базовом уровне, использование двумерных таблиц в качестве простой и естественной структуры данных, параллельный поиск по содержимому памяти и выполнение базовых операций поиска за время, которое пропорционально числу битовых столбцов заданной матрицы, но не числу её строк [2].

В работах [3, 4] построена абстрактная модель типа SIMD (STAR-машина), которая моделирует работу систем вертикальной обработки. Эта модель использует группу элементарных операций, которые позволяют обрабатывать таблицы по содержимому памяти. Для представления ассоциативных параллельных алгоритмов построен язык высокого уровня STAR. На STAR-машине ассоциативные параллельные алгоритмы представляются в виде соответствующих процедур, корректность которых доказывает-

ся. В [4] построены базовые ассоциативные параллельные алгоритмы, которые используются для проектирования ассоциативных алгоритмов для различных приложений. Следуя Фостеру [5], мы полагаем, что каждая элементарная операция STAR-машины выполняется за единицу времени. Отметим, что системы вертикальной обработки наилучшим образом приспособлены для решения задач на графах. Для STAR-машины построены как новые ассоциативные параллельные алгоритмы на графах, так и ассоциативные версии хорошо известных последовательных алгоритмов на графах. В первую очередь были построены ассоциативные версии группы классических алгоритмов на графах. Эта группа, в частности, включает ассоциативные версии алгоритма Дейкстры для нахождения кратчайших путей из заданной вершины [6], алгоритма Флойда для нахождения кратчайших путей между любыми парами вершин [7], алгоритмов Краскала и Прима — Дейкстры для нахождения минимального остовного дерева [8].

Особый интерес представляет реализация на STAR-машине динамических алгоритмов на графах. Целью динамического алгоритма является обработка решения проблемы после динамических изменений в графе быстрее, чем перевычисление графа целиком после каждого локального изменения в нём самым быстрым статическим алгоритмом. На STAR-машине построена группа динамических алгоритмов на графах, которая, в частности, включает ряд ассоциативных алгоритмов для динамической обработки кратчайших путей.

Задача нахождения кратчайших путей возникает в различных приложениях. Известны следующие версии этой проблемы: нахождение кратчайших путей из одной вершины (the single-source shortest paths problem), нахождение подграфа кратчайших путей ориентированного взвешенного графа с единственным стоком (the single-sink shortest paths problem) и нахождение кратчайших путей между каждой парой вершин (the all-pairs shortest paths problem). Типичными операциями для преобразования кратчайших путей являются добавление или удаление одной дуги либо изменение веса одной дуги. Если граф представляет коммуникационную или транспортную сеть, то добавление или удаление дуги отражает такие реальные изменения в сети, как добавление или удаление связей во время существования сети. Алгоритм называется *полностью динамическим* (fully dynamic), если он позволяет выполнять любую последовательность упомянутых операций. Алгоритм называется *инкрементальным*, если он допускает только добавление дуги или уменьшение веса дуги. Если алгоритм допускает только удаление дуги или увеличение веса дуги, то он называется *декрементальным*.

Перечислим группу алгоритмов для динамической обработки кратчайших путей, которые были представлены на STAR-машине. В работах [9, 10] построены ассоциативные версии алгоритмов Рамалингама для динамической обработки подграфа кратчайших путей ориентированного взвешенного графа с одним стоком после добавления к графу новой дуги и после удаления из него одной дуги. На STAR-машине эти версии представлены соответственно в виде процедур InsertArc и DeleteArc, корректность которых доказана. В работах [11, 12] построены ассоциативные версии алгоритмов Рамалингама для динамической обработки кратчайших путей между любыми парами вершин после удаления одной дуги и после добавления новой дуги к заданному графу. Эти версии используют модификацию соответствующих алгоритмов для динамической обработки подграфа кратчайших путей с выделенным стоком после удаления одной дуги и после добавления новой дуги к заданному графу. Ассоциативные версии построены в виде процедур DeleteEdge и InsertEdge, корректность которых доказана. В [13] построен ассоциативный параллельный алгоритм для динамической обработки

дерева кратчайших путей ориентированного графа с неотрицательными весами дуг после удаления из графа одной дуги. Ассоциативный параллельный алгоритм представлен в виде процедуры DeleteArcSPT, корректность которой доказана. Показано, что эта процедура выполняется на STAR-машине за время $O(hk)$, где h — число битов, которое требуется для кодирования максимума кратчайших расстояний от корня, а k — число вершин, для которых строятся новые кратчайшие пути и вычисляются новые кратчайшие расстояния. Показано также, что выполнение процедуры DeleteArcSPT проще, чем процедуры DeleteArc.

В данной работе строится ассоциативный параллельный алгоритм для динамической обработки дерева кратчайших путей ориентированного графа с неотрицательными весами дуг после добавления к графу новой дуги. Алгоритм представлен на STAR-машине в виде процедуры InsertArcSPT, корректность которой доказана. Показано, что эта процедура выполняется на STAR-машине за время $O(hk)$. В работе показано, как построить структуру данных, которая используется в ассоциативном алгоритме, и что выполнение процедуры InsertArcSPT проще, чем процедуры InsertArc. Приводятся результаты тестирования инкрементального алгоритма InsertArcSPT на графическом ускорителе и сравнивается время выполнения этой процедуры с временем работы статического алгоритма DistSPT [13].

1. Модель ассоциативного параллельного процессора

Приведём краткое описание STAR-машины, которая моделирует работу систем с вертикальной обработкой данных. Модель использует некоторые свойства машины Staran [5] и отечественного ассоциативного процессора [14]. В работе [15] приводится сравнение нашей модели с другими моделями ассоциативной обработки.

STAR-машина определяется как абстрактная модель типа SIMD с вертикальной обработкой информации, основными компонентами которой являются: последовательное устройство управления, в котором записаны программа и скалярные константы; устройство ассоциативной обработки, состоящее из p одноразрядных процессорных элементов; матричная память.

Чтобы моделировать обработку информации в матричной памяти, STAR-машина использует типы данных **slice**, **word** и **table**. С помощью переменной типа **slice** моделируется доступ к таблице по столбцам, а типа **word** — по строкам. С каждой переменной типа **table** ассоциируется матрица из n строк и k столбцов, где $n \leqslant p$. С каждой переменной типа **slice** ассоциируется последовательность из p компонентов, принадлежащих множеству $\{0, 1\}$. Для простоты переменную типа **slice** условимся называть *слайсом*. Более подробно STAR-машина и операции языка STAR описаны в [16].

Приведём группу базовых процедур [4, 6], которые будем использовать в данной работе. С помощью глобального слайса X будем указывать *позиции* анализируемых строк в соответствующей процедуре. В [4, 6] показано, что каждая базовая процедура выполняется за время $O(k)$.

Процедура TCOPY1(T, j, h, F) записывает h столбцов из матрицы T, начиная с $(1 + (j - 1)h)$-го столбца, в результирующую матрицу F, $j \geqslant 1$.

Процедура TMERGE(T, X, F) записывает строки матрицы T, отмеченные «1» в слайсе X, в соответствующие строки матрицы F. Остальные строки матрицы F не меняются.

Процедура SETMIN(T, F, X, Z) определяет позиции строк матрицы T, которые меньше соответствующих строк матрицы F. Она возвращает слайс Z, в котором $Z(j) = 1$ тогда и только тогда, когда $\mathrm{ROW}(j, T) < \mathrm{ROW}(j, F)$ и $X(j) = 1$.

Процедура ADDC(T, X, v, F) одновременно добавляет двоичное слово v к тем строкам матрицы T, которые отмечены «1» в слайсе X, и записывает результат в соответствующие строки матрицы F. Остальные строки матрицы F состоят из нулей.

2. Основные понятия

Пусть $G = (V, E, wt)$ — *ориентированный взвешенный граф*, в котором $V = \{1, 2, \ldots, n\}$ — множество вершин, $E \subseteq V \times V$ — множество ориентированных рёбер (*дуг*) и $wt(e)$ — функция веса. Полагаем, что $|V| = n$ и $|E| = m$. Будем рассматривать графы, дуги которых имеют неотрицательный вес. Будем считать, что $wt(i, j) = \infty$, если $(i, j) \notin E$.

Значение бесконечности выбирается как $\sum_{i=1}^{n} c_i$, где c_i — максимальный вес дуг, выходящих из вершины i в G. Обозначим через h число битов, используемых для кодирования этой величины.

Пусть $e = (i, j)$ — дуга, которая ориентирована от вершины i к вершине j. При этом вершина j называется *головой* дуги e, а вершина i — её *хвостом*.

Кратчайшим путём из вершины v_1 до вершины v_k в графе G назовём последовательность вершин $v_1, v_2, \ldots, v_k$, в которой $(v_i, v_{i+1}) \in E$ для $1 \leqslant i < k-1$ и сумма весов дуг минимальна. Обозначим через $\mathrm{dist}(l)$ *длину* кратчайшего пути из вершины v_1 до вершины l.

Деревом кратчайших путей T_s с корнем s назовём связный подграф без циклов, который содержит все вершины графа G и в котором путь от корня до любой вершины является *минимальным*.

Пусть дуга (i, j) добавлена к графу G. Вершина y называется *аффектной* в графе G, если кратчайший путь в дереве T_s до вершины y включает дугу (i, j) и длина этого пути меньше, чем длина кратчайшего пути до вершины y, не включающего дугу (i, j).

3. Структура данных

Будем использовать следующую структуру данных:

— матрица смежности G размера $n \times n$, i-й столбец которой, $i = 1, \ldots, n$, хранит с помощью «1» головы дуг, выходящих из вершины i;
— матрица SPT размера $n \times n$, i-й столбец которой хранит головы дуг, выходящих из вершины i и принадлежащих дереву кратчайших путей;
— матрица $Weight$ размера $n \times hn$, элементами которой являются веса дуг. Она состоит из n полей, каждое поле — из h битовых столбцов. Вес дуги (i, j) записывается в j-ю строку i-го поля;
— матрица $Dist$ размера $n \times h$, i-я строка которой хранит кратчайшее расстояние от корня до вершины i;
— слайс $AffectedV$, который хранит с помощью «1» позиции аффектных вершин.

Поясним, как построить приведённую структуру данных. Исходный граф задаём в виде матрицы весов $Weight$ и матрицы смежности G, которая получается из матрицы весов с помощью вспомогательной процедуры Adj [7]. Слайс $AffectedV$ получаем с помощью вспомогательной процедуры UpdateOutgoingEdges, которая приведена в работе далее. Матрицу кратчайших расстояний $Dist$ и матрицу SPT для дерева кратчайших путей будем получать с помощью вспомогательной процедуры DistSPT [13], которая

является специальной реализацией на STAR-машине классического алгоритма Дейкстры [6] для нахождения кратчайших расстояний.

4. Инкрементальный ассоциативный алгоритм для обработки дерева кратчайших путей

Приведём содержательное описание ассоциативного инкрементального алгоритма для обработки дерева кратчайших путей T_s ориентированного графа. Пусть дуга (i, j) добавляется к графу G. Проверяем, уменьшится ли кратчайшее расстояние от корня до вершины j, если кратчайший путь до вершины j будет включать дугу $(i.j)$. Если это верно, то вершина j становится аффектной и новое кратчайшее расстояние до вершины j записывается в матрицу кратчайших расстояний. Поскольку в каждую вершину дерева входит единственная дуга, то в дереве кратчайших путей удаляем дугу, которая первоначально заходила в вершину j, и добавляем туда дугу (i, j). Понятно, что необходимо вычислить расстояние до тех вершин, в которые заходят дуги из вершины j. Если для каких-то вершин расстояние уменьшилось, то эти вершины необходимо пометить как аффектные, расстояния до них записать в соответствующие строки матрицы кратчайших расстояний, а соответствующие дуги внести в дерево кратчайших путей.

Ассоциативный параллельный алгоритм для обработки дуг, выходящих из текущей аффектной вершины, реализован на STAR-машине в виде процедуры UpdateOutgoingEdges (алгоритм 1).

Алгоритм 1. Ассоциативный параллельный алгоритм UpdateOutgoingEdges

Вход: $k, h, G, Weight, Dist, SPT$.
Выход: $Dist, SPT, Y$.

1: С помощью слайса X сохранить позиции дуг, выходящих из k в G.
2: С помощью матрицы $R1$ сохранить веса дуг, выходящих из k в $Weight$.
3: С помощью матрицы $R1$ сохранить новые расстояния от корня до тех вершин графа G, позиции которых отмечены «1» в слайсе X.
4: С помощью слайса Y сохранить номера тех вершин из слайса X, для которых новые кратчайшие расстояния от корня уменьшились // *слайс Y будет хранить новые аффектные вершины.*
5: Записать в матрицу $Dist$ новые кратчайшие расстояния от корня до тех аффектных вершин, позиции которых хранятся в слайсе Y.
6: Перестроить дерево кратчайших путей следующим образом. Пусть слово $w0$ содержит единственную «1» в k-м бите. Тогда в дереве кратчайших путей SPT записать слово $w0$ в каждую строку, отмеченную «1» в слайсе Y.

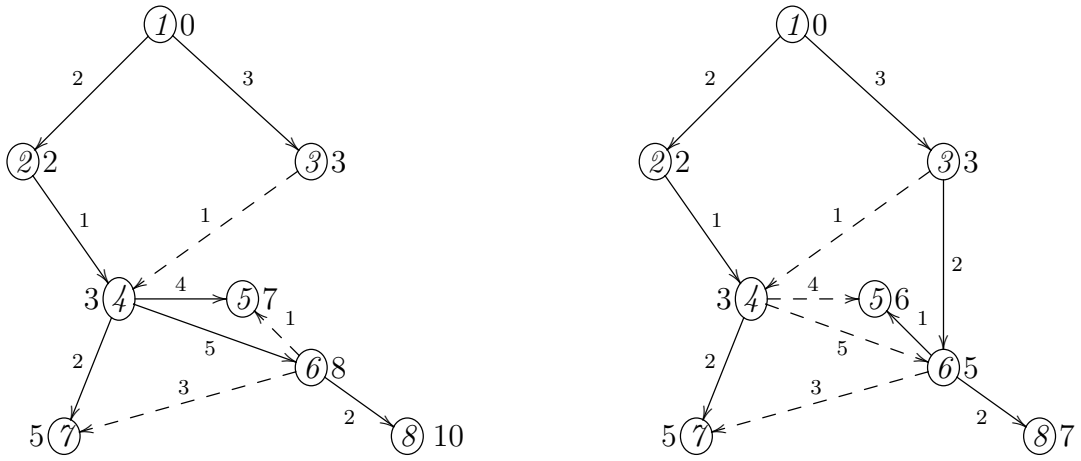
Ассоциативный параллельный алгоритм для динамической обработки дерева кратчайших путей после добавления к графу G дуги (i, j) весом $v1$ реализован на STAR-машине в виде процедуры InsertArcSPT (алгоритм 2).

На рис. 1 показан пример изменения кратчайших расстояний и дерева кратчайших расстояний после добавления дуги (*3*,*6*) с весом 2. Метками у вершин показаны кратчайшие расстояния, а сплошными стрелками обозначены дуги, принадлежащие дереву кратчайших расстояний. Аффектными будут вершины *6*, *5* и *8*. При этом дуга (*6*,*8*) принадлежала дереву кратчайших расстояний и до добавления дуги, в то время как дуга (*6*,*5*) заменила в дереве дугу (*4*,*5*).

Алгоритм 2. Ассоциативный параллельный алгоритм InsertArcSPT

Вход: $(i,j), v1, G, Weight, Dist, SPT$.
Выход: $Dist, SPT$.

1: Включить дугу (i,j) в матрицу смежности G, а её вес $v1$ добавить в матрицу $Weight$.
2: С помощью слова $v2$ сохранить кратчайшее расстояние до вершины i, с помощью слова $v3$ — длину нового пути до вершины j.
3: С помощью слова $v4$ сохранить старое кратчайшее расстояние до вершины j.
4: **Если** $v3 < v4$, **то**
5: включить вершину j в слайс $AffectedV$ и записать новое кратчайшее расстояние $v3$ до вершины j в матрицу $Dist$;
6: удалить из матрицы SPT дугу, которая заходила в вершину j перед добавлением к графу G дуги (i,j), и включить в матрицу SPT дугу (i,j).
7: **Пока** $AffectedV \neq \varnothing$:
8: в слайсе $AffectedV$ выделить верхнюю вершину k с помощью операции STEP;
9: выполнить процедуру UpdateOutgoingEdges$(k, h, SPT, Weight, Dist, Y)$;
10: добавить в слайс $AffectedV$ новые аффектные вершины, позиции которых хранятся в слайсе Y.



а *б*

Рис. 1. Дерево кратчайших расстояний до (*а*) и после (*б*) добавления дуги (*3,6*) с весом 2

5. Выполнение инкрементального ассоциативного алгоритма на STAR-машине

Приведём вспомогательную процедуру UpdateOutgoingEdges. Зная аффектную вершину k и текущие матрицы SPT, $Weight$ и $Dist$, она возвращает слайс Y, который хранит аффектные вершины, смежные с вершиной k, и матрицу $Dist$, которая хранит новые кратчайшие расстояния от корня до этих вершин.

```
procedure UpdateOutgoingEdges(k,h:integer; G:table; Weight:table;
  var Dist:table; var SPT:table; var Y:slice(G));
/* Здесь k - это аффектная вершина.*/
var v: word(Dist); w:word(SPT);
```

```
 X:\,slice(SPT);
R1,R2: table;
i:integer;
 1. Begin X:=COL(k,G);
 2.    TCOPY1(Weight,k,h,R1);
/* Матрица R1 хранит веса дуг, выходящих из вершины k. */
 3.    v:=ROW(k,Dist);
/* Слово v хранит новое кратчайшее расстояние до вершины k. */
 4.    ADDC(R1,X,v,R2);
/* Матрица R1 хранит новые расстояния от корня до вершин,
выходящих из вершины k в матрице SPT. */
 5.    SETMIN(R2,Dist,X,Y);
/* Слайс Y хранит новые аффектные вершины.\,*/
 6.    TMERGE(R2,Y,Dist);
/* Новые расстояния до аффектных вершин смежных с вершиной k
записываются в соответсвующие строки матрицы Dist. */
 7.    CLR(w); w(k):='1'; X:=Y;
/* Вершина k помечается как вероятный отец для этих вершин */
 8.    While SOME(X) do
 9.    begin
10.       i:=FND(X);
11.       ROW(i,SPT):=w;
12.    end;
13. End;
```

Утверждение 1. Пусть граф G задаётся матрицами G, $Weight$, SPT и $Dist$. Пусть также заданы добавляемая дуга (i, j) и аффектная вершина k. Тогда после выполнения процедуры UpdateOutgoingArcs слайс Y будет хранить аффектные вершины, смежные с вершиной k, матрица $Dist$ — новые кратчайшие расстояния от корня до этих вершин, а матрица SPT — обновлённое дерево кратчайших путей.

Доказательство. Пусть вершина r смежна с вершиной k в матрице G и является аффектной, однако после выполнения процедуры UpdateOutgoingArcs r-й бит слайса Y равен нулю. Докажем, что это противоречит выполнению процедуры UpdateOutgoingArcs.

Действительно, после выполнения строк 1–2 слайс X хранит позиции дуг, выходящих в матрице G из вершины k, а матрица $R1$ — веса этих дуг. По предположению вершина r смежна с вершиной k в матрице G. Поэтому $X(r) = 1$, а в r-й строке матрицы $R1$ записан вес дуги (k, r). После выполнения строк 3–6 в r-й строке матрицы $R2$ будет записано новое кратчайшее расстояние от корня до вершины r. Все вершины, для которых это расстояние меньше $\mathrm{ROW}(r, Dist)$, отмечаются «1» в слайсе Y. Поэтому после выполнения строк 5–6 получаем, что $Y(r) = 1$ и новое кратчайшее расстояние от корня до вершины r записано в r-ю строку матрицы $Dist$. После выполнения строки 7 слово w содержит единственную «1» в k-м бите, а в слайсе X отмечены все вершины, смежные k, для которых расстояние уменьшилось. Тогда в дереве кратчайших путей SPT записываем слово w в каждую строку i, отмеченную «1» в слайсе X, таким образом вставляя дуги (k, i) в дерево кратчайших расстояний. Это противоречит нашему допущению. ■

Приведём основную процедуру InsertArcSPT. Зная параметр h, добавляемую в граф дугу (i, j) весом $v1$, текущие матрицы G, $Weight$, $Dist$ и SPT, она возвращает изменённые матрицы G, SPT, $Weight$ и $Dist$.

```
procedure InsertArcSPT(i,j,h:integer; v1:word(Dist);
    var Weight:table; var G,SPT:table; var Dist:table);
/* Дуга (i,j) весом v1 добавляется в граф G.*/
var k,r,l,l1,l2:integer;
   AffectedV,X,Y:slice(G);
   v2,v3,v4:word(Dist);  v5:word(G);
   v6:word(Weight);
 1. Begin X:=COL(i,G); X(j):='1'; COL(i,G):=X;
 /* Дуга (i,j) добавлена в граф G. */
 2.   l1:=1+(i-1)h; l2:=ih;
 3.   v6:=ROW(j,Weight); Rep(l1,l2,v1,v6);
 4.   ROW(j,Weight):=v6;
 /* В матрицу Weight записывается вес новой дуги (i,j). */
 5.   CLR(AffectedV); v2:=ROW(i,Dist);
 /* Строка v2 хранит кратчайшее расстояние от корня до вершины i. */
 6.   v3:=ADD(v1,v2);
 /* Строка v3 хранит новое кратчайшее расстояние от корня до вершины j.*/
 7.   v4:=ROW(j,Dist);
 /* Строка v4 хранит старое кратчайшее расстояние от корня до вершины j. */
 8.   if LESS(v3,v4) then
 9.   begin AffectedV(j):='1';
10.       ROW(j,Dist):=v3;
 /* Новое кратчайшее расстояние от корня до вершины j
 записывается в матрицу Dist. */
11.       CLR(v5); v5(i):='1'; ROW(j,SPT):=v5;
 /* Дуга (i,j) добавлена в матрицу SPT. */
12.       while SOME(AffectedV) do
13.       begin k:=STEP(AffectedV);
14.          UpdateOutgoingEdges(k,h,G,Weight,Dist,SPT,Y);
 /* Слайс Y хранит новые аффектные вершины, которые будут добавляться
 в слайс AffectedV. */
15.          AffectedV:=AffectedV or Y;
16.       end;
17.   end;
18. End;
```

Теорема 1. Пусть заданы параметр h и дуга (i, j) веса $v1$, которая добавляется к графу G. Тогда после выполнения процедуры InsertArcSPT матрицы G, $Weight$, SPT и $Dist$ будут задавать текущее состояние изменённого графа.

Доказательство. Индукция по числу аффектных вершин $r \geqslant 0$, которые образуются после добавления дуги (i, j) к графу G.

Б а з и с докажем для случая, когда $r \leqslant 1$. После выполнения строк 1–4 дуга (i, j) будет добавлена к графу G, а её вес — в матрицу $Weight$. После выполнения строк 5–6 слайс $AffectedV$ состоит из нулей, переменная $v3$ хранит новое кратчайшее расстояние от корня до вершины j, а $v4$ — старое кратчайшее расстояние до вершины j. Если

$v3 \geqslant v4$, то переходим на конец процедуры (строка 18). В противном случае переходим на строку 9. После выполнения строк 9–10 позиция вершины j отмечается «1» в слайсе $AffectedV$ и в матрицу $Dist$ записывается новое кратчайшее расстояние до вершины j. Осталось внести изменения в матрицу SPT — включить новую дугу (i, j). После выполнения строки 11 в j-ю строку матрицы SPT записывается слово, содержащее единственную «1» в i-й позиции. Так как $AffectedV \neq \varnothing$, выполняем цикл `while SOME(AffectedV) do` (строки 12–16). После выполнения строки 13 получаем, что $k = j$ и $AffectedV = \varnothing$. По предположению индукции имеется не более одной аффектной вершины. Поэтому в результате выполнения строки 14 получаем, что $Y = \varnothing$. Тогда после выполнения строки 15 переходим на конец процедуры.

Ш а г и н д у к ц и и. Пусть утверждение теоремы выполняется для случая $r \geqslant 1$. Докажем справедливость теоремы для $r + 1$ аффектных вершин. По индуктивному предположению после обработки первых r аффектных вершин дуга (i, j) будет добавлена в матрицу G, её вес — в матрицу $Weight$, дуга (l, j) удалена из матрицы SPT, а дуга (i, j) туда добавлена. Кроме того, в матрицу $Dist$ будут записаны новые кратчайшие расстояния до первых r аффектных вершин. Поскольку после обработки первых r аффектных вершин в слайсе $AffectedV$ останется единственная аффектная вершина, то после выполнения строки 14 по утверждению 1 для этой вершины будут обновлены матрицы $Dist$ и SPT. ■

6. Результаты тестирования на GPU

Для использования ассоциативных параллельных алгоритмов на практике разработана реализация STAR-машины на графических ускорителях:

1) в виде библиотеки на CUDA реализованы типы данных для языка STAR и простейшие операции над ними [16, 17];
2) библиотека стандартных процедур языка STAR эффективно реализована на GPU [18].

С помощью реализации STAR-машины проведено тестирование алгоритма InsertArcSPT на графическом ускорителе NVIDIA GEFORCE 920M.

Оценка производительности проводилась на R-MAT-графах. Они хорошо моделируют реальные графы из социальных сетей и интернета, а также являются достаточно сложными для анализа. Генерация графов производилась пакетом GraphHPC-1.0 [19] со следующими параметрами: количество вершин задаётся степенью двойки, средняя степень связности вершины равна 32. В таком R-MAT-графе имеется одна большая связная компонента и некоторое количество небольших связных компонент или висящих вершин.

Напомним, что порядок сложности ассоциативного параллельного алгоритма и его реализации на CUDA отличаются на $O(\log_{64} n)$, если в алгоритме используются операции, критичные к синхронизации($STEP(X)$, $FND(X)$, $SOME(X)$). Таким образом, сложность реализации алгоритма InsertArcSPT равна $O(hk \log_{64}(n))$, а реализации DistSPT — $O(hn \log_{64}(n))$. Здесь n — число вершин в графе; k — число аффектных вершин; h — число битов, которое требуется для кодирования длины максимального кратчайшего пути (по умолчанию 32). В экспериментах использовались два режима:

— **R-MAT*** — добавлялись дуги с весом 0 (пессимистичный сценарий);
— **R-MAT** — добавлялись дуги со случайным весом (реалистичный сценарий).

Учитывалось не только время работы процедуры, но и количество аффектных вершин (вершин, для которых длина кратчайшего пути изменилась). Для каждого теста про-

водилось $\approx n/10$ запусков с добавлением дуги. После этого запуски распределялись по количеству аффектных вершин.

На рис. 2 показано распределение запусков по количеству аффектных вершин при добавлении дуги с весом 0, а на рис. 3 — дуги со случайным весом. Видно, что для R-MAT-графов в большинстве случаев (более 50 % при добавлении дуги с нулевым весом и более 88 % — со случайным весом) кратчайшие пути и расстояния не изменяются. В 99 % случаев число аффектных вершин не превышает 5 в первом режиме и 2 — во втором. Отметим, что в отличие от ассоциативного алгоритма, ведущего обход графа по вершинам (исходящие дуги обрабатываются параллельно), в последовательном варианте обход графа совершается по дугам, и число дуг, которые необходимо проверить, может доходить до 2000 в первом режиме и до 100 — во втором.

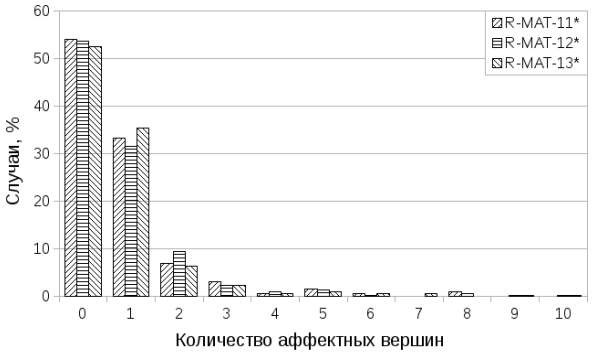


Рис. 2. Распределение запусков при добавлении дуги с весом 0

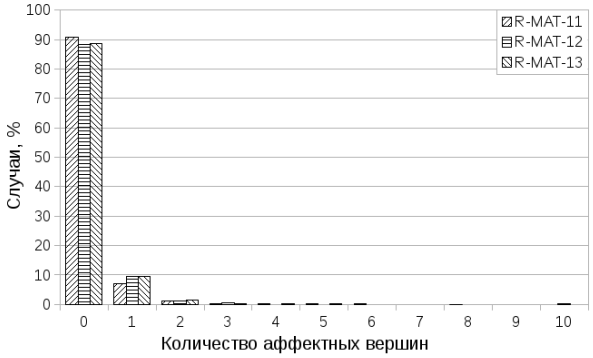


Рис. 3. Распределение запусков при добавлении дуги со случайным весом

В таблице приводятся среднее время выполнения процедуры DistSPT и максимальное время выполнения процедуры InsertArcSPT в двух режимах добавления дуг. Здесь n — число вершин в графе, k — число аффектных вершин в худшем случае (число аффектных вершин при добавлении дуги с максимальным временем обработки/максимальное число аффектных вершин). Время и количество аффектных вершин для режима R-MAT* отмечены «*».

Сравнение времени работы статического и динамического ассоциативных алгоритмов на R-MAT-графах

Граф	n	DistSPT	InsertArcSPT*	k*	InsertArcSPT	k
R-MAT-11	2048	6,171	0,012	6/8	0,009	1/8
R-MAT-12	4096	9,643	0,017	5/10	0,012	3/10
R-MAT-13	8192	29,475	0,038	7/15	0,020	4/13

Таким образом, при использовании динамического алгоритма время определения кратчайших расстояний уменьшается на несколько порядков, так как InsertArcSPT в большей степени зависит от числа аффектных вершин, чем от общего числа вершин в графе.

Заключение

Построен ассоциативный параллельный алгоритм для динамической обработки дерева кратчайших путей после добавления в ориентированный граф одной дуги. Этот алгоритм использует простую структуру данных, которая позволяет выполнять доступ к данным по содержимому памяти. Ассоциативный алгоритм представлен на STAR-

машине в виде процедуры InsertArcSPT, корректность которой доказана. Процедура выполняется на STAR-машине за время $O(hk)$, где h — число битов для кодирования бесконечности, а k — число аффектных вершин, для которых строятся новые кратчайшие пути. Полученная оценка является оптимальной, поскольку параметр h не меняется с ростом k, а обрабатывать надо все аффектные вершины.

Представлены результаты тестирования реализации данного алгоритма на графических ускорителях. При тестировании на R-MAT-графах фиксировалось не только время выполнения алгоритма, но и количество аффектных вершин. Получено, что для этого класса графов в более чем 50 % случаев при добавлении дуги кратчайшие расстояния не изменяются. Количество аффектных вершин не превышает 5 в 99 % случаев.

Приведено сравнение времени работы динамического алгоритма InsertArcSPT с временем работы статического алгоритма DistSPT.

ЛИТЕРАТУРА

1. *Fet Y. I.* Vertical processing systems: a survey // IEEE Micro. 1995. V. 15. Iss. 1. P. 65–75.
2. *Potter J. L.* Associative Computing: a Programming Paradigm for Massively Parallel Computers. Boston: Perseus Publishing, 1991. 304 p.
3. *Nepomniaschaya A. Sh.* Language STAR for associative and parallel computation with vertical data processing // Parallel Computing Technologies. Singapore: World Scientific, 1991. P. 258–265.
4. *Nepomniaschaya A. Sh.* Basic associative parallel algorithms for vertical processing systems // Bulletin of the Novosibirsk Computing Center. 2009. Ser. Comp. Sci. No. 9. P. 63–77.
5. *Foster C. C.* Content Addressable Parallel Processors. N.Y.: John Wiley & Sons, 1976. 233 p.
6. *Nepomniaschaya A. Sh. and Dvoskina M. A.* A simple implementation of Dijkstra's shortest path algorithm on associative parallel processors // Fundamenta Informaticae. IOS Press, 2000. V. 43. P. 227–243.
7. *Nepomniaschaya A. S.* Solution of path problems using associative parallel processors // Proc. ICPADS'97. Seoul: IEEE Computer Society Press,1997. P. 610–617.
8. *Непомнящая А. Ш.* Сравнение алгоритмов Прима — Дейкстры и Краскала с помощью ассоциативного параллельного процессора // Кибернетика и системный анализ. 2000. № 2. С. 19–27.
9. *Непомнящая А. Ш.* Ассоциативная версия алгоритма Рамалингама для динамической обработки подграфа кратчайших путей после добавления к графу новой дуги // Кибернетика и системный анализ. 2012. № 3. С. 45–57.
10. *Nepomniaschaya A. S.* Efficient parallel implementation of the Ramalingam decremental algorithm for updating the shortest paths subgraph // Computing and Informatics. 2013. V. 32. P. 331–354.
11. *Nepomniaschaya A. S.* Associative version of the Ramalingam decremental algorithm for the dynamic all-pairs shortest-path problem // Bulletin of the Novosibirsk Computing Center. 2016. Ser. Comp. Sci. No. 39. P. 37–50.
12. *Nepomniaschaya A. S.* Associative version of the Ramalingam incremental algorithm for the dynamic all-pairs shortest-path problem // Bulletin of the Novosibirsk Computing Center. 2016. Ser. Comp. Sci. No. 40. P. 75–86.
13. *Непомнящая А. Ш.* Ассоциативный параллельный алгоритм для динамической обработки дерева кратчайших путей // Моделирование и анализ информационных систем. 2013. Т. 20. № 2. С. 5–22.

14. *Mirenkov N. N.* The Siberian approach for an open-system high-performance computing architecture // Computing and Control Engineering J. 1992. V. 3. No. 3. P. 137–142.
15. *Непомнящая А. Ш., Владыко М. А.* Сравнение моделей ассоциативного вычисления // Программирование. 1997. № 6. С. 41–50.
16. *Снытникова Т. В., Непомнящая А. Ш.* Решение задач на графах с помощью STAR-машины, реализуемой на графических ускорителях // Прикладная дискретная математика. 2016. № 3(33). С. 98–115.
17. *Snytnikova T. V. and Snytnikov A. V.* Implementation of the STAR-machine on GPU // Bulletin of the Novosibirsk Computing Center. 2016. Ser. Comp. Sci. No. 39. P. 51–60.
18. *Снытникова Т. В.* Реализация модели ассоциативных вычислений на GPU: библиотека базовых процедур языка STAR // Вычислительные методы и программирование. Новые вычислительные технологии. 2018. № 19(1). С. 85–95.
19. `http://www.dislab.org/GraphHPC-2018/contest/GraphHPC-1.0.tgz` — GraphHPC-1.0. 2018.

REFERENCES

1. *Fet Y. I.* Vertical processing systems: a survey. IEEE Micro, 1995, vol. 15, iss. 1, pp. 65–75.
2. *Potter J. L.* Associative Computing: a Programming Paradigm for Massively Parallel Computers. Boston, Perseus Publishing, 1991. 304 p.
3. *Nepomniaschaya A. Sh.* Language STAR for associative and parallel computation with vertical data processing. Parallel Computing Technologies, Singapore, World Scientific, 1991, pp. 258–265.
4. *Nepomniaschaya A. Sh.* Basic associative parallel algorithms for vertical processing systems. Bulletin of the Novosibirsk Computing Center, 2009, ser. Comp. Sci, no. 9. pp. 63–77.
5. *Foster C. C.* Content Addressable Parallel Processors. N.Y., John Wiley & Sons, 1976. 233 p.
6. *Nepomniaschaya A. Sh. and Dvoskina M. A.* A simple implementation of Dijkstra's shortest path algorithm on associative parallel processors. Fundamenta Informaticae, IOS Press, 2000, vol. 43, pp. 227–243.
7. *Nepomniaschaya A. S.* Solution of path problems using associative parallel processors. Proc. ICPADS'97, Seoul, IEEE Computer Society Press, 1997, pp. 610–617.
8. *Nepomniaschaya A. S.* Comparison of performing the Prim — Dijkstra algorithm and the Kruskal algorithm by means of associative parallel processors. Cybernetics and System Analysis, 2000, no. 2, pp. 19–27.
9. *Nepomniaschaya A. S.* Associative version of the Ramalingam algorithm for the dynamic update of the shortest paths subgraph after inserting a new edge. Cybernetics and System Analysis, 2012, no. 3, pp. 45–57.
10. *Nepomniaschaya A. S.* Efficient parallel implementation of the Ramalingam decremental algorithm for updating the shortest paths subgraph. Computing and Informatics, 2013, vol. 32, pp. 331–354.
11. *Nepomniaschaya A. S.* Associative version of the Ramalingam decremental algorithm for the dynamic all-pairs shortest-path problem. Bulletin of the Novosibirsk Computing Center, 2016, ser. Comp. Sci., no. 39, pp. 37–50.
12. *Nepomniaschaya A. S.* Associative version of the Ramalingam incremental algorithm for the dynamic all-pairs shortest-path problem // Bulletin of the Novosibirsk Computing Center, 2016, ser. Comp. Sci., no. 40, pp. 75–86.
13. *Nepomniaschaya A. S.* Assotsiativnyy parallel'nyy algoritm dlya dinamicheskoy obrabotki dereva kratchayshikh putey [Associative parallel algorithm for dynamic update shortest

paths tree]. Modeling and Analysis of Information Systems, 2013, vol. 20, no. 2, pp. 5–22. (in Russian)

14. *Mirenkov N. N.* The Siberian approach for an open-system high-performance computing architecture. Computing and Control Engineering J., 1992, vol. 3, no. 3, pp. 137–142.
15. *Nepomniaschaya A. S. and Vladyko M. A.* A comparison of associative computation models. Programming and Computer Software, 1997, no. 6, pp. 319–324.
16. *Snytnikova T. V. and Nepomniaschaya A. Sh.* Resheniye zadach na grafakh s pomoshch'yu STAR-mashiny, realizuyemoy na graficheskikh uskoritelyakh [Solution of graph problems by means of the STAR-machine being implemented on GPUs]. Prikladnaya Diskretnaya Matematika, 2016, no. 3(33), pp. 98–115. (in Russian)
17. *Snytnikova T. V. and Snytnikov A. V.* Implementation of the STAR-machine on GPU. Bulletin of the Novosibirsk Computing Center, 2016, ser. Comp. Sci., no. 39, pp. 51–60.
18. *Snytnikova T. V.* Realizatsiya modeli assotsiativnyh vychisleniy na GPU: biblioteka bazovyh protsedur yazyka STAR [Implementation of an associative-computing model on GPU: a basic procedure library of the STAR language]. Numerical Methods and Programming. Advanced Computing, 2018, no. 19(1), pp. 85–95.(in Russian)
19. `http://www.dislab.org/GraphHPC-2018/contest/GraphHPC-1.0.tgz` — GraphHPC-1.0, 2018.

УДК 510.52

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ КЛАСТЕРИЗАЦИИ ГРАФОВ[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

Генерический подход к алгоритмическим проблемам предложен Мясниковым, Каповичем, Шуппом и Шпильрайном в 2003 г. В рамках этого подхода рассматривается поведение алгоритмов на множествах почти всех входов. В данной работе изучается генерическая сложность проблемы кластеризации графов. В этой задаче структура взаимосвязей объектов задаётся с помощью графа, вершины которого соответствуют объектам, а рёбра соединяют похожие объекты. Требуется разбить множество объектов на попарно непересекающиеся группы (кластеры) так, чтобы минимизировать число связей между кластерами и число недостающих связей внутри кластеров. Доказывается, что при условии $\mathrm{P} \neq \mathrm{NP}$ и $\mathrm{P} = \mathrm{BPP}$ для проблемы кластеризации графов не существует полиномиального сильно генерического алгоритма. Сильно генерический алгоритм решает проблему не на всём множестве входов, а на подмножестве, последовательность частот которого при увеличении размера экспоненциально быстро сходится к 1.

Ключевые слова: *генерическая сложность, кластеризация графа.*

DOI 10.17223/20710410/46/6

ON GENERIC COMPLEXITY OF THE GRAPH CLUSTERING PROBLEM

A. N. Rybalov

Sobolev Institute of Mathematics, Omsk, Russia

E-mail: alexander.rybalov@gmail.com

Generic-case approach to algorithmic problems was suggested by Miasnikov, Kapovich, Schupp and Shpilrain in 2003. This approach studies behavior of algorithms on typical (almost all) inputs and ignores the rest of inputs. In this paper, we study the generic complexity of the problem of clustering graphs. In this problem the structure of relations of objects is presented as a graph: vertices correspond to objects, and edges connect similar objects. It is required to divide a set of objects into disjoint groups (clusters) to minimize the number of connections between clusters and the number of missing links within clusters. It is proved that under the condition $\mathrm{P} \neq \mathrm{NP}$ and $\mathrm{P} = \mathrm{BPP}$, for the graph clustering problem there is no polynomial strongly generic algorithm. A strongly generic algorithm solves a problem not on the whole set of inputs, but on its subset, in which the sequence of frequencies of inputs converges exponentially fast to 1 with increasing its size.

Keywords: *generic complexity, graph clustering.*

[1]Работа поддержана грантом РНФ № 18-71-10028.

Введение

Теория генерической вычислимости и сложности вычислений предложена в 2003 г. [1]. В рамках этого подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Такие входы образуют так называемое генерическое множество. Понятие «почти все» формализуется введением естественной меры на множестве входных данных. С точки зрения практики алгоритмы, решающие быстро проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов. Классическим примером такого алгоритма является симплекс-метод — он за полиномиальное время решает задачу линейного программирования для большинства входных данных, но имеет экспоненциальную сложность в худшем случае. Более того, может так оказаться, что проблема трудноразрешима или вообще неразрешима в классическом смысле, но легкоразрешима на генерическом множестве. Отметим, что похожий подход для изучения проблем оптимизации был предложен ранее Э. Х. Гимади, Н. И. Глебовым и В. А. Перепелицей [2].

Одной из важных проблем машинного обучения является проблема кластеризации графов. В этой задаче структура взаимосвязей объектов задается с помощью графа, вершины которого соответствуют объектам, а ребра соединяют похожие объекты. Требуется разбить множество объектов на попарно непересекающиеся группы (кластеры) так, чтобы минимизировать число связей между кластерами и число недостающих связей внутри кластеров. В работах [3 – 9] доказана NP-трудность проблемы кластеризации графа для различных её постановок. Таким образом, при условии $\mathrm{P} \neq \mathrm{NP}$ полиномиального алгоритма для решения этой задачи не существует. А при условии совпадения классов P и BPP (класс проблем, решаемых за полиномиальное время вероятностными алгоритмами) для неё не существует и полиномиальных вероятностных алгоритмов. Имеются серьёзные доводы в пользу равенства $\mathrm{P} = \mathrm{BPP}$. В частности, доказано [10], что это равенство следует из весьма правдоподобных гипотез о вычислительной сложности некоторых трудных проблем.

Данная работа посвящена изучению генерической сложности задачи кластеризации графов. Доказывается, что при условии $\mathrm{P} \neq \mathrm{NP}$ и $\mathrm{P} = \mathrm{BPP}$ для проблемы кластеризации графов не существует полиномиального сильно генерического алгоритма. Сильно генерический алгоритм решает проблему не на всём множестве входов, а на подмножестве, последовательность частот которого при увеличении размера экспоненциально быстро сходится к 1.

1. Генерические алгоритмы

Пусть I — некоторое множество входов, I_n — подмножество входов размера n. Для подмножества $S \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где $S_n = S \cap I_n$ — множество входов из S размера n. Заметим, что $\rho_n(S)$ — это вероятность попасть в S при случайной и равновероятной генерации входов из I_n. *Асимптотической плотностью* S назовём предел

$$\rho(S) = \overline{\lim_{n\to\infty}} \rho_n(S).$$

Верхний предел здесь нужен потому, что часто при кодировании входных данных не для каждого n существуют коды размера n. Множество S называется *пренебрежи-*

мым, если $\rho(S) = 0$, и *сильно пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к 0, т. е. существуют константы σ, $0 < \sigma < 1$, и $C > 0$ такие, что для любого n имеет место $\rho_n(S) < C\sigma^n$.

Алгоритм $\mathcal{A}$ с множеством входов I и множеством выходов $J \cup \{?\}$ ($? \notin J$) называется (*сильно*) *генерическим*, если

1) $\mathcal{A}$ останавливается на всех входах из I;
2) множество $\{x \in I : \mathcal{A}(x) = ?\}$ является (сильно) пренебрежимым.

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если для всех $x \in I$ выполнено

$$(\mathcal{A}(x) = y \in J) \Rightarrow (f(x) = y).$$

Ситуация $\mathcal{A}(x) = ?$ означает, что $\mathcal{A}$ не может вычислить функцию f на аргументе x. Но условие 2 гарантирует, что $\mathcal{A}$ корректно вычисляет f на почти всех входах (входах из генерического множества). Различие между генерически разрешимыми проблемами и сильно генерически разрешимыми проблемами поясняется в работе [11].

2. Проблема кластеризации графа

Здесь и далее будем рассматривать неориентированные графы без петель и кратных рёбер. Граф называется *кластерным*, если каждая его компонента связности является полным графом. Обозначим через $\mathcal{M}(V)$ множество всех кластерных графов на множестве вершин V. Если $G_1 = (V, E_1)$ и $G_2 = (V, E_2)$ — графы на одном и том же множестве вершин V, то *расстояние* $\rho(G_1, G_2)$ между ними есть число несовпадающих рёбер в графах G_1 и G_2, то есть

$$\rho(G_1, G_2) = |E_1 \triangle E_2| = |E_1 \setminus E_2| + |E_2 \setminus E_1|.$$

Проблема кластеризации графа состоит в следующем. Задан граф $G = (V, E)$. Найти такой граф $M^* \in \mathcal{M}(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}(V)} \rho(G, M).$$

Лемма 1. Пусть G_1 и G_2 — два графа с непересекающимися множествами вершин и M^* — кластерный граф, являющийся решением проблемы кластеризации для графа $G_1 \cup G_2$. Тогда $M^* = M_1^* \cup M_2^*$, где M_i^* — решение проблемы кластеризации для графа G_i, $i = 1, 2$.

Доказательство. Пусть $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$. Допустим, что существует кластерный граф M, являющийся решением проблемы кластеризации для графа $G_1 \cup$ $\cup\, G_2$, такой, что

$$M = C_1 \cup C_2 \cup \ldots \cup C_m,$$

где C_i, $i = 1, \ldots, m$, — непересекающиеся полные компоненты связности, причём среди них есть компонента $C_k = K(V_c)$, которая имеет непустое пересечение как с графом G_1, так и с графом G_2. Заменим в кластерном графе M его компоненту C_k на два полных графа $C_{k,i} = K(V_c \cap V_i)$, $i = 1, 2$. Вершины первого лежат в графе G_1, а второго — в G_2. Обозначим через M' получившийся кластерный граф. Заметим, что

$$\rho(G_1 \cup G_2, M') < \rho(G_1 \cup G_2, M),$$

так как в новых компонентах $C_{k,i}$, $i = 1, 2$, присутствуют все рёбра старой компоненты C_k, обе вершины которых лежат либо в G_1, либо в G_2, и отсутствуют все рёбра

старой компоненты C_k, одна вершина которых лежит в G_1, а другая — в G_2. Последних рёбер нет и в графе $G_1 \cup G_2$.

Полученное противоречие показывает, что для кластерного графа M^*, являющегося решением проблемы кластеризации для графа $G_1 \cup G_2$, имеет место $M^* = M_1^* \cup M_2^*$, где M_i^* — решение проблемы кластеризации для графа G_i, $i = 1, 2$, так как иначе кластерный граф M_i^* можно заменить на более подходящий, уменьшив тем самым расстояние $\rho(G_1 \cup G_2, M^*)$. ■

3. Основной результат

Для изучения генерической сложности проблемы кластеризации графов будем использовать представление графов с помощью матриц смежности. Поскольку графы неориентированные, для кодирования графа с n вершинами достаточно верхней части матрицы, состоящей из $n(n-1)/2$ бит. Таким образом, будем считать, что размер графа с n вершинами равен $n(n-1)/2$.

Теорема 1. Если существует сильно генерический полиномиальный алгоритм, решающий проблему кластеризации графа, то существует вероятностный полиномиальный алгоритм, разрешающий эту проблему на всём множестве входов.

Доказательство. Допустим, что существует сильно генерический полиномиальный алгоритм $\mathcal{A}$, решающий проблему кластеризации графа. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, разрешающий эту проблему на всём множестве входов, который на графе G с n вершинами (размера $n(n-1)/2$) работает следующим образом:

1) Генерирует случайный граф H с $n^2 - n$ вершинами.
2) Запускает алгоритм $\mathcal{A}$ на графе $G \cup H$.
3) Если $\mathcal{A}(G \cup H) \neq ?$, то по решению проблемы кластеризации для графа $G \cup H$, согласно лемме 1, выдаёт решение проблемы кластеризации для графа G.
4) Если $\mathcal{A}(G \cup H) = ?$, то выдаёт ответ K_n.

Заметим, что алгоритм $\mathcal{B}$ выдаёт правильный ответ на шаге 3, а на шаге 4 может выдать неправильный ответ. Нужно доказать, что вероятность того, что ответ выдаётся на шаге 4, меньше $1/2$.

Граф $G \cup H$ имеет n^2 вершин, то есть его размер равен $m = (n^4 - n^2)/2$. Вероятность того, что для случайного графа $G \cup H$ имеет место $\mathcal{A}(G \cup H) = ?$, не больше

$$\frac{|\{G \in \mathcal{G} : \mathcal{A}(G) \neq ?\}_m|}{|\{G \cup H : H \in \mathcal{G}\}_m|} = \frac{|\{G \in \mathcal{G} : \mathcal{A}(G) \neq ?\}_m|}{|\mathcal{G}_m|} \frac{|\mathcal{G}_m|}{|\{G \cup H : H \in \mathcal{G}\}_m|}.$$

Так как множество $\{G \in \mathcal{G} : \mathcal{A}(G) \neq ?\}$ сильно пренебрежимое, существует константа $\alpha > 0$, такая, что

$$\frac{|\{G \in \mathcal{G} : \mathcal{A}(G) \neq ?\}_m|}{|\mathcal{G}_m|} < \frac{1}{2^{\alpha m}} = \frac{1}{2^{\alpha(n^4-n^2)/2}}$$

для любого n.

С другой стороны, так как граф H имеет $n^2 - n$ вершин, то

$$|\{G \cup H : H \in \mathcal{G}\}_m| = |\{H : H \in \mathcal{G}\}_{((n^2-n)^2-(n^2-n))/2}| = 2^{(n^4-2n^3+n)/2}.$$

Отсюда

$$\frac{|\mathcal{G}_m|}{|\{G \cup H : H \in \mathcal{G}\}_m|} = \frac{2^{(n^4-n^2)/2}}{2^{(n^4-2n^3+n)/2}} = 2^{(2n^3-n^2+n)/2}.$$

Поэтому искомая вероятность не больше

$$\frac{2^{(2n^3-n^2+n)/2}}{2^{\alpha(n^4-n^2)/2}} < \frac{1}{2}$$

при больших n. ■

Непосредственным следствием теоремы 1 является следующее утверждение.

Теорема 2. Если P $\neq$ NP и P = BPP, то не существует сильно генерического полиномиального алгоритма для решения проблемы кластеризации графов.

ЛИТЕРАТУРА

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
2. *Гимади Э.Х., Глебов Н.И., Перепелица В.А.* Алгоритмы с оценками для задач дискретной оптимизации // Проблемы кибернетики. 1975. Т. 31. С. 35–42.
3. *Kŕivanek M. and Morávek J.* NP-hard problems in hierarchical-tree clustering // Acta Informatica. 1986. V. 23. P. 311–323.
4. *Bansal N., Blum A., and Chawla S.* Correlation clustering // Machine Learning. 2004. V. 56. P. 89–113.
5. *Shamir R., Sharan R., and Tsur D.* Cluster graph modification problems // Discrete Appl. Math. 2004. V. 144. No. 1–2. P. 173–182.
6. *Агеев А. А., Ильев В. П., Кононов А. В., Талевнин А. С.* Вычислительная сложность задачи аппроксимации графов // Дискретный анализ и исследование операций. Сер. 1. 2006. Т. 13. № 1. С. 3–11.
7. *Ильев В. П., Ильева С. Д.* О задачах кластеризации графов // Вестник Омского университета. 2016. № 2. С. 16–18.
8. *Ильев А. В., Ильев В. П.* Об одной задаче кластеризации графа с частичным обучением // Прикладная дискретная математика. 2018. № 42. С. 66–75.
9. *Талевнин А. С.* О сложности задачи аппроксимации графов // Вестник Омского университета. 2004. № 4. С. 22–24.
10. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma // Proc. 29th STOC. El Paso: ACM, 1997. P. 220–229.
11. *Рыбалов А. Н.* О генерической сложности проблемы общезначимости булевых формул // Прикладная дискретная математика. 2016. № 2(32). С. 119–126.

REFERENCES

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
2. *Gimadi E.H., Glebov N.I., and Perepelitsa V.A.* Algoritmy s ocenkami dlya zadach diskretnoi optimizacii [Algorithms with bounds for problems of discrete optimization]. Problemy Kibernetiki, 1975, vol. 31, pp. 35–42. (in Russian)
3. *Kŕivanek M. and Morávek J.* NP-hard problems in hierarchical-tree clustering. Acta Informatica, 1986, vol. 23, pp. 311–323.
4. *Bansal N., Blum A., and Chawla S.* Correlation clustering. Machine Learning, 2004, vol. 56, pp. 89–113.
5. *Shamir R., Sharan R., and Tsur D.* Cluster graph modification problems. Discrete Appl. Math., 2004, vol. 144, no. 1–2, pp. 173–182.
6. *Ageev A. A., Il'ev V. P., Kononov A. V., and Talevnin A. S.* Computational complexity of the graph approximation problem. J. Appl. Ind. Math., 2007, vol. 1, no. 1, pp. 1–8.

7. *Il'ev V. P. and Il'eva S. D.* O zadachah klasterizacii grafov [On problems of graph clustering]. Vestnik Omskogo Universiteta, 2016, no. 2, pp. 16–18. (in Russian)
8. *Il'ev A. V. and Il'ev V. P.* Ob odnoi zadache klasterizacii grafa s chastichnym obucheniem [On a problem of graph clustering with partial learning]. Prikladnaya Diskretnaya Matematika, 2018, no. 42, pp. 66–75. (in Russian)
9. *Talevnin A. S.* O slozhnosti zadachi approksimacii grafov [On the complexity of the graph approximation problem]. Vestnik Omskogo Universiteta, 2004, no. 4, pp. 22–24. (in Russian)
10. *Impagliazzo R. and Wigderson A.* P = BPP unless E has subexponential circuits: Derandomizing the XOR Lemma. Proc. 29th STOC, El Paso, ACM, 1997, pp. 220–229.
11. *Rybalov A. N.* O genericheskoy slozhnosti problemy obshcheznachimosti bulevykh formul [On generic complexity of the validity problem for Boolean formulas]. Prikladnaya Diskretnaya Matematika, 2016, no. 2(32), pp. 119–126. (in Russian)

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.7

АЛГОРИТМЫ ВЫЧИСЛЕНИЯ КРИПТОГРАФИЧЕСКИХ ХАРАКТЕРИСТИК ВЕКТОРНЫХ БУЛЕВЫХ ФУНКЦИЙ[1]

Н. М. Киселева, Е. С. Липатова, И. А. Панкратова, Е. Е. Трифонова

Национальный исследовательский Томский государственный университет, г. Томск, Россия

Представлены алгоритмы вычисления следующих криптографических характеристик векторых булевых функций: порядка корреляционной иммунности, нелинейности, компонентной алгебраической иммунности и показателя дифференциальной равномерности. Компоненты векторной булевой функции перебираются в порядке, задаваемом кодом Грея. Приводятся результаты экспериментов для случайных векторных булевых функций, подстановок и двух специальных классов $\mathcal{K}_n$ и $\mathcal{S}_{n,k}$ обратимых векторных булевых функций от n переменных, координаты которых существенно зависят от всех и заданного числа $k < n$ переменных соответственно. Доказаны некоторые свойства дифференциальной равномерности для функций из классов $\mathcal{K}_n$ и $\mathcal{S}_{n,k}$.

Ключевые слова: *векторная булева функция, корреляционная иммунность, нелинейность векторной булевой функции, компонентная алгебраическая иммунность, показатель дифференциальной равномерности.*

DOI 10.17223/20710410/46/7

ALGORITHMS FOR COMPUTING CRYPTOGRAPHIC CHARACTERISTICS OF VECTORIAL BOOLEAN FUNCTIONS

N. M. Kiseleva, E. S. Lipatova, I. A. Pankratova, E. E. Trifonova

National Research Tomsk State University, Tomsk, Russia

E-mail: kiselyov-natalya@mail.ru, katrinelipatova@gmail.com, pank@mail.tsu.ru, lizatrif@gmail.com

There are presented algorithms for calculating the cryptographic characteristics of vectorial Boolean functions, such as the order of correlation immunity, nonlinearity, component algebraic immunity, and differential uniformity order. In these algorithms, the components of a vectorial Boolean function are enumerated according to the Gray code. Experimental results are given for random vectorial Boolean functions, permutations, and two known classes $\mathcal{K}_n$ and $\mathcal{S}_{n,k}$ of invertible vectorial Boolean functions in n variables with coordinates essentially depending on all variables and on k variables, $k < n$, respectively. Some properties of differential uniformity are theoretically

[1]Работа поддержана грантом РФФИ, проект № 17-01-00354.

proved for functions in $\mathcal{K}_n$ and $\mathcal{S}_{n,k}$, namely, the differential uniformity order δ_F equals 2^n for any $F \in \mathcal{S}_{n,k}$, and the inequality $2^n - 4(n-1) \leqslant \delta_F \leqslant 2^n - 4$ holds for any $F \in \mathcal{K}_n$.

Keywords: *vectorial Boolean function, nonlinearity, correlation immunity, component algebraic immunity, differential uniformity.*

Введение

Криптосистемы, стойкость которых основана на сложности решения систем нелинейных уравнений над конечным полем, являются предположительно стойкими к квантовым атакам [1]. К этому классу, в частности, относятся криптосистемы с функциональными ключами, построенные на векторных булевых функциях [2–4]. Для анализа стойкости таких криптосистем, помимо разработки специальных атак, нужно исследовать известные криптографические характеристики используемых функций. В данной работе приведены алгоритмы вычисления таких характеристик, как порядок корреляционной иммунности, нелинейность, компонентная алгебраическая иммунность и показатель дифференциальной равномерности. Алгоритмы достаточно «прямолинейны» и, скорее всего, не являются лучшими по сложности, однако их реализация позволила исследовать характеристики функций из разных классов и сформулировать ряд утверждений о свойствах функций этих классов. В п. 3 доказано несколько утверждений о дифференциальной равномерности подстановок, координатные функции которых существенно зависят от заданного числа переменных.

Приведём необходимые определения [5–7]. Пусть дана векторная булева функция $F = (f_1 \dots f_m) : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

Компонентой функции F называется булева функция $vF = v_1 f_1 \oplus \dots \oplus v_m f_m$, где $v = v_1 \dots v_m \in \mathbb{F}_2^m \setminus \{0^m\}$; 0^m — нулевой вектор длины m.

Порядком корреляционной иммунности $\mathrm{cor}(F)$, *нелинейностью* $N(F)$ и *компонентной алгебраической иммунностью* $\mathrm{AI}_{\mathrm{comp}}(F)$ векторной функции F называются минимальные порядок корреляционной иммунности, нелинейность и алгебраическая иммунность её компонент соответственно:

$$\mathrm{cor}(F) = \min_{v \in \mathbb{F}_2^m \setminus \{0^m\}} \mathrm{cor}(vF), \quad N(F) = \min_{v \in \mathbb{F}_2^m \setminus \{0^m\}} N(vF), \quad \mathrm{AI}_{\mathrm{comp}}(F) = \min_{v \in \mathbb{F}_2^m \setminus \{0^m\}} \mathrm{AI}(vF)$$

(определения соответствующих характеристик для *булевой* функции см. в [8, с. 277 и определение 2.50] и [5]).

Для функции F и векторов $a \in \mathbb{F}_2^n$, $b \in \mathbb{F}_2^m$ обозначим

$$\delta_F(a, b) = |\{x \in \mathbb{F}_2^n : F(x) \oplus F(x \oplus a) = b\}|.$$

Показателем дифференциальной равномерности функции F называется

$$\delta_F = \max_{a \neq 0^n, b} \delta_F(a, b).$$

Значения $\mathrm{cor}(F)$, $N(F)$ и $\mathrm{AI}_{\mathrm{comp}}(F)$ характеризуют стойкость криптосистемы с функцией F в качестве блока преобразования к корреляционному, линейному и алгебраическому методам криптоанализа соответственно, и они должны быть большими. Заметим, что для всех подстановок $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ имеет место $\mathrm{cor}(F) = 0$, поскольку для уравновешенных функций $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ верно неравенство $m \leqslant n - \mathrm{cor}(F)$ [5].

Показатель δ_F связан со способностью шифра противостоять дифференциальному криптоанализу в случаях, когда функция F используется в качестве блока замены в DES-подобном шифре [7] или (в общем случае) в произвольном итеративном блочном шифре с аддитивным раундовым ключом [9] — чем он меньше, тем лучше.

1. Алгоритмы вычисления криптографических характеристик

Для вычисления характеристик $\mathrm{cor}(F)$, $N(F)$ и $\mathrm{AI}_{\mathrm{comp}}(F)$ по определению нужно перебрать все компоненты vF функции F, $v \in \mathbb{F}_2^m \backslash \{0^m\}$. Чтобы минимизировать время вычисления компонент, будем перебирать векторы v в соответствии с *кодом Грея* — в этом случае «соседние» значения v различаются ровно в одном двоичном разряде, а значит, очередная компонента функции F получается из предыдущей прибавлением одной координатной функции (алгоритм 1).

Алгоритм 1. Перебор компонент векторной булевой функции

Вход: функция $F = (f_1 \dots f_m) : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

1: Создать булеву функцию $f : \mathbb{F}_2^n \to \mathbb{F}_2$, $f := \mathrm{const}\, 0$, и вектор $v_0 \in \mathbb{F}_2^m$, $v_0 := 0^m$.
2: **Для** $i = 1, \dots, 2^m - 1$:
3: $\quad v_i := i \oplus (i \gg 1)$ *// преобразование двоичного кода в код Грея;* $\gg$ *— операция сдвига вправо;* i *рассматривается и как целое число, и как булев вектор длины* m.
4: $\quad$ Положить k равным номеру (единственной) единицы в векторе $v_{i-1} \oplus v_i$;
5: $\quad f := f \oplus f_k$ *// очередная компонента.*
6: $\quad$ Обработка f.

Согласно [5], порядок корреляционной иммунности $\mathrm{cor}(F)$ функции $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ равен t, если и только если: 1) $W_{vF}(u) = 0$ для всех наборов $u \in \mathbb{F}_2^n$ веса от 1 до t включительно и всех компонент vF и 2) $W_{vF}(u) \neq 0$ для некоторого набора u веса $t+1$ и некоторой компоненты vF, где $W_{vF}(u) = \sum\limits_{x \in \mathbb{F}_2^n} (-1)^{vF(x) \oplus ux}$ — коэффициент преобразования Уолша — Адамара функции vF. Таким образом, для вычисления $\mathrm{cor}(F)$ нужно найти ненулевой вектор u минимального веса, на котором преобразование Уолша — Адамара хотя бы одной компоненты принимает ненулевое значение. Другими словами, нам интересны такие значения u, что $W_{vF}(u) = 0$ *для всех* компонент. Поэтому, перебирая компоненты vF, будем «накапливать» в одном массиве дизъюнкцию коэффициентов $W_{vF}(u)$, $u \in \mathbb{F}_2^n$ (интерпретируя целые числа как булевы векторы) и только потом проанализируем полученный вектор (алгоритм 2).

Алгоритм 2. Вычисление $\mathrm{cor}(F)$

Вход: функция $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

1: Создать массив целых чисел D размера 2^n, обнулить все его элементы.
2: Перебираем компоненты vF, как в алгоритме 1.
3: **Для** каждой компоненты:
4: $\quad w := W_{vF}$; $D := D \vee w$ (поэлементно) *// шаг* 6 *алгоритма* 1.
5: **Для всех** $i = 1, \dots, n$:
6: $\quad$ **Для всех** векторов $u \in \mathbb{F}_2^n$ веса i:
7: $\quad\quad$ **если** $D_u \neq 0$, **то выход**, ответ: $i - 1$.
8: **Выход**, ответ: n.

Оценим сложность алгоритма 2:

— всего компонент $(2^m - 1)$, переход к следующей компоненте (сложение текущей с координатной функцией) — $O(2^n)$ операций, вычисление преобразования Уолша — Адамара по схеме Грина — $O(2^n n)$ [8], поэлементная дизъюнкция векторов — $O(2^n)$ операций; итого сложность шагов 2–4 составляет $O(2^{n+m}n)$ операций;

— на шагах 5–7 в худшем случае (для функции-константы) придётся перебрать $2^n - 1$ векторов.

Таким образом, сложность алгоритма 2 равна $O(2^{n+m}n)$.

Нелинейность $N(F)$ функции $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ тоже можно вычислить с использованием преобразования Уолша — Адамара по следующей формуле [5]:

$$N(F) = 2^{n-1} - \frac{1}{2} \max_{\substack{u \in \mathbb{F}_2^n, \\ v \in \mathbb{F}_2^m \setminus \{0^m\}}} W_{vF}(u). \tag{1}$$

Получаем алгоритм 3 вычисления нелинейности.

Алгоритм 3. Вычисление $N(F)$

Вход: функция $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

1: $c := 0$.
2: Перебираем компоненты vF, как в алгоритме 1.
3: **Для** каждой компоненты:
4: $w := W_{vF}$; $d := \max\limits_{i=0,\ldots,2^n-1} |w_i|$; **если** $d > c$, **то** $c := d$.

Выход: $N(F) = 2^{n-1} - c/2$ (по формуле (1)).

Сложность алгоритма 3 равна $O(2^{n+m}n)$.

Для описания алгоритма вычисления компонентной алгебраической иммунности $\mathrm{AI}_{\mathrm{comp}}(F)$ введём следующие понятия [5]. *Аннигилятором подмножества* $A \subseteq \mathbb{F}_2^n$ называется любая булева функция от n переменных, не равная тождественно 0 и принимающая значение 0 на всех наборах из A. *Алгебраической иммунностью* множества A называется минимальная из алгебраических степеней аннигиляторов, не равных тождественно 0; обозначается $\mathrm{AI}(A)$. Для булевой функции f от n переменных обозначим $M_f^\sigma = \{x \in \mathbb{F}_2^n : f(x) = \sigma\}$, $\sigma \in \{0, 1\}$. Тогда

$$\mathrm{AI}_{\mathrm{comp}}(F) = \min_{v \in \mathbb{F}_2^m \setminus \{0^m\}} \mathrm{AI}(vF) = \min_{v \in \mathbb{F}_2^m \setminus \{0^m\}} \min(\mathrm{AI}(M_{vF}^0), \mathrm{AI}(M_{vF}^1)). \tag{2}$$

В соответствии с (2) получаем алгоритм 4 вычисления компонентной алгебраической иммунности.

Алгебраическую иммунность множества $A = \{a_1, \ldots, a_k\} \subseteq \mathbb{F}_2^n$ будем искать методом неопределённых коэффициентов [10] (алгоритм 5). Для того чтобы определить, существует ли аннигилятор множества A степени не выше d, построим матрицу $B(A, d)$, столбцы которой соответствуют элементам множества A, строки — всевозможным мономам $m_1, \ldots, m_t$ от переменных $x_1, \ldots, x_n$ степеней от 0 до d, где $t = \sum\limits_{i=0}^{d} \binom{n}{i}$; на пересечении строки i и столбца j запишем значение $m_i(a_j)$. Если $\operatorname{rang} B(A, d) < t$, то существует нетривиальная нулевая линейная комбинация строк матрицы: $m_{i_1} \oplus \ldots \oplus m_{i_s} = 0^k$; тогда функция $g(x) = m_{i_1}(x) \oplus \ldots \oplus m_{i_s}(x)$ — аннигилятор множества A.

Алгоритм 4. Вычисление $\mathrm{AI}_{\mathrm{comp}}(F)$

Вход: функция $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

1: $c := \lceil n/2 \rceil$ // верхняя граница $\mathrm{AI}_{\mathrm{comp}}(F)$ [10].
2: Перебираем компоненты vF, как в алгоритме 1.
3: **Для** каждой компоненты:
4: $m_0 := \mathrm{AI}(M_{vF}^0)$; $m_1 := \mathrm{AI}(M_{vF}^1)$ // по алгоритму 5;
5: $c := \min(c, m_0, m_1)$.

Выход: $\mathrm{AI}_{\mathrm{comp}}(F) = c$.

Алгоритм 5. Вычисление алгебраической иммунности множества

Вход: множество $A \subseteq \mathbb{F}_2^n$, $|A| = k$.

1: **Если** $A = \varnothing$, **то выход**, ответ: $\mathrm{AI}(A) = 0$.
2: Построить матрицу $B = B(A, 1)$; $d := 1$; $t := n + 1$.
3: **Если** $\operatorname{rang} B < t$, **то выход**, ответ: $\mathrm{AI}(A) = d$.
4: $d := d + 1$; $t := t + \binom{n}{d}$; **если** $t > k$, **то выход**, ответ: $\mathrm{AI}(A) = d$.
5: Добавить к матрице B строки, соответствующие мономам степени d, перейти к п. 3.

Сложность алгоритма 5 при вычислении ранга методом Гаусса составляет $\mathrm{O}(k^3 d)$. Отметим, что при использовании этого алгоритма для вычисления $\mathrm{AI}_{\mathrm{comp}}(F)$ можно ввести дополнительное ограничение на шаге 4 после увеличения d: «**если** $d \geqslant \lceil n/2 \rceil$, **то выход**, ответ: $\mathrm{AI}(A) = d$». Сложность алгоритма 4 равна $\mathrm{O}(2^{m+3n} n)$.

Алгоритм 6 вычисления показателя дифференциальной равномерности получаем непосредственно из его определения.

Алгоритм 6. Вычисление показателя δ_F

Вход: функция $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

1: Создать вспомогательный массив целых чисел DDT размера 2^m, обнулить все его элементы.
2: $\delta := 0$.
3: **Для всех** $a \in \mathbb{F}_2^n \setminus \{0^n\}$:
4: **Для всех** $x \in \mathbb{F}_2^n$:
$b := F(x) \oplus F(x \oplus a)$; $DDT[b] := DDT[b] + 1$;
5: $d := \max\limits_{b \in \mathbb{F}_2^m} DDT[b]$; **если** $d > \delta$, **то** $\delta := d$.
6: Обнулить массив DDT.

Выход: $\delta_F = \delta$.

Сложность алгоритма 6 равна $\mathrm{O}(2^{n+\max(n,m)})$.

2. Результаты экспериментов

Все алгоритмы реализованы на языке ЛЯПАС-Т [11] и исследованы в компьютерном эксперименте на случайных функциях, подстановках и двух специальных классах обратимых функций, координаты которых зависят от заданного числа переменных. Определим эти классы [12, 13].

Для $n \in \mathbb{N}$, $n \geqslant 3$, рассмотрим класс функций $\mathcal{K}_n$, функции $F = (f_1 \dots f_n) : \mathbb{F}_2^n \to \mathbb{F}_2^n$ в котором получаются из тождественной подстановки G на $\mathbb{F}_2^n$ с помощью n следующих независимых транспозиций: выбираем n непересекающихся множеств $M_i = \{x^{(i)}, y^{(i)}\}$, где векторы $\{x^{(i)}, y^{(i)}\}$ соседние по i-й координате, и полагаем $F(x^{(i)}) = G(y^{(i)})$, $F(y^{(i)}) = G(x^{(i)})$, $i = 1, \dots, n$; $F(x) = G(x)$ для всех $x \notin \bigcup\limits_{i=1}^{n} M_i$. По построению, для любого $a = a_1 \dots a_n \in \mathbb{F}_2^n$ имеет место

$$f_i(a) = \begin{cases} a_i \oplus 1, & \text{если } a \in M_i, \\ a_i & \text{иначе.} \end{cases} \tag{3}$$

Доказано [12], что все координаты функций из $\mathcal{K}_n$ существенно зависят от всех n переменных и $\mathcal{K}_n \neq \varnothing$ для всех $n \geqslant 3$.

Для $k, n \in \mathbb{N}$, $3 \leqslant k \leqslant n$, построим функцию $F = (f_1 \dots f_n) : \mathbb{F}_2^n \to \mathbb{F}_2^n$ так. Пусть $H = (h_1 \dots h_k)$ — некоторая функция из $\mathcal{K}_k$. Положим $f_i(x_1, \dots, x_n) = h_i(x_1, \dots, x_k)$, $i = 1, \dots, k$ (переменные $x_{k+1}, \dots, x_n$ у функций $f_1, \dots, f_k$ фиктивные), и $f_i = x_i \oplus$ $\oplus\, \varphi_i(x_1, \dots, x_{i-1})$ для $i = k+1, \dots, n$, где $\varphi_{k+1}, \dots, \varphi_n$ — произвольные функции, существенно зависящие ровно от $(k-1)$ из своих переменных. Обозначим класс функций, построенных таким образом, через $\mathcal{S}_{n,k}$. В [13, утверждение 3] доказано, что функции из $\mathcal{S}_{n,k}$ являются подстановками на $\mathbb{F}_2^n$; все их координаты существенно зависят ровно от k переменных. Необходимость ограничивать количество существенных переменных у координат векторных булевых функций может возникнуть, в частности, при построении криптосистем, где такие функции являются ключами [2–4], а значит, нужно уметь их эффективно задавать и быстро вычислять.

Эксперименты на функциях $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ дали следующие результаты.

1) Порядок корреляционной иммунности случайной функции, как правило, равен 0, редко — 1.
2) Для нелинейности:
   - при фиксированном n с ростом m значение $N(F)$ убывает; например, при $n = 5$ среднее значение $N_{\text{ср}}(F) = 6{,}6$ при $m = 5$ и $N_{\text{ср}}(F) \leqslant 2$ при $m \geqslant 18$;
   - если $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ — подстановка, то её нелинейность растёт с ростом n, приближаясь к теоретической верхней границе $N_{\max} = 2^{n-1} - 2^{n/2-1}$: от 47 % при $n = 5$ до 93 % при $n = 12$;
   - если $F \in \mathcal{K}_n$, то $N(F) = 2$ (доказано в [14]); отсюда следует, что $N(F) \leqslant 2^{n-k+1}$ для $F \in \mathcal{S}_{n,k}$, поскольку первые k координат такой функции F получаются добавлением $(n-k)$ фиктивных переменных, каждая из которых увеличивает нелинейность функции в два раза. В экспериментах для функций $F \in \mathcal{S}_{n,k}$ всегда выполняется $N(F) = 2^{n-k+1}$, кроме случаев $k = 3$ (всегда получаем 0) и $k = 4$ (получаем 0 или 2^{n-3}).
3) Для алгебраической иммунности:
   - для случайных подстановок F почти всегда $\text{AI}_{\text{comp}}(F) = n/2$ (максимально возможное) при чётном n и $\text{AI}_{\text{comp}}(F) = (n-1)/2$ (на 1 меньше максимально возможного) при нечётном n; это согласуется с результатами [15] для уравновешенных булевых функций f: для чётного n вероятность того, что минимальная среди степеней аннигиляторов множества M_f^1 равна $n/2$, близка к 1; для нечётного n эта степень с бóльшей вероятностью (0,711 против 0,289) равна $(n-1)/2$, чем $(n+1)/2$ (максимально возможной);

— если $F \in \mathcal{K}_n$, то $\mathrm{AI}_{\mathrm{comp}}(F) = 2$ (доказано в [14]); отсюда следует, что $\mathrm{AI}_{\mathrm{comp}}(F) \leqslant 2$ для $F \in \mathcal{S}_{n,k}$, поскольку добавление фиктивных переменных не влияет на алгебраическую иммунность функции. В экспериментах получено: $\mathrm{AI}_{\mathrm{comp}}(F) = 1$ для функций $F \in \mathcal{S}_{n,k}$ всегда при $k = 3$ и иногда при $k \geqslant 4$; последнее — чем больше n и k, тем реже (почти всегда $\mathrm{AI}_{\mathrm{comp}}(F) = 2$).

4) Для показателя дифференциальной равномерности:
   — при $n \geqslant m$, как правило, $\delta_F = 2^n$;
   — при фиксированном n с ростом m значение δ_F убывает;
   — если $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ — подстановка, то её дифференциальная равномерность в среднем много меньше, чем дифференциальная равномерность случайной такой функции (без условия обратимости).

Эксперименты на функциях из класса $\mathcal{K}_n$ позволили сформулировать некоторые утверждения об их дифференциальной равномерности.

3. Свойство дифференциальной равномерности функций из классов $\mathcal{K}_n$ и $\mathcal{S}_{n,k}$

Для $j \in \{1, \ldots, n\}$ обозначим e_j булев вектор длины n с единственной 1 в j-й координате.

Утверждение 1. Пусть $F \in \mathcal{K}_n$, $a = \mathrm{e}_{i_1} \oplus \mathrm{e}_{i_2} \oplus \ldots \oplus \mathrm{e}_{i_k}$, $j \in \{i_1, \ldots, i_k\}$. Тогда

$$\sum_{b=(b_1 \ldots b_n) \in \mathbb{F}_2^n : b_j = 0} \delta_F(a, b) = \begin{cases} 4, & k > 1, \\ 0, & k = 1. \end{cases}$$

Доказательство. Запишем:

$$\sum_{b=(b_1 \ldots b_n) \in \mathbb{F}_2^n : b_j = 0} \delta_F(a, b) = \sum_{b=(b_1 \ldots b_n) \in \mathbb{F}_2^n : b_j = 0} |\{x \in \mathbb{F}_2^n : F(x) \oplus F(x \oplus a) = b\}| =$$
$$= |\{x \in \mathbb{F}_2^n : f_j(x) = f_j(x \oplus a)\}|.$$

Заметим, что x и $x \oplus a$ различаются в j-й координате. Тогда из формулы (3) следует, что $f_j(x) = f_j(x \oplus a)$, если и только если ровно один из векторов x и $x \oplus a$ принадлежит M_j, где $M_j = \{c, c \oplus \mathrm{e}_j\}$ для некоторого $c \in \mathbb{F}_2^n$. Если $k = 1$ и, следовательно, $a = \mathrm{e}_j$, то x и $x \oplus a$ одновременно либо принадлежат, либо не принадлежат M_j. При $k > 1$ условие выполняется для $x \in X = \{c, c \oplus \mathrm{e}_j, c \oplus a, c \oplus \mathrm{e}_j \oplus a\}$; ввиду того, что вес вектора a больше 1, все векторы в X различны. ■

Утверждение 2. Пусть $F \in \mathcal{K}_n$, $a = \mathrm{e}_{i_1} \oplus \mathrm{e}_{i_2} \oplus \ldots \oplus \mathrm{e}_{i_k}$, $j \notin \{i_1, \ldots, i_k\}$. Тогда

$$\sum_{b=(b_1 \ldots b_n) \in \mathbb{F}_2^n : b_j = 1} \delta_F(a, b) = 4.$$

Доказательство. Аналогично доказательству утверждения 1 получим

$$\sum_{b=(b_1 \ldots b_n) \in \mathbb{F}_2^n : b_j = 1} \delta_F(a, b) = |\{x \in \mathbb{F}_2^n : f_j(x) \neq f_j(x \oplus a)\}|.$$

Поскольку j-е координаты x и $x \oplus a$ совпадают, из формулы (3) следует, что $f_j(x) \neq$ $\neq f_j(x \oplus a)$, если и только если ровно один из векторов x и $x \oplus a$ принадлежит $M_j =$ $= \{c, c \oplus \mathrm{e}_j\}$. Это условие выполняется для $x \in X = \{c, c \oplus \mathrm{e}_j, c \oplus a, c \oplus \mathrm{e}_j \oplus a\}$; ввиду того, что $a \neq \mathrm{e}_j$, все векторы в X различны. ■

Из утверждений 1 и 2 следует, что $\delta_F(a,b) \leqslant 4$ для всех $F \in \mathcal{K}_n$ и $a \neq b$; ввиду чётности значений $\delta_F(a,b)$ это означает, что $\delta_F(a,b) \in \{0,2,4\}$.

Утверждение 3. Пусть $F \in \mathcal{K}_n$, $a = \mathrm{e}_{i_1} \oplus \mathrm{e}_{i_2} \oplus \ldots \oplus \mathrm{e}_{i_k}$, $b = (b_1 \ldots b_n) \in \mathbb{F}_2^n$ и $b_j = 0$ для всех $j \in \{i_1, \ldots, i_k\}$. Тогда $\delta_F(a,b) = 0$.

Доказательство. Обозначим $X = \{x \in \mathbb{F}_2^n : \forall\, j \in \{i_1, \ldots, i_k\}(f_j(x) = f_j(x \oplus a))\}$. Тогда для a и b в условии утверждения $\delta_F(a,b) = |X|$ и для $x \in X$ должно выполняться условие: ровно один из векторов x и $x \oplus a$ принадлежит M_j, $j = i_1, \ldots, i_k$. Поскольку множества M_j не пересекаются, это условие не может выполняться при $k > 2$; случай $k = 1$ рассмотрен в утверждении 1. Осталось рассмотреть случай $k = 2$.

Пусть $a = \mathrm{e}_{i_1} \oplus \mathrm{e}_{i_2}$, $M_{i_1} = \{c, c \oplus \mathrm{e}_{i_1}\}$, $M_{i_2} = \{d, d \oplus \mathrm{e}_{i_2}\}$ и без ограничения общности $x \in M_{i_1}$, $x \oplus a \in M_{i_2}$. Если $x = c$, то $x \oplus a = x \oplus \mathrm{e}_{i_1} \oplus \mathrm{e}_{i_2}$; при $x \oplus a = d$ получаем $d \oplus \mathrm{e}_{i_2} = c \oplus \mathrm{e}_{i_1} \in M_{i_1}$, а если $x \oplus a = d \oplus \mathrm{e}_{i_2}$, то $d = c \oplus \mathrm{e}_{i_1} \in M_{i_1}$; оба вывода противоречат тому, что $M_{i_1} \cap M_{i_2} = \varnothing$. Случай $x = c \oplus \mathrm{e}_{i_1}$ рассматривается аналогично. ■

Утверждение 4. Пусть $F \in \mathcal{K}_n$, $a \in \mathbb{F}_2^n$. Тогда $\delta_F(a,a) \geqslant 2^n - 4n$, если вес a больше 1, и $\delta_F(a,a) \geqslant 2^n - 4(n-1)$ иначе.

Доказательство. Из того, что $\sum\limits_{b \in \mathbb{F}_2^n} \delta_F(a,b) = 2^n$, следует

$$\delta_F(a,a) = 2^n - \sum_{b \neq a} \delta_F(a,b). \tag{4}$$

Пусть $a = \mathrm{e}_{i_1} \oplus \mathrm{e}_{i_2} \oplus \ldots \oplus \mathrm{e}_{i_k}$. Запишем:

$$\sum_{b \neq a} \delta_F(a,b) \leqslant \sum_{j=1}^{n} \sum_{a,b \in \mathbb{F}_2^n : b_j \neq a_j} \delta_F(a,b) = \underbrace{\sum_{j \in \{i_1,\ldots,i_k\}} \sum_{b \in \mathbb{F}_2^n : b_j = 0} \delta_F(a,b)}_{S_1} + \underbrace{\sum_{j \notin \{i_1,\ldots,i_k\}} \sum_{b \in \mathbb{F}_2^n : b_j = 1} \delta_F(a,b)}_{S_2}.$$

Из утверждения 1 следует, что

$$S_1 = \begin{cases} 4k, & \text{если } k > 1, \\ 0, & \text{если } k = 1, \end{cases}$$

из утверждения 2 получаем $S_2 = 4(n-k)$. Тогда

$$\sum_{b \neq a} \delta_F(a,b) \leqslant \begin{cases} 4n, & \text{если } k > 1, \\ 4(n-1), & \text{если } k = 1. \end{cases} \tag{5}$$

Из (4) и (5) следует справедливость утверждения 4. ■

Утверждение 5. Для любой функции $F \in \mathcal{K}_n$ имеет место

$$2^n - 4(n-1) \leqslant \delta_F \leqslant 2^n - 4.$$

Доказательство. Нижняя граница следует непосредственно из утверждения 4. Из утверждений 1 и 2 получаем, что $\sum\limits_{b \neq a} \delta_F(a,b) \geqslant 4$ для любого ненулевого $a \in \mathbb{F}_2^n$, а значит, $\delta_F(a,a) \leqslant 2^n - 4$. Тогда из того, что $\delta_F(a,b) \leqslant 4$ для любых $a \neq b$ и неравенства $4 \leqslant 2^n - 4$, верного при всех $n \geqslant 3$, следует $\delta_F \leqslant 2^n - 4$. ■

Утверждение 6. Пусть $G = (g_1 \ldots g_n) \in \mathcal{S}_{n,k}$ и $k < n$. Тогда $\delta_G = 2^n$.

Доказательство. По построению, переменная x_n является линейной для функции g_n и фиктивной для остальных координат. Тогда $G(x) \oplus G(x \oplus \mathrm{e}_n) = \mathrm{e}_n$ для всех $x \in \mathbb{F}_2^n$, т. е. $\delta_G(\mathrm{e}_n, \mathrm{e}_n) = 2^n$, следовательно, $\delta_G = 2^n$. ■

Заключение

Исследования показали, что функции из классов $\mathcal{K}_n$ и $\mathcal{S}_{n,k}$ являются криптографически слабыми: их нелинейность и компонентная алгебраическая иммунность малы, а показатель дифференциальной равномерности близок (или равен — для класса $\mathcal{S}_{n,k}$) к максимальному (т. е. худшему) значению 2^n. К достоинствам таких функций можно отнести возможность управлять количеством существенных переменных у их координат; для функций класса $\mathcal{K}_n$ дополнительно — отсутствие линейных и фиктивных переменных у компонент, не равных координатам, и высокая степень $(n-1)$ — максимально возможная для подстановок. Направления дальнейших исследований:

1) изучение применимости атак, эксплуатирующих отмеченные слабости функций, к криптосистемам [2 – 4], как следствие — оценка стойкости этих криптосистем;
2) изучение композиций функций из классов $\mathcal{K}_n$ и $\mathcal{S}_{n,k}$ между собой и с другими функциями с целью нивелирования их недостатков при сохранении положительных свойств;
3) разработка новых алгоритмов генерации обратимых векторных булевых функций с «хорошими» криптографическими характеристиками.

Авторы благодарят Г. П. Агибалова за внимание к работе и ценные замечания.

ЛИТЕРАТУРА

1. *Ding J. and Yang B. Y.* Multivariate public key cryptography // Post-Quantum Cryptography / eds. D. J. Bernstein, J. Buchmann, and E. Dahmen. Berlin; Heidelberg: Springer, 2009. P. 193–241.
2. *Agibalov G. P.* Substitution block ciphers with functional keys // Прикладная дискретная математика. 2017. № 38. С. 57–65.
3. *Agibalov G. P. and Pankratova I. A.* Asymmetric cryptosystems on Boolean functions // Прикладная дискретная математика. 2018. № 40. С. 23–33.
4. *Agibalov G. P.* ElGamal cryptosystems on Boolean functions // Прикладная дискретная математика. 2018. № 42. С. 57–65.
5. *Carlet C.* Vectorial Boolean Functions for Cryptography. Cambridge: Cambridge University Press, 2010. 93 p.
6. *Canteaut A.* Lecture Notes on Cryptographic Boolean Functions. Paris: Inria, 2016. 48 p.
7. *Nyberg K.* Differentially uniform mappings for cryptography // LNCS. 1994. V. 765. P. 55–64.
8. *Логачев О. А., Сальников А. А., Ященко В. В.* Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2004. 472 с.
9. *Агибалов Г. П.* Элементы теории дифференциального криптоанализа итеративных блочных шифров с аддитивным раундовым ключом // Прикладная дискретная математика. 2008. № 1 (1). С. 34–42.
10. *Meier W., Pasalic E., and Carlet C.* Algebraic attacks and decomposition of Boolean functions // LNCS. 2004. V. 3027. P. 474–491.
11. *Агибалов Г. П., Липский В. Б., Панкратова И. А.* О криптографическом расширении и его реализации для русского языка программирования // Прикладная дискретная математика. 2013. № 3(21). С. 93–104.
12. *Pankratova I. A.* Construction of invertible vectorial Boolean functions with coordinates depending on given number of variables // Материалы Междунар. науч. конгресса по информатике: Информационные системы и технологии. Республика Беларусь, Минск, 24–27 окт. 2016. Минск: БГУ, 2016. С. 519–521.

13. *Панкратова И. А.* Об обратимости векторных булевых функций // Прикладная дискретная математика. Приложение. 2015. № 8. С. 35–37.
14. *Панкратова И. А.* Свойства компонент некоторых классов векторных булевых функций // Прикладная дискретная математика. 2019. № 44. С. 5–11.
15. *Canteaut A.* Open problems related to algebraic attacks on stream ciphers // Proc. WCC'2005, Bergen, Norway, March 14–18, 2005. P. 120–134.

REFERENCES

1. *Ding J. and Yang B. Y.* Multivariate public key cryptography. Post-Quantum Cryptography (eds. D. J. Bernstein, J. Buchmann, and E. Dahmen). Berlin; Heidelberg, Springer, 2009, pp. 193–241.
2. *Agibalov G. P.* Substitution block ciphers with functional keys. Prikladnaya Diskretnaya Matematika, 2017, no. 38, pp. 57–65.
3. *Agibalov G. P. and Pankratova I. A.* Asymmetric cryptosystems on Boolean functions. Prikladnaya Diskretnaya Matematika, 2018, no. 40, pp. 23–33.
4. *Agibalov G. P.* ElGamal cryptosystems on Boolean functions. Prikladnaya Diskretnaya Matematika, 2018, no. 42, pp. 57–65.
5. *Carlet C.* Vectorial Boolean Functions for Cryptography. Cambridge: Cambridge University Press, 2010. 93 p.
6. *Canteaut A.* Lecture Notes on Cryptographic Boolean Functions. Paris, Inria, 2016. 48 p.
7. *Nyberg K.* Differentially uniform mappings for cryptography. LNCS, 1994, vol. 765, pp. 55–64.
8. *Logachev O. A., Sal'nikov A. A., and Yashchenko V. V.* Bulevy funktsii v teorii kodirovaniya i kriptologii [Boolean Functions in Coding Theory and Cryptology]. Moscow, MCCME Publ., 2004, 472 p. (in Russian)
9. *Agibalov G. P.* Elementy teorii differentsial'nogo kriptoanaliza iterativnykh blochnykh shifrov s additivnym raundovym klyuchom [Some theoretical aspects of differential cryptanalysis of the iterated block ciphers with additive round key]. Prikladnaya Diskretnaya Matematika, 2008, no. 1 (1), pp. 34–42. (in Russian)
10. *Meier W., Pasalic E., and Carlet C.* Algebraic attacks and decomposition of Boolean functions. LNCS, 2004, vol. 3027, pp. 474–491.
11. *Agibalov G. P., Lipskiy V. B., and Pankratova I. A.* O kriptograficheskom rasshirenii i ego realizatsii dlya russkogo yazyka programmirovaniya [Cryptographic extension and its implementation for Russian programming language]. Prikladnaya Diskretnaya Matematika, 2013, no. 3(21), pp. 93–104. (in Russian)
12. *Pankratova I. A.* Construction of invertible vectorial Boolean functions with coordinates depending on given number of variables. Proc. CSIST'16, Minsk, BSU Publ., 2016, pp. 519–521.
13. *Pankratova I. A.* Ob obratimosti vektornykh bulevykh funktsiy [On the invertibility of vector Boolean functions]. Prikladnaya Diskretnaya Matematika. Prilozhenie, 2015, no. 8, pp. 35–37. (in Russian)
14. *Pankratova I. A.* Svoystva komponent nekotorykh klassov vektornykh bulevykh funktsiy [Properties of components for some classes of vectorial Boolean functions]. Prikladnaya Diskretnaya Matematika, 2019, no. 44, pp. 5–11. (in Russian)
15. *Canteaut A.* Open problems related to algebraic attacks on stream ciphers. Proc. WCC'2005, Bergen, Norway, March 14–18, 2005, pp. 120–134.

УДК 519.7

ВЫЧИСЛЕНИЕ ДЕТЕРМИНАНТА И ПРОИЗВЕДЕНИЯ МАТРИЦ В СТРУКТУРЕ КЛЕТОЧНОГО АВТОМАТА

В. С. Кожевников*, И. В. Матюшкин**

**Московский физико-технический институт (Национальный исследовательский университет), г. Долгопрудный, Россия*
***Национальный исследовательский университет «Московский институт электронной техники», г. Москва, Россия*

Приведено формальное описание двумерных клеточных автоматов, реализующих умножение матриц и нахождение определителя. Все алгоритмы даны для границ с замыканием, шаблона окрестности Неймана и закрытых вычислений (т. е. данные не вводятся в процессе расчёта). Отдельно реализовано умножение матрицы на вектор-столбец. Для вычисления определителя предложены два алгоритма, основанные на методе Гаусса. Один имеет линейную сложность и использует управляющий флаг с пятью состояниями, однако применим не ко всякой матрице. Другой работает для любой матрицы, но имеет квадратичную сложность и его управляющий флаг принимает одиннадцать состояний.

Ключевые слова: *клеточный автомат, определитель, детерминант, умножение матриц, параллельные вычисления.*

DOI 10.17223/20710410/46/8

COMPUTATION OF A DETERMINANT AND A MATRIX PRODUCT IN CELLULAR AUTOMATA

V. S. Kozhevnikov*, I. V. Matyushkin**

**Moscow Institute of Physics and Technology, Dolgoprudny, Russia*
***National Research University of Electronic Technology, Moscow, Russia*

E-mail: vladislavkozhevnikov@gmail.com, imatyushkin@niime.ru

In the paper, possible implementations in cellular automata of matrix–vector multiplication, matrix multiplication, and determinant computation are described. The algorithms are formalised in terms of a cellular automaton model using a two-dimensional field with closed boundaries and von Neumann neighbourhood. The proposed algorithms are notable for isolated calculations (meaning that no data is entered during the computation process), which is a feature of a classical cellular automaton as opposed to a systolic array. Matrix multiplication is implemented for two square matrices as well as specifically for a matrix and a column vector, the latter implementation using less control flag states and therefore less additional memory. The matrix multiplication algorithm adapts Katona's scheme for matrix multiplication in a systolic array to an isolated cellular automaton. For calculation of a determinant, two algorithms based on Gaussian elimination are proposed. One has linear complexity and uses a control flag with only 5 states, being, however, inapplicable to an arbitrary matrix. The other one is modified to seek the largest leading element during row reduction, which makes its complexity quadratic and its control flags assume 11 states but allows the algorithm to be applied to an arbitrary matrix and be more numerically stable.

Keywords: *cellular automaton, determinant, matrix multiplication, parallel computing.*

Введение

Несмотря на то, что в настоящее время фокус исследований по массивному вычислительному параллелизму и нетрадиционным парадигмам компьютинга сместился с однородных вычислительных систем (ОВС), популярных в 60–80-х гг. XX века [1, 2], на нейронные сети и нейроморфные системы, исследование клеточно-автоматного (КА) параллелизма, взятого в классической парадигме вычислений, предполагающей явно заданный алгоритм решения задачи, по-прежнему представляет теоретический интерес. Данная работа продолжает цикл публикаций по клеточно-автоматным алгоритмам типовых матричных операций. Ранее были даны точные КА-формулировки, решающие очень простые задачи: унарной поэлементной операции [3], сортировки строк [4], отражения матрицы относительно центральной вертикали или горизонтали, а также главной диагонали, т. е. транспонирования [3]. В поисках более простого КА для задачи транспонирования мы нашли алгоритм, проходящий некоторые псевдослучайные пермутации матрицы [5], в том числе поворот на 90 и 180° (отражение от центра).

В традиционном курсе линейной алгебры перечисленные преобразования, не затрагивающие элементы матрицы, считаются самоочевидными, но для ОВС (или клеточно-автоматного вычислительного узла) они достаточно важны. Ещё большей важностью обладают операции, меняющие элементы матрицы, и прежде всего — произведение матриц, вычисление определителя и обратной матрицы. К ним условно причислим нахождение обычного произведения натуральных чисел по схеме Атрубина [4], где каждый сомножитель задается вектором его цифр. Решение первых двух задач средствами КА является целью настоящей работы. Способы параллельной реализации этих операций в многопроцессорной системе достаточно известны [6]. В гораздо меньшей степени исследовалось КА-распараллеливание, располагающее более скудными средствами (прежде всего, это запрет на «дальнодействие», т. е. удалённые ячейки памяти не могут обмениваться данными даже через центральный процессорный элемент). Впервые мы дадим точную КА-формулировку для вычисления определителя матрицы.

1. Обозначения и предварительные замечания

В работе используются следующие обозначения для элементов двумерного клеточного автомата, имеющего структуру квадратной решётки размера $n \times n$ на торе, т. е. с замкнутыми границами. Состояние клетки определяется конечным набором компонент $\left\langle a^{(1)}, \ldots, a^{(K)} \right\rangle$, каждая из которых принимает значения в своём множестве состояний $D^{(k)}, k \in \{1, \ldots, K\}$. Удобно неформально разбивать по семантике компоненты на три группы: данные, флаги управления, стек. Глобальную конфигурацию КА, взятую по отдельной компоненте, назовём *слоем*. Клетки автомата индексируются как матричные элементы, первый индекс является номером строки, а второй — столбца. Компоненты клетки снабжаются теми же индексами, что и данная клетка. То есть $a_{ij}^{(k)}$ — компонента $a^{(k)}$ клетки, расположенной в i-й строке и j-м столбце. Используется относительная индексация для обозначения соседних клеток. Если выбрана центральная клетка окрестности, то индексы у её компонент опускаются, пишется просто $a^{(k)}$. Компоненты соседей обозначаются стрелками, например, $a_{\rightarrow}^{(k)}$ отвечает ближайшей клетке справа, $a_{\uparrow}^{(k)}$ — клетке сверху по отношению к центральной. Локальная функ-

ция перехода (ЛФП) задаётся таблицей условий и соответствующих переходов. Мы не требуем, чтобы условия были взаимоисключающими, поэтому в процессе применения ЛФП последовательно проверяются все условия. Эти условия, перечисленные в соответствующей таблице, не обязательно исчерпывают все возможные конфигурации окрестности. Если окрестность клетки имеет конфигурацию, отсутствующую в таблице, то её состояние не меняется. Иногда такие (тождественные) переходы всё же включаются в таблицу и обозначаются словом idem. В приведённых алгоритмах ЛФП для граничных ячеек не требует отдельной спецификации.

Обычно в практике моделирования КА-расчёт прерывают либо внешним образом по достижении некого числа итераций или по какому-то условию, типичному для численных методов, либо внутренним образом при равенстве текущей и прошлой глобальных конфигураций. В теории алгоритмов для останова обязательно нужны синтаксические средства (стоп-символ для машины Тьюринга, заключительная формула подстановки в нормальных алгорифмах Маркова и т. п.). При реализации КА-алгоритмов имеются две альтернативы: останов по idem и останов по стоп-значению. Первая альтернатива означает, что в каждой ячейке КА действует ЛФП типа idem (обычно контролируемая стоп-значением «заморозки» одного из флагов); она довольно естественна, но неудобна при техническом исполнении ОВС. Вторая альтернатива предполагает, что одна, любая или выделенная, ячейка КА приходит в стоп-состояние. Она предполагает другое написание ЛФП, чем в первом случае, и удобна с технической точки зрения. В основном мы будем использовать останов по idem.

Для нас принципиально разделять вычисления на закрытые и открытые. Вычисление назовём закрытым, если всё необходимое для него имеется в вычислителе до начала вычисления, а его результаты выводятся вовне после завершения. Иными словами, КА как математический объект не имеет средств ввода/вывода. Обычно классические КА в теории предстают закрытыми объектами, хотя открытым КА посвящена отдельная глава в классической работе [7]. Будем считать, что КА сконфигурирован до расчёта, т. е. вычисления закрытые. С точки зрения ОВС более удобны открытые, что характерно для схем Атрубина и Катоны. Здесь данные вводятся в процессе исполнения алгоритма. Несмотря на то, что мы в значительной мере используем идеи подобных алгоритмов, новизна работы состоит в их адаптации к закрытым вычислениям.

2. Умножение матриц

2.1. Традиционный параллелизм матричного умножения

В отличие от последовательных алгоритмов умножения матриц, которые используют всевозможные аналитические приёмы от несложных матричных преобразований [8] до инструментария тензорного анализа и теории групп [9] для сокращения (хотя бы асимптотического) числа необходимых арифметических операций, существующие параллельные алгоритмы работают практически по определению матричного произведения. Наиболее простая классификация условно разделяет их на два типа: ленточный и блочный.

В основе *ленточного* метода лежит разбиение матриц (или одной матрицы) на ленты. Под *лентой* понимается строка (столбец) или несколько смежных строк (столбцов). Пусть требуется найти произведение $AB = C$ квадратных матриц порядка n. Естественным представляется разбить матрицу A на строки, а B — на столбцы. Тогда n^2 процессоров могут вычислить элементы произведения C, параллельно перемножая соответствующие строки и столбцы. Если процессоров меньше, чем элементов C, то A

и B разбиваются на ленты, содержащие по несколько строк и столбцов, а процессоры находят уже не отдельные элементы C, а подматрицы.

Блочный метод, являющийся, пожалуй, наиболее распространённым, можно считать обобщением ленточного. Матрицы разбиваются на *блоки*, то есть подматрицы. Например, имея дело с квадратными матрицами A и B порядка n, кратного некоторому натуральному числу q (в частности, $q = n$), их можно разбить на q^2 равных по объёму блоков:

$$AB = \begin{pmatrix} A_{11} & \dots & A_{1q} \\ \vdots & \ddots & \vdots \\ A_{q1} & \cdots & A_{qq} \end{pmatrix} \begin{pmatrix} B_{11} & \dots & B_{1q} \\ \vdots & \ddots & \vdots \\ B_{q1} & \cdots & B_{qq} \end{pmatrix} = \begin{pmatrix} C_{11} & \dots & C_{1q} \\ \vdots & \ddots & \vdots \\ C_{q1} & \cdots & C_{qq} \end{pmatrix} = C.$$

Тогда полученные блочные матрицы перемножаются аналогично числовым:

$$C_{ij} = \sum_{k=1}^{q} A_{ik} B_{kj}.$$

Здесь C_{ij} — блоки произведения C, каждый из которых может быть вычислен отдельным процессором. В пределах одного процессора могут использоваться любые методы последовательного умножения матриц для более эффективного нахождения произведения блоков (например, алгоритм Штрассена [10]).

Из конкретных алгоритмов, реализующих блочный метод, широко известны схемы Фокса и Кэннона [6]. Они подразумевают использование сети процессоров с распределённой памятью, что близко к идее КА. Существуют алгоритмы, реализующие матричное умножение на систолических массивах процессоров [10, 11], а также его частный случай — умножение матрицы на вектор [12], которые тоже следует отнести скорее к блочным, чем к ленточным.

Что касается времени вычисления произведения, то для двумерной решётки процессоров, сообщающихся лишь с соседями в своей окрестности Неймана, минимально возможное время вычисления не менее $\Theta(n)$, где n — порядок перемножаемых квадратных матриц, вне зависимости от числа процессоров [13]. Имеется достаточное количество примеров алгоритмов с такой сложностью, перемножающих две [10, 11, 14] и даже три матрицы [15]. Для сетей с более сложной топологией и количеством процессоров, превышающим $\Theta(n^2)$, предложены алгоритмы, производящие умножение за время $\Theta(\log_2 n)$ [14]. Алгоритмы с меньшим временем исполнения нам не известны. КА накладывает жёсткое ограничение на структуру сети, предполагая наличие решётки.

2.2. Алгоритм 1. Умножение матрицы на столбец

Сначала рассмотрим задачу умножения матрицы на столбец. Выбранный способ в некотором смысле является ленточным, что отличает его от применяемого нами далее. Состояние ячейки КА является трёхкомпонентным (табл. 1), R — некоторое кольцо, например $\mathbb{R}$ или $\mathbb{C}$.

Таблица 1

Компоненты автомата, умножающего матрицу на столбец

Компонента	Множество значений	Назначение
a	Кольцо R	Хранение матрицы
b	Кольцо R	Хранение столбца
s	$\{0, 1, 2\}$	Управление

Запишем начальные условия:

1) $s_{1j} = 1$, $j \in \{1, \ldots, n-1\}$; остальные s_{ij} равны 0;
2) $b_{1j} = B_j$, $j \in \{1, \ldots, n\}$; остальные b_{ij} произвольные;
3) $\|a_{ij}\| = A$.

Здесь $B_{n\times 1}$ — столбец, на который умножается матрица $A_{m\times n}$, то есть матрица записывается в автомат естественным образом, а столбец — транспонированным в первую строку. На каждом шаге столбец B копируется вниз, пока не будет записан в каждой строке. Как только столбец достигает строки, в ней начинается процесс умножения и суммирования. Сумма вычисляется последовательно, начиная с конца строки, и в итоге записывается в её начало. Таким образом, ответ будет располагаться в первом столбце компоненты a. В табл. 2 представлена ЛФП, реализующая описанный алгоритм; условия, разделённые вертикальной чертой, задают конъюнкцию ($\wedge$), а горизонтальной — дизъюнкцию ($\vee$).

Таблица 2

ЛФП автомата, умножающего матрицу на столбец

№ п/п	Условие		Правило перехода
1	$s = 0$	$s_{\leftarrow} = 0$	$s := s_{\uparrow}$
		$s_{\leftarrow} = 1$	$s := 2$
	$(s = 1) \wedge (s_{\rightarrow} \neq 1)$		
2	$s = 0$		$b := b_{\uparrow}$
	$s = 1$	$s_{\rightarrow} = 0$	$a := a \cdot b + a_{\rightarrow} \cdot b_{\rightarrow}$
		$s_{\rightarrow} = 2$	$a := a \cdot b + a_{\rightarrow}$
3	$s = 2$		idem

В табл. 3, где иллюстрируется работа КА на примере умножения матрицы размера 4×5 на соответствующий столбец, t — номер шага, $t = 0$ — начальная конфигурация в алгоритме 1; в столбце, соответствующем флагу s, цветом выделены «замороженные» клетки; в столбце компоненты a различными цветами выделены клетки, содержащие частичную сумму (например, 12), полную сумму (например, 15) и окончательный результат (например, 19); в столбце компоненты b цветом выделены клетки, содержащие элементы столбца B, а клетки с незначащим содержимым заштрихованы.

Строки, состоящие из нулей, ещё не содержат столбца B. Как только в флаге s строки появляются единицы, в ней начинается умножение и суммирование. Значение $s = 2$ является остановочным, то есть при переходе в состояние с $s = 2$ клетка «замораживается» (обозначено словом idem в табл. 2). Автомат завершает работу на шаге $t = m+n-2$. Алгоритм работает лишь для $n \geqslant 2$, то есть столбец B должен содержать более одного элемента.

Таблица 3

Эволюция трёх компонент $\langle a, b, s\rangle$ КА при умножении матрицы размера 4×5

t	s					a					b				
0	1	1	1	1	0	5	1	3	1	0	4	2	3	0	2
	0	0	0	0	0	2	3	0	1	4	0	0	0	0	0
	0	0	0	0	0	-1	0	-1	2	-1	0	0	0	0	0
	0	0	0	0	0	0	3	1	-1	5	0	0	0	0	0
1	1	1	1	2	2	5	1	3	0	0	4	2	3	0	0
	1	1	1	1	0	2	3	0	1	4	4	2	3	0	2
	0	0	0	0	0	-1	0	-1	2	-1	0	0	0	0	0
	0	0	0	0	0	0	3	1	-1	5	0	0	0	0	0
2	1	1	2	2	2	5	1	9	0	0	4	2	3	0	0
	1	1	1	2	2	2	3	0	8	4	4	2	3	0	0
	1	1	1	1	0	-1	0	-1	2	-1	4	2	3	0	2
	0	0	0	0	0	0	3	1	-1	5	0	0	0	0	0
3	1	2	2	2	2	5	11	9	0	0	4	2	3	0	0
	1	1	2	2	2	2	3	8	8	4	4	2	3	0	0
	1	1	1	2	2	-1	0	-1	-2	-1	4	2	3	0	0
	1	1	1	1	0	0	3	1	-1	5	4	2	3	0	2
4	2	2	2	2	2	31	11	9	0	0	4	2	3	0	0
	1	2	2	2	2	2	14	8	8	4	4	2	3	0	0
	1	1	2	2	2	-1	0	-5	-2	-1	4	2	3	0	0
	1	1	1	2	2	0	3	1	10	5	4	2	3	0	0
5	2	2	2	2	2	31	11	9	0	0	4	2	3	0	0
	2	2	2	2	2	22	14	8	8	4	4	2	3	0	0
	1	2	2	2	2	-1	-5	-5	-2	-1	4	2	3	0	0
	1	1	2	2	2	0	3	13	10	5	4	2	3	0	0
6	2	2	2	2	2	31	11	9	0	0	4	2	3	0	0
	2	2	2	2	2	22	14	8	8	4	4	2	3	0	0
	2	2	2	2	2	-9	-5	-5	-2	-1	4	2	3	0	0
	1	2	2	2	2	0	19	13	10	5	4	2	3	0	0
7	2	2	2	2	2	31	11	9	0	0	4	2	3	0	0
	2	2	2	2	2	22	14	8	8	4	4	2	3	0	0
	2	2	2	2	2	-9	-5	-5	-2	-1	4	2	3	0	0
	2	2	2	2	2	19	19	13	10	5	4	2	3	0	0

2.3. Алгоритм 2. Умножение матриц в два этапа

Приведённый здесь алгоритм умножения квадратных матриц использует идею [11] (рис. 1). Элементы матриц A и B поступают в поле КА с дискретным запаздыванием (отображено сдвигом строк/столбцов) во время расчёта.

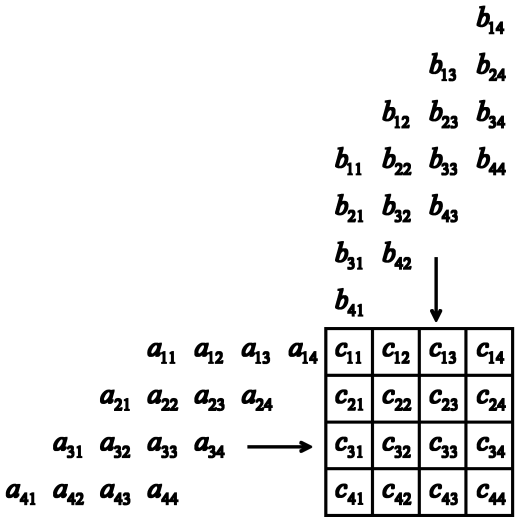


Рис. 1. Схема умножения Катоны

Двумерный клеточный автомат хранит элементы перемножаемых матриц A и B, а также их произведения C. Сначала происходит относительный сдвиг строк матрица A и столбцов матрицы B, а затем они (столбцы и строки) перемещаются так, что на каждом шаге в каждой клетке остаётся лишь перемножать содержащиеся в ней элементы A и B и складывать произведение с элементом C в той же клетке. Иными словами, производится поклеточное накопление суммы по формуле $c := c + a \cdot b$, перемещения же частичных сумм, как в алгоритме 1, не происходит, что является прин-

ципиальной особенностью алгоритма Катоны. В отличие от оригинального алгоритма Катоны, предлагаемый алгоритм ориентирован на закрытые вычисления. Структура данных КА показана в табл. 4.

Таблица 4

Компоненты автомата, перемножающего матрицы в два этапа

Компонента	Множество значений	Назначение
a	Кольцо R	Хранит первый множитель, матрицу A
b	Кольцо R	Хранит второй множитель, матрицу B
c	Кольцо R	Хранит результат произведения
s	$\{0,1\}$	Флаг синхронизации
v	$\{0,1\}$	Флаг вертикального перемещения
h	$\{0,1\}$	Флаг горизонтального перемещения

Этап 1. Смещение матричных элементов. Матрица A: строки с флагом $h = 1$ двигаются вправо. Флаг распространяется из первой строки вниз. Матрица B: столбцы с флагом $v = 1$ двигаются вниз. Флаг распространяется из первого столбца вправо. Соответствующая ЛФП представлена в табл. 5.

Этап 2. Перемножение. При указанных правилах перехода после смещения строки матрицы A и столбцы матрицы B будут продолжать двигаться, но уже синхронно, смещённые друг относительно друга. Если, начиная с шага $t = n$, добавить правило 3, приведённое в табл. 6, то автомат произведёт умножение за следующие n шагов (начиная с шага $t = n$).

Таблица 5

ЛФП на первом этапе

№ п/п	Условие	Правило перехода
1	$h_{\uparrow} = 1$	$h := 1$
	$h = 1$	$a := a_{\leftarrow}$
2	$v_{\leftarrow} = 1$	$v := 1$
	$v = 1$	$b := b_{\uparrow}$

Таблица 6

ЛФП для флага s

№ п/п	Условие	Правило перехода
3	$t \geqslant n$	$c := c + a \cdot b$
4	Всегда	$s := s_{\uparrow}$

Синхронизация. Автомат следует остановить через $2n - 1$ шагов с момента запуска. Останов на шаге $t = 2n - 1$ и изменение правил с шага $t = n$ происходит по значению. Для этой цели вводится флаг синхронизации $s \in \{0,1\}$, изначально равный 1 лишь в левой верхней клетке $(1,1)$ и двигающийся вниз. При достижении им левой нижней клетки $(n,1)$ добавляется правило 3 (табл. 6) для всех клеток одновременно, а при повторном её достижении автомат останавливается. Выпишем полностью начальные условия алгоритма 2:

$$\forall j \in \{1,\ldots,n\}\,(h_{1j} = 1) \ \wedge \ \forall i \neq 1 \forall j \in \{1,\ldots,n\}\,(h_{ij} = 0) \ \wedge \ \forall i \in \{1,\ldots,n\}\,(v_{i1} = 1),$$
$$\forall j \neq 1 \forall i \in \{1,\ldots,n\}\,(v_{ij} = 0) \ \wedge \ s_{11} = 1,$$
$$\forall (i,j) \neq (1,1)\,(s_{ij} = 0) \ \wedge \ \forall i,j \in \{1,\ldots,n\}\,(c_{ij} = 0).$$

2.4. Последовательное выполнение КА-алгоритмов и проблема синхронизации

На примере алгоритма 2 мы встретились с проблемой последовательного сочленения КА, когда логика решения задачи требует её декомпозиции на несколько этапов и соответственно на несколько простых КА со своими ЛФП. Для последовательных вычислений она почти незаметна, так как объединяются множества инструкций и данных. Имеет ли формализм КА внутренние средства реализации последовательного выполнения двух КА над общим полем данных?

Абстрактный КА с двумя компонентами можно представить четверкой $\Xi = \langle \mathrm{T}, S = A \times B, LTF \rangle$. Здесь T — топология поля и шаблона окрестности; S — множество состояний ячейки (для определённости состоящее из двух компонент); LTF — локальная функция перехода, т. е. $\begin{pmatrix} a^{t+1} \\ b^{t+1} \end{pmatrix} = \begin{pmatrix} f\left(a^t, b^t\right) \\ g\left(a^t, b^t\right) \end{pmatrix}$, где t — номер шага, $a \in A$, $b \in B$. Пусть есть два КА Ξ_1, Ξ_2 с общей топологией $\mathrm{T} \equiv \mathrm{T}_1 = \mathrm{T}_2$, перекрывающимися состояниями $S_1 = A \times B$, $S_2 = B \times C$ и различными функциями перехода $LTF_1 = \begin{pmatrix} f_1 \\ g_1 \end{pmatrix}, LTF_2 = \begin{pmatrix} g_2 \\ h_2 \end{pmatrix}$, которые должны функционировать последовательно, будучи связаны конечной и начальной конфигурациями для произвольной ячейки поля неким правилом переноса $\begin{pmatrix} b^{t_2=0} \\ c^{t_2=0} \end{pmatrix} = TR \begin{pmatrix} a^{t_1=t_F} \\ b^{t_1=t_F} \end{pmatrix}$ в момент времени t_F. Каждый автомат работает в своем времени — t_1 и t_2 соответственно. Требуется построить третий КА, эмулирующий их последовательную работу в едином времени t. Обозначим его как $\Xi = \Xi_1 \oplus \Xi_2$.

Естественно сразу взять $S = A \times B \times C$, а ЛФП представить в виде

$$LTF = \begin{cases} LTF_1, & t \leqslant t_F, \\ LTF_2, & t > t_F \end{cases}$$

или в покомпонентном виде

$$(a, b, c)^{t+1} = \begin{cases} \left(f_1\left(a^t, b^t\right), g_1\left(a^t, b^t\right), \mathrm{idem}\right), & t \leqslant t_F, \\ \left(\mathrm{idem}, g_2\left(b^t, c^t\right), f_1\left(b^t, c^t\right)\right), & t > t_F. \end{cases}$$

Однако такой вид недопустим, поскольку идеология КА запрещает при формализации ЛФП ссылаться на абсолютные величины — номер итерации, индексы ячейки в поле и т. п. Отдельная ячейка «не знает» о глобальных событиях наподобие ситуации останова. В алгоритме 2 нам пришлось ввести флаг синхронизации, чтобы реализовать условный останов по значению первого КА и воспользоваться им для останова второго КА. Даже в этой ситуации требуется внешнее вмешательство, чтобы синтезировать ЛФП с управлением по флагу, полагая

$$LTF = \begin{cases} LTF_1, & s = 0, \\ LTF_2, & s = 1. \end{cases}$$

Как синхронизировать ячейки КА по управляющему флагу, если он изменился только в одной ячейке? Проблема известна как задача синхронизации стрелков (Firing Squad Synchronization Problem, FSSP) [16]. Однако заметим, что при синтезе Ξ нам придётся идти на затраты памяти в виде дополнительных управлений U, и тогда $S = A \times B \times \times C \times U$, и на затраты времени на синхронизацию.

Если применить для алгоритма 2 результат, полученный в [17], как использующий наименьшее среди известных решений количество состояний, то флаг s должен будет принимать значения в множестве $\{\omega, \lambda, \alpha, \beta, \gamma\}$. Также потребуется новый флаг f, идентичный s, так что $U \leftrightarrow \binom{s}{f}$. Синхронизация производится независимо в каждой строке. Правила перехода для флагов s и f (будем называть их соответственно s-правилами и f-правилами) заимствованы из [17, рис. 15] с той разницей, что отсутствует фиктивное состояние, используемое для задания правил перехода граничных клеток, так как мы используем КА с замкнутыми границами. Отсутствует также состояние выстрела. Синхронизация считается выполненной, если все клетки одновременно переходят в состояние ω, причём до этого паттерн $\omega\omega\omega$ не встречается.

Правила перехода для флагов s и f не отличаются, однако начальные состояния должны быть подобраны так, чтобы флаг s производил синхронизацию за $n-1$ шагов, а f — за $2n-1$ шагов. Итак, в новом варианте автомата в правила 1, 2 и 3 добавляется условие синхронизации, правило 4 заменяется s-правилами и добавляются f-правила (табл. 7).

Таблица 7

ЛФП синхронизированного КА

№ п/п	Условие	Правило перехода
1	$h_{\uparrow} = 1$	$h := 1$
	$\left(\overline{f_{\leftarrow} f f_{\rightarrow}} \neq \omega\omega\omega\right) \wedge (h = 1)$	$a := a_{\leftarrow}$
2	$v_{\leftarrow} = 1$	$v := 1$
	$\left(\overline{f_{\leftarrow} f f_{\rightarrow}} \neq \omega\omega\omega\right) \wedge (v = 1)$	$b := b_{\uparrow}$
3	$\left(\overline{f_{\leftarrow} f f_{\rightarrow}} \neq \omega\omega\omega\right) \wedge \left(\overline{s_{\leftarrow} s s_{\rightarrow}} = \omega\omega\omega\right)$	$c := c + a \cdot b$
4	s-правила	
5	f-правила	

Для флагов h, v и c начальные условия остаются прежними. Основная сложность заключается в задании начальных условий для флагов синхронизации s и f.

На примере алгоритма 2 мы видим, что накладные затраты на последовательное соединение КА могут быть сравнимы с основными затратами и даже превосходить их.

2.5. Алгоритм 3. Эффективный алгоритм умножения матриц

Оказывается, что можно обойтись вовсе без синхронизации. Для этого умножение происходит не одновременно во всех клетках и лишь на втором этапе, а в определённой последовательности, начиная с первого же шага. При этом используется всего один вспомогательный флаг $s \in \{0, r, d, 1\}$, имеющий четыре состояния. Границы двух этапов теперь несколько меняются и вообще становятся условными.

На первом этапе алгоритма, занимающем $n-2$ шага, процесс изменения флага s эквивалентен работе введённых ранее флагов h и v, если считать $s = 0 \Leftrightarrow h = v = 0$, $s = 1 \Leftrightarrow h = v = 1$, $s = r \Leftrightarrow (h = 0) \wedge (v = 1)$ и $s = d \Leftrightarrow (h = 1) \wedge (v = 0)$. На следующем этапе переходы усложняются. На первом этапе умножение производится лишь в клетках с $s = 1$, на втором — во всех клетках с $s \neq 0$. При $s = 0$ клетка находится в состоянии покоя, то есть она не производит никаких операций над матричными элементами. Подчеркнём, что всё это достигается применением одной и той же ЛФП (табл. 8) на обоих условных этапах.

Таблица 8

ЛФП алгоритма 3 умножения матриц

№ п/п	Условие			Правило перехода
1	$s=0$	$s_\uparrow = r$	$s_\leftarrow = 0$	$s := r$
			$s_\leftarrow = d$	$s := 1$
		$(s_\uparrow = 0) \wedge (s_\leftarrow = d)$		$s := d$
	$s=r$	$(s_\leftarrow = r) \wedge (s_\uparrow = 1)$		$s := 1$
		$s_\leftarrow = 1$	$(s_\rightarrow = r) \wedge (s_\uparrow \neq 1)$	
			$(s_\rightarrow = 1) \vee (s_\uparrow = 1)$	$s := d$
		$s_\leftarrow = 0$		$s := 0$
	$s=d$	$(s_\uparrow = d) \wedge (s_\leftarrow = 1)$		$s := 1$
		$s_\uparrow = 1$	$(s_\downarrow = d) \wedge (s_\leftarrow \neq 1)$	
			$(s_\downarrow = 1) \vee (s_\leftarrow = 1)$	$s := r$
		$s_\uparrow = 0$		$s := 0$
	$s=1$	$(s_\uparrow = r) \wedge (s_\leftarrow = d)$		$s := 0$
		$(s_\uparrow = 0) \vee (s_\leftarrow = 0)$		
		$(s_\uparrow = 1) \wedge (s_\leftarrow = d)$		$s := d$
		$(s_\uparrow = r) \wedge (s_\leftarrow = 1)$		$s := r$
2	$(s = r) \vee (s = 1)$			$a := a_\leftarrow$
	$(s = d) \wedge (s_\leftarrow = 1)$			
3	$(s = d) \vee (s = 1)$			$b := b_\uparrow$
	$(s = r) \wedge (s_\uparrow = 1)$			
4	$s = 1$			$c := c + a \cdot b$
	$(s = r) \wedge (s_\uparrow = 1)$			
	$(s = d) \wedge (s_\leftarrow = 1)$			
5	$s = s_\rightarrow = s_\downarrow = s_\leftarrow = s_\uparrow = 0$			idem

Запишем начальные условия:

$$s_{11} = 1 \ \wedge \ \forall j \in \{2, \ldots, n\}\,(s_{1j} = r) \ \wedge \ \forall i \in \{2, \ldots, n\}\,(s_{i1} = d);$$

остальные $s_{ij} = 0$;

$\|a_{ij}\| = A$, $\|b_{ij}\| = B$ — первый и второй сомножители;

c_{ij} произвольные.

В табл. 9, где представлена динамика КА для $n = 5$, t — номер шага, $t = 0$ — начальная конфигурация, в столбце флага s цветом выделены «замороженные» клетки (например, 0), а в столбце компоненты c — окончательный результат (например, 5).

В алгоритме 3 останов происходит по idem, которому соответствует нулевая конфигурация окрестности (фон Неймана), то есть $s = 0$ в самой клетке и у всех её соседей. Автомат завершает работу за $(2n - 1)$ шагов.

Таблица 9

Эволюция компонент $\langle a, b, c, s\rangle$ КА, работающего по алгоритму 3 при умножении матриц порядка $n = 5$

Этап 1					Этап 2				
t	s	a	b	c	t	s	a	b	c
0	1 *r r r r*	3 1 1 0 -1	1 -2 -1 2 -2	0 0 0 0 0	4	1 1 1 1 *d*	1 1 0 -1 3	2 -1 -1 -1 -2	3 -5 -6 6 0
	d 0 0 0 0	-2 0 -1 -1 2	2 -2 0 -2 -1	0 0 0 0 0		1 1 1 1 *d*	-1 -1 2 -2 0	-1 4 3 2 -1	-7 2 2 0 0
	d 0 0 0 0	3 1 3 1 0	-1 -1 2 1 1	0 0 0 0 0		1 1 1 1 *d*	1 0 3 1 3	3 -1 -1 -2 1	3 -8 6 3 0
	d 0 0 0 0	0 2 4 3 0	3 4 -1 -1 -1	0 0 0 0 0		1 1 1 1 *d*	0 0 2 4 3	-1 -2 0 1 -1	0 -4 8 -3 0
	d 0 0 0 0	-1 1 -1 -2 1	-1 -1 3 -1 1	0 0 0 0 0		*r r r r* 1	-1 1 -1 -2 1	1 -2 2 -1 1	0 0 0 0 0
1	1 1 *r r r*	-1 3 1 1 0	-1 -2 -1 2 -2	3 0 0 0 0	5	0 *r r r r*	3 1 1 0 -1	1 -2 2 -1 1	5 -6 -6 7 -6
	1 1 *r r r*	-2 0 -1 -1 2	1 -2 0 -2 -1	0 0 0 0 0		*d* 1 1 1 1	0 -1 -1 2 -2	2 -1 -1 -1 -2	-6 -2 8 -4 0
	d d 0 0 0	3 1 3 1 0	2 -1 2 1 1	0 0 0 0 0		*d* 1 1 1 1	3 1 0 3 1	-1 4 3 2 -1	6 -8 3 1 3
	d d 0 0 0	0 2 4 3 0	-1 4 -1 -1 -1	0 0 0 0 0		*d* 1 1 1 1	3 0 0 2 4	3 -1 -1 -2 1	0 -4 8 1 -3
	d d 0 0 0	-1 1 -1 -2 1	3 -1 3 -1 1	0 0 0 0 0		*d* 1 1 1 1	1 -1 1 -1 -2	-1 -2 0 1 -1	-1 -2 -2 2 1
2	1 1 1 *r r*	0 -1 3 1 1	3 -1 -1 2 -2	4 -6 0 0 0	6	0 0 1 1 1	3 3 1 1 0	1 -2 0 1 -1	5 -8 -4 7 -7
	1 1 1 *r r*	2 -2 0 -1 -1	-1 -2 0 -2 -1	-2 0 0 0 0		0 0 *r r r*	-2 0 -1 -1 2	1 -2 2 -1 1	-6 -1 9 -6 4
	1 1 1 *r r*	3 1 3 1 0	1 -2 2 1 1	0 0 0 0 0		1 *d* 1 1 1	1 3 1 0 3	2 -1 -1 -1 -2	3 -4 3 7 2
	d d d 0 0	0 2 4 3 0	2 -1 -1 -1 -1	0 0 0 0 0		1 *d* 1 1 1	4 3 0 0 2	-1 4 3 2 -1	9 -4 8 -3 1
	d d d 0 0	-1 1 -1 -2 1	-1 4 3 -1 1	0 0 0 0 0		1 *d* 1 1 1	-2 1 -1 1 -1	3 -1 -1 -2 1	-2 0 -2 1 3
3	1 1 1 1 *r*	1 0 -1 3 1	-1 4 3 2 -2	4 -5 -3 0 0	7	0 0 0 1 1	3 3 3 1 1	1 -2 -1 -2 1	5 -8 -4 8 -7
	1 1 1 1 *r*	-1 2 -2 0 -1	3 -1 -1 -2 -1	-4 4 0 0 0		0 0 0 1 1	-2 0 0 -1 -1	1 -2 0 1 -1	-6 -1 7 -5 6
	1 1 1 1 *r*	0 3 1 3 1	-1 -2 0 1 1	3 -2 6 0 0		0 0 0 *r r*	3 1 3 1 0	1 -2 2 -1 1	5 -7 2 7 -4
	1 1 1 1 *r*	0 2 4 3 0	1 -2 2 -1 -1	0 0 0 0 0		1 1 *d* 1 1	2 4 3 0 0	2 -1 -1 -1 -2	5 8 8 -3 -1
	d d d d 0	-1 1 -1 -2 1	2 -1 -1 -1 1	0 0 0 0 0		1 1 *d* 1 1	-1 -2 1 -1 1	-1 4 3 2 -1	-8 -1 -1 -1 2
					8	0 0 0 0 1	3 3 3 3 1	1 -2 -1 2 -1	5 -8 -4 6 -6
						0 0 0 0 1	-2 0 0 0 -1	1 -2 0 -2 1	-6 -1 7 -6 7
						0 0 0 0 1	3 1 3 3 1	1 -2 2 1 -1	5 -7 2 6 -4
						0 0 0 0 *r*	0 2 4 3 0	1 -2 2 -1 1	9 4 5 -3 -1
						1 1 1 *d* 1	1 -1 -2 1 -1	2 -1 -1 -1 -2	-7 -9 2 -3 1
					9	0 0 0 0 0	3 3 3 3 3	1 -2 -1 2 -2	5 -8 -4 6 -7
						0 0 0 0 0	-2 0 0 0 0	1 -2 0 -2 -1	-6 -1 7 -6 6
						0 0 0 0 0	3 1 3 3 3	1 -2 2 1 1	5 -7 2 6 -5
						0 0 0 0 0	0 2 4 3 3	1 -2 2 -1 -1	9 4 5 -3 -1
						0 0 0 0 0	-1 1 -1 -2 1	1 -2 2 -1 1	-5 -8 4 -4 3

3. Вычисление определителя матрицы

Как известно, определителем квадратной матрицы $\|a_{ij}\|_{i,j=1}^{n}$ называется сумма следующего вида:

$$\det\|a_{ij}\| = \sum_{i_1,\ldots,i_n=1}^{n} \varepsilon_{i_1,\ldots,i_n} a_{1i_1} \ldots a_{ni_n},$$

где $\varepsilon_{i_1,\ldots,i_n}$ — символ Леви-Чивиты.

Приведённая формула известна как формула полного разложения [18], которая содержит $n!$ однородных произведений. Существуют и другие равносильные определения, например основанное на разложении Лапласа:

$$\det\|a_{ij}\| = \sum_{j=1}^{n} a_{ij} A_{ij},$$

где A_{ij} — алгебраическое дополнение элемента a_{ij}, i — произвольный номер от 1 до n. Эта формула имеет рекурсивный характер, отсылая к определителям меньшего порядка и также предполагая чередование знаков. Наличие рекурсии ставит под сомнение возможность КА-реализации.

Перечислим некоторые элементарные свойства определителя, которые потребуются в дальнейшем:

— при перестановке двух строк (столбцов) матрицы её определитель меняет знак;
— при прибавлении к строке (столбцу) матрицы линейной комбинации остальных строк (столбцов) её определитель не меняется.

3.1. Вычислительный параллелизм для нахождения детерминанта

Наиболее простым является вычисление по определению, например по обобщённому правилу Саррюса, основанному на формуле полного разложения [19]. Разумеется, такой подход далёк от оптимального, поскольку имеет факториальную вычислительную сложность.

Также известен своей простотой метод Гаусса [18], использующий элементарные преобразования матрицы для приведения её к треугольному виду. Сложность его выполнения на одиночном процессоре составляет $\Theta(n^3)$ (n — порядок матрицы) [20]. Однако помимо того, что он плохо поддаётся распараллеливанию, он включает в себя деление, что делает его неприменимым над произвольными кольцами.

Алгоритмы, ориентированные на параллельные вычисления, способны находить детерминант за полилогарифмическое время [20]. При нижней оценке вычислительной сложности $\Theta(\log_2 n)$ существуют алгоритмы со сложностью $\Theta\left((\log_2 n)^2\right)$ [21]. При этом многие алгоритмы не используют деления, что значительно расширяет круг их применения. Так, алгоритм, описанный в [22], применим к матрицам над произвольным коммутативным кольцом с единицей. Тем не менее подобные алгоритмы требуют полиномиального количества процессоров $p = p(n)$. В различных работах приводятся следующие оценки: $p = \mathrm{O}(n^6)$ [20], $p = \mathrm{O}(n^4)$ [21], $p = \mathrm{O}(n^{3{,}496})$ [22]. Ясно, что использование такого количества вычислительных элементов непрактично. Кроме того, такие алгоритмы представляют процессоры нелокально связанными, что противоречит классическому определению КА.

Таким образом, существующие в теории быстрые алгоритмы не замещают более медленные аналоги с полиномиальной сложностью. В частности, упомянутый метод Гаусса широко применяется в современных вычислительных системах [23 – 26].

3.2. Алгоритм 4. Простой алгоритм вычисления определителя

Описанный ниже двумерный клеточный автомат (табл. 10) вычисляет определитель матрицы за линейное время (относительно порядка матрицы $n > 1$) методом Гаусса. Предполагается, что в процессе не возникает строк с нулевым ведущим элементом. Здесь F — некоторое поле, например $\mathbb{R}$ или $\mathbb{C}$.

Таблица 10

Компоненты клетки автомата, вычисляющего определитель методом Гаусса

Компонента	Множество значений	Назначение
s	$\{0, 1, 2, 3, 4\}$	Управляющий флаг
a	Поле F	Хранение матрицы
b	Поле F	Копирование элементов первой строки вниз
c	Поле F	Копирование элементов первого столбца вправо

Запишем начальные условия:

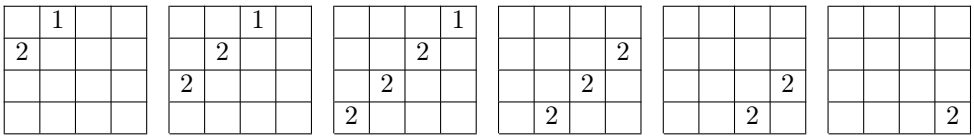
$$s_{11} = 1;\ \text{остальные } s_{ij} = 0,$$
$$\|a_{ij}\| = A \text{ — рассматриваемая матрица,}$$
b и c произвольные.

Флаг s определяет все переходы: $s = 0$ отвечает состоянию покоя (начальному состоянию большинства клеток), $s = 3$ — состоянию «заморозки», $s = 1$ обозначает клетку строки, вычитаемой из нижних строк (с множителем согласно методу Гаусса), $s = 2$ — клетку, переносящую элемент вычитаемой строки, делённый на ведущий элемент (хранится в компоненте b), и первый элемент данной строки (хранится в компоненте c), $s = 4$ — вспомогательное значение, позволяющее определить начало очередного этапа метода Гаусса.

Под *этапом метода Гаусса* понимается вычитание строки, умноженной на соответствующие коэффициенты, из нижележащих строк, в результате которого элементы, находящиеся под ведущим (первым ненулевым) элементом вычитаемой строки, обнуляются. Один этап осуществляется последовательностью конфигураций флага s, приведённой в табл. 11.

Таблица 11

Последовательность конфигураций флага s, соответствующая одному этапу (первому) метода Гаусса для $n = 4$



Однако не имеет смысла ожидать завершения одного этапа для начала следующего, поэтому алгоритм работает по принципу конвейера. Запуск каждого следующего этапа происходит на третий шаг после начала предыдущего. Полная последовательность конфигураций до остановки всех компонент автомата для $n = 5$ представлена в табл. 12. В ней цветом выделены клетки с флагами, принявшими окончательное значение (например, 0), и клетки, в которых компонента b содержит частичную сумму определителя (например, 1), а также итог вычислений (-8).

В табл. 13 приведены ЛФП алгоритма вычисления детерминанта. Условия накладывают ограничение только на флаг s. Они записаны в графическом виде. В каждом условии перечислены допустимые паттерны окрестностей. Для более сложных условий используются логические символы, знак дизъюнкции опускается. В пределах одного паттерна используются символы «|» в качестве «или» и «¬» в качестве «не». Как видно из табл. 13, эволюция флага s происходит независимо от компонент памяти a, b и c, так как их значения не фигурируют в условиях перехода. Останов происходит естественным образом, когда флаг s во всех клетках принимает значение 3, соответствующее тождественному переходу. Время работы алгоритма составляет $(3n - 4)$ шагов.

Недостатком приведённого алгоритма является то, что он неприменим для произвольной матрицы. Более того, его применимость можно выяснить лишь в процессе вычисления. Эта проблема устраняется выбором главного, то есть наибольшего по модулю элемента в столбце перед каждым этапом метода Гаусса. Состояние клетки в соответствующей модификации алгоритма имеет четыре компоненты $\langle s, a, b, c\rangle$,

аналогично простому алгоритму. Однако флаг синхронизации s принимает значения в множестве $\{0, \#, -1, -2, -3, -4, -5, 1, 2, 3, 4\}$. ЛФП представлена в табл. 14.

Запишем начальные условия:

$$s_{11} = -1;\ \text{остальные } s_{ij} = 0,$$
$$\|a_{ij}\| = A \text{ — рассматриваемая матрица,}$$
b и c произвольные.

Алгоритм с выбором главного элемента имеет уже не линейную, а квадратичную сложность, а именно n^2 шагов. Как и алгоритм 4, он имеет три компоненты памяти (одна для данных, две другие используются как буферные) и один флаг. Однако компонента флага имеет не пять состояний, а одиннадцать.

Таблица 12

Эволюция компонент в алгоритме 4 для матрицы порядка $n = 5$

t	s	a	b	c	t	s	a	b	c
0	1 0 0 0 0	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0	6	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0
	0 0 0 0 0	1 2 2 4 2	0 0 0 0 0	0 0 0 0 0		3 3 3 3 1	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1
	0 0 0 0 0	2 4 2 4 2	0 0 0 0 0	0 0 0 0 0		3 3 0 2 2	0 0 -2 -4 -4	1 -1 -2 1 3	2 2 2 2 2
	0 0 0 0 0	1 1 2 5 1	0 0 0 0 0	0 0 0 0 0		3 4 2 2 0	0 0 2 2 1	1 -1 -2 3 0	1 2 2 1 0
	0 0 0 0 0	1 4 2 2 5	0 0 0 0 0	0 0 0 0 0		3 2 2 0 0	0 0 -2 2 5	1 -1 4 0 0	1 -1 1 0 0
1	4 1 0 0 0	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0	7	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0
	2 0 0 0 0	0 2 2 4 2	1 0 0 0 0	1 0 0 0 0		3 3 3 3 3	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1
	0 0 0 0 0	2 4 2 4 2	0 0 0 0 0	0 0 0 0 0		3 3 4 1 2	0 0 -2 -4 -2	1 -1 -2 1 -1	2 2 2 2 2
	0 0 0 0 0	1 1 2 5 1	0 0 0 0 0	0 0 0 0 0		3 3 2 2 2	0 0 0 0 -2	1 -1 -2 1 3	1 2 -1 2 1
	0 0 0 0 0	1 4 2 2 5	0 0 0 0 0	0 0 0 0 0		3 3 2 2 0	0 0 -4 -1 5	1 -1 -2 3 0	1 -1 -1 1 0
2	3 3 1 0 0	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0	8	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0
	4 2 0 0 0	0 -1 2 4 2	1 3 0 0 0	1 1 0 0 0		3 3 3 3 3	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1
	2 0 0 0 0	0 4 2 4 2	1 0 0 0 0	2 0 0 0 0		3 3 3 3 1	0 0 -2 -4 -2	1 -1 -2 1 -1	2 2 2 2 2
	0 0 0 0 0	1 1 2 5 1	0 0 0 0 0	0 0 0 0 0		3 3 4 2 2	0 0 0 -4 0	1 -1 -2 -4 -1	1 2 -1 -1 2
	0 0 0 0 0	1 4 2 2 5	0 0 0 0 0	0 0 0 0 0		3 3 2 2 2	0 0 0 0 2	1 -1 2 1 3	1 -1 2 -1 1
3	3 3 3 1 0	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0	9	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0
	3 0 2 0 0	0 -1 -2 4 2	1 3 4 0 0	1 1 1 0 0		3 3 3 3 3	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1
	4 2 0 0 0	0 -2 2 4 2	1 3 0 0 0	2 2 0 0 0		3 3 3 3 3	0 0 -2 -4 -2	1 -1 -2 1 -1	2 2 2 2 2
	2 0 0 0 0	0 1 2 5 1	1 0 0 0 0	1 0 0 0 0		3 3 3 0 2	0 0 0 -4 -2	1 -1 -2 -4 -2	1 2 -1 -1 -1
	0 0 0 0 0	1 4 2 2 5	0 0 0 0 0	0 0 0 0 0		3 3 3 2 2	0 0 0 8 1	1 -1 2 -4 -1	1 -1 2 2 -1
4	3 3 3 3 1	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0	10	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0
	3 4 1 2 0	0 -1 -2 1 2	1 3 4 3 0	1 1 1 1 0		3 3 3 3 3	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1
	3 2 2 0 0	0 0 -6 4 2	1 -1 4 0 0	2 2 2 0 0		3 3 3 3 3	0 0 -2 -4 -2	1 -1 -2 1 -1	2 2 2 2 2
	4 2 0 0 0	0 -2 2 5 1	1 3 0 0 0	1 1 0 0 0		3 3 3 4 1	0 0 0 -4 -2	1 -1 -2 -4 -2	1 2 -1 -1 -1
	2 0 0 0 0	0 4 2 2 5	1 0 0 0 0	1 0 0 0 0		3 3 3 2 2	0 0 0 0 5	1 -1 2 -8 -2	1 -1 2 -2 2
5	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0	11	3 3 3 3 3	1 3 4 3 3	0 0 0 0 0	0 0 0 0 0
	3 3 3 1 2	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1		3 3 3 3 3	0 -1 -2 1 -1	1 3 4 3 3	1 1 1 1 1
	3 4 2 2 0	0 0 -2 -2 2	1 -1 -2 3 0	2 2 2 2 0		3 3 3 3 3	0 0 -2 -4 -2	1 -1 -2 1 -1	2 2 2 2 2
	3 2 2 0 0	0 0 -2 5 1	1 -1 4 0 0	1 2 1 0 0		3 3 3 3 3	0 0 0 -4 -2	1 -1 -2 -4 -2	1 2 -1 -1 -1
	3 2 0 0 0	0 1 2 2 5	1 3 0 0 0	1 1 0 0 0		3 3 3 3 3	0 0 0 0 1	1 -1 2 -8 -8	1 -1 2 -2 2

Таблица 13

ЛФП простого алгоритма вычисления детерминанта

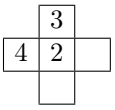
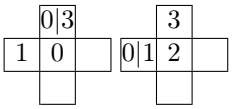
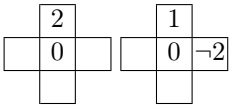
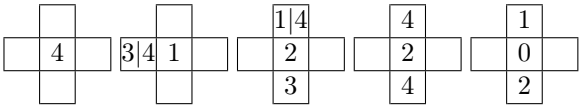
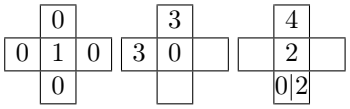
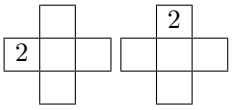
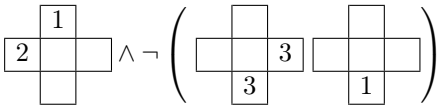
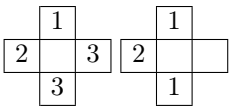
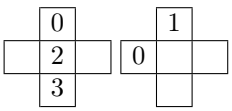
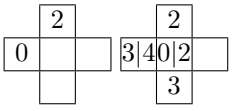
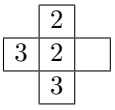
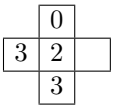
№ п/п	Условие перехода	Правило перехода
1		$s := 0$
		$s := 1$
		$s := 2$
		$s := 3$
		$s := 4$
2		$a := a - c_{\leftarrow} \cdot b_{\uparrow}$, $b := b_{\uparrow}$, $c := c_{\leftarrow}$
		$a := a - c_{\leftarrow} \cdot a_{\uparrow}$, $b := a_{\uparrow}$, $c := c_{\leftarrow}$
		$a := a - c_{\leftarrow} \cdot a_{\uparrow}$, $b := b_{\leftarrow} \cdot (a - c_{\leftarrow} \cdot a_{\uparrow})$
		$a := 0$, $c := a/a_{\uparrow}$, $b := a_{\uparrow}$
		$a := 0$, $c := a/b_{\uparrow}$, $b := b_{\uparrow}$
		$a := 0$, $c := a/b_{\uparrow}$, $b := b_{\leftarrow} \cdot b_{\uparrow}$
		$a := 0$, $c := a/a_{\uparrow}$, $b := b_{\leftarrow} \cdot a_{\uparrow}$

Таблица 14

ЛФП алгоритма нахождения детерминанта с выбором главного элемента

Условие перехода		Правило перехода
$(s=-1)\wedge(s_\uparrow=0)\wedge(\lvert a\rvert>\lvert a_\uparrow\rvert)$		$s:=-4$
$(s=-1)\wedge(s_\uparrow=0)\wedge\neg(\lvert a\rvert>\lvert a_\uparrow\rvert)$		$s:=-3$
$(s=-1)\wedge(s_\uparrow=\#)$		$s:=1$
$(s=-2)\wedge(s_\uparrow\in\{0,-3\})\wedge(\lvert a\rvert>\lvert a_\uparrow\rvert)$		$s:=-4$
$(s=-2)\wedge(s_\uparrow\in\{0,-3,-4\})\wedge\neg(\lvert a\rvert>\lvert a_\uparrow\rvert)$		$s:=0$
$(s=0)\wedge(s_\uparrow\in\{0,-3\})\wedge(s_\downarrow\in\{-1,-2\})\wedge(s_\leftarrow\notin\{-4,-5,1,2,3,4\})$		$s:=-2$
$(s=0)\wedge(s_\uparrow=\#)\wedge(s_\downarrow\in\{-1,-2\})\wedge(s_\leftarrow\notin\{-4,-5,1,2,3,4\})$		$s:=1$
$(s=0)\wedge(s_\uparrow\notin\{1,3,4\})\wedge(s_\leftarrow\in\{-4,-5\})$		$s:=-5$
$(s=0)\wedge(s_\uparrow\notin\{1,3,4\})\wedge(s_\leftarrow\in\{1,2\})$		$s:=2$
$(s=0)\wedge(s_\uparrow\notin\{1,3\})\wedge(s_\leftarrow\in\{3,4\})\wedge(s_\downarrow\notin\{\#,2\})$		$s:=4$
$(s=0)\wedge(s_\uparrow\in\{1,3\})$		$s:=3$
$(s=0)\wedge(s_\uparrow=4)\wedge(s_\leftarrow=3)\wedge(s_\downarrow=\#)$		$s:=-1$
$(s=0)\wedge(s_\uparrow=2)\wedge(s_\leftarrow=3)\wedge(s_\downarrow\in\{\#,2\})$		$s:=\#$
$(s=-3)\wedge(s_\uparrow\notin\{3,1\})\wedge(s_\downarrow=-2)$		$s:=1$
$(s=-3)\wedge(s_\uparrow\in\{3,1\})$		$s:=3$
$(s=-4)\wedge(s_\uparrow\neq1)\wedge(s_\downarrow\neq-2)$		$s:=-3$
$(s=-4)\wedge(s_\uparrow\neq1)\wedge(s_\downarrow=-2)$		$s:=1$
$(s=-4)\wedge(s_\uparrow=1)$		$s:=3$
$(s=-5)\wedge(s_\leftarrow\notin\{3,4\})$		$s:=0$
$(s=-5)\wedge(s_\leftarrow\in\{3,4\})\wedge\neg((s_\uparrow=2)\wedge(s_\downarrow\in\{\#,2\}))$		$s:=4$
$(s=-5)\wedge(s_\leftarrow=3)\wedge(s_\uparrow=2)\wedge(s_\downarrow\in\{\#,2\})$		$s:=\#$
$s=1$		$s:=\#$
$s\in\{2,3\}$		$s:=\#$
$s=4$		$s:=0$
$(s\in\{-1,-2\})\wedge(s_\leftarrow\neq1)\wedge(s_\uparrow\in\{0,-3,-4\})\wedge(\lvert a\rvert>\lvert a_\uparrow\rvert)$		$a:=-a_\uparrow$
$(s=0)\wedge(s_\leftarrow\notin\{1,2\})\wedge(s_\uparrow\notin\{1,2,3,4\})\wedge(s_\downarrow=-5)$		$a:=a_\downarrow$
$(s=0)\wedge(s_\leftarrow\in\{1,2\})$		$b:=a$
$(s=0)\wedge(s_\leftarrow\notin\{1,2\})\wedge(s_\uparrow=1)$	Всегда	$a:=0,\ b:=a_\uparrow$
	$a_\uparrow=0$	$c:=0$
	$a_\uparrow\neq0$	$c:=a/a_\uparrow$
$(s=0)\wedge(s_\leftarrow\notin\{1,2\})\wedge(s_\uparrow=3)$	Всегда	$a:=0,\ b:=b_\uparrow$
	$b_\uparrow=0$	$c:=0$
	$b_\uparrow\neq0$	$c:=a/b_\uparrow$
$(s=0)\wedge(s_\leftarrow\notin\{1,2\})\wedge(s_\uparrow\in\{2,4\})\wedge\neg((s_\uparrow=2)\wedge(s_\downarrow\in\{\#,2\}))$		$a:=a-b_\uparrow\cdot c_\leftarrow,\ b:=b_\uparrow,\ c:=c_\leftarrow$
$(s=0)\wedge(s_\leftarrow=3)\wedge(s_\uparrow=2)\wedge(s_\downarrow\in\{\#,2\})$		$a:=a-b_\uparrow\cdot c_\leftarrow,\ b:=b_\leftarrow\cdot(a-b_\uparrow\cdot c_\leftarrow)$
$(s=0)\wedge(s_\leftarrow\notin\{1,2\})\wedge(s_\uparrow\notin\{1,2,3,4\})\wedge(s_\downarrow\in\{-1,-2\})\wedge(\lvert a_\downarrow\rvert>\lvert a\rvert)$		$a:=a_\downarrow$

Окончание табл. 14

Условие перехода		Правило перехода
$(s = -3) \wedge (s_\uparrow \notin \{1, 3\}) \wedge (s_\downarrow \in \{-1, -2\}) \wedge (\lvert a_\downarrow \rvert > \lvert a \rvert)$		$a := a_\downarrow$
$(s = -3) \wedge (s_\uparrow = 1) \wedge \neg ((s_\downarrow = \#) \wedge (s_\leftarrow = \#))$	Всегда	$a := 0,\ b := a_\uparrow$
	$a_\uparrow = 0$	$c := 0$
	$a_\uparrow \neq 0$	$c := a/a_\uparrow$
$(s = -3) \wedge (s_\uparrow = 1) \wedge (s_\downarrow = \#) \wedge (s_\leftarrow = \#)$	Всегда	$a := 0,\ b := a_\uparrow \cdot b_\leftarrow$
	$a_\uparrow = 0$	$c := 0$
	$a_\uparrow \neq 0$	$c := a/a_\uparrow$
$(s = -3) \wedge (s_\uparrow = 3) \wedge \neg ((s_\downarrow = \#) \wedge (s_\leftarrow = \#))$	Всегда	$a := 0,\ b := b_\uparrow$
	$b_\uparrow = 0$	$c := 0$
	$b_\uparrow \neq 0$	$c := a/b_\uparrow$
$(s = -3) \wedge (s_\uparrow = 3) \wedge (s_\downarrow = \#) \wedge (s_\leftarrow = \#)$	Всегда	$a := 0,\ b := b_\uparrow \cdot b_\leftarrow$
	$b_\uparrow = 0$	$c := 0$
	$b_\uparrow \neq 0$	$c := a/b_\uparrow$
$(s = -4) \wedge (s_\uparrow \neq 1) \wedge (s_\downarrow \in \{-1, -2\}) \wedge (\lvert a_\downarrow \rvert > \lvert a \rvert)$		$a := a_\downarrow$
$(s = -4) \wedge (s_\uparrow = 1) \wedge \neg ((s_\downarrow = \#) \wedge (s_\leftarrow = \#))$	Всегда	$a := 0,\ b := a_\uparrow$
	$a_\uparrow = 0$	$c := 0$
	$a_\uparrow \neq 0$	$c := a/a_\uparrow$
$(s = -4) \wedge (s_\uparrow = 1) \wedge (s_\downarrow = \#) \wedge (s_\leftarrow = \#)$	Всегда	$a := 0,\ b := a_\uparrow \cdot b_\leftarrow$
	$a_\uparrow = 0$	$c := 0$
	$a_\uparrow \neq 0$	$c := a/a_\uparrow$
$(s = -5) \wedge (s_\uparrow \neq 2)$		$a := -a_\uparrow$
$(s = -5) \wedge (s_\uparrow = 2) \wedge \neg ((s_\uparrow = 2) \wedge (s_\downarrow \in \{\#, 2\}))$		$a := -a_\uparrow - a \cdot c_\leftarrow,$ $b := a,\ c := c_\leftarrow$
$(s = -5) \wedge (s_\uparrow = 2) \wedge (s_\leftarrow = 3) \wedge (s_\downarrow \in \{\#, 2\})$		$a := -a_\uparrow - a \cdot c_\leftarrow,$ $b := b_\leftarrow \cdot (-a_\uparrow - a \cdot c_\leftarrow)$
$(s = 2) \wedge (s_\downarrow = -5)$		$a := a_\downarrow$

Заключение

Из предложенных алгоритмов обратим внимание на алгоритм 3 матричного умножения и модификацию алгоритма вычисления определителя как наиболее универсальные. Так как алгоритм 3 достигает асимптотически минимальной сложности $\Theta(n)$ для топологии двумерной решётки [13], задачу умножения матриц на клеточном автомате можно считать решённой. Иначе обстоит дело с нахождением детерминанта. Как уже отмечено, модифицированный алгоритм не является оптимальным с точки зрения временны́х затрат, а простой алгоритм не определён для некоторых матриц. Более того, данные алгоритмы используют операцию деления, что ограничивает их общность. Хотелось бы найти алгоритм, опирающийся на формулу полного разложения, преобразованную вынесением общих множителей таким образом, чтобы количество слагаемых было минимально. Подобный подход был бы применим для матрицы с элементами в произвольном кольце. Выразим надежду, что такой КА-алгоритм существует и его можно найти.

ЛИТЕРАТУРА

1. *Legendi T. et al.* Megacell machine // Parallel Computing. 1988. V. 8. No. 1–3. P. 195–199.
2. *Kramer P. P. G. and van Leeuwen J.* Systolic computation and VLSI // Foundations of Computer Science IV. P. 1. / eds. J. W. de Bakker and J. van Leeuven. Amsterdam, 1983. P. 75–103.
3. *Матюшкин И. В., Заплетина М. А.* Отражение и транспонирование данных в матрице клеточно-автоматного вычислителя // Изв. вузов. Электроника. 2019. Т. 24. № 1. С. 51–63.
4. *Матюшкин И. В., Жемерикин А. В., Заплетина М. А.* Клеточно-автоматные алгоритмы сортировки строк и умножения целых чисел по схеме Атрубина // Изв. вузов. Электроника. 2016. Т. 21. № 6. С. 557–565.
5. *Матюшкин И. В., Кожевников В. С.* Клеточно-автоматные алгоритмы пермутации матриц // Труды МФТИ. 2019. Т. 11. № 1. С. 39–52.
6. *Гергель В. П.* Теория и практика параллельных вычислений: учеб. пособие. М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. 423 с.
7. *Кудрявцев В. Б., Подколзин А. С., Болотов А. А.* Основы теории однородных структур. М.: Наука, 1990. 296 с.
8. *Strassen V.* Gaussian elimination is not optimal // Numer. Math. 1969. V. 13. No. 4. P. 354–356.
9. *Cohn H., Kleinberg R., Szegedy B., and Umans C.* Group-theoretic algorithms for matrix multiplication // Proc. Ann. IEEE Symp. FOCS. 2005. P. 379–388.
10. *Фельдман Л. П., Назарова И. А., Хорошилов А. В.* Параллельные блочные алгоритмы умножения матриц для мультикомпьютеров с распределенной памятью // Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка». 2007. Вып. 8(120). С. 297–308.
11. *Katona E.* Cellular algorithms for binary matrix operations // LNCS. 1981. V. 111. P. 203–216.
12. *Stojanovic N. M., Milovanovic I. Z., Stojcev M. K., and Milovanovic E. I.* Matrix-vector multiplication on a fixed size unidirectional systolic array // 8th Intern. Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Services, Nis, Serbia. 2007. P. 457–460.
13. *Gentleman W. M.* Some complexity results for matrix computations on parallel processors // J. ACM. 1978. V. 25. No. 1. P. 112–115.
14. *Dekel E., Nassimi D., and Sahni S.* Parallel matrix and graph algorithms // SIAM J. Computing. 1981. V. 10. No. 4. P. 657–675.
15. *Moraga C.* On a case of symbiosis between systolic arrays // Integration. 1984. V. 2. No. 3. P. 243–253.
16. *Umeo H.* Firing squad synchronization algorithms for two-dimensional cellular automata // J. Cellular Automata. 2009. V. 4. No. 1. P. 1–20.
17. *Mazoyer J.* A six-state minimal time solution to the firing squad synchronization problem // Theor. Comput. Sci. 1987. V. 50. No. 2. P. 183–238.
18. *Беклемишев Д. В.* Курс аналитической геометрии и линейной алгебры: учебник для вузов. 10-е изд., испр. М.: Физматлит, 2005. 304 с.
19. *Аршон С.* Обобщенное правило Саррюса // Матем. сб. 1935. Т. 42. № 1. С. 121–128.
20. *Mahajan M. and Vinay V.* Determinant: Combinatorics, Algorithms, and Complexity. Chicago J. Theor. Comput. Sci., 1997.
21. *Csanky L.* Fast parallel inversion algorithms // SIAM J. Computing. 1976. V. 5. No. 4. P. 818–823.

22. *Berkowitz S. J.* On computing the determinant in small parallel time using a small number of processors // Inform. Processing Lett. 1984. V. 18. No. 3. P. 147–150.
23. *Beliakov G. and Matiyasevich Y.* A parallel algorithm for calculation of determinants and minors using arbitrary precision arithmetic // BIT Numerical Math. 2015. V. 56. No. 1. P. 33–50.
24. *Marco A. and Martínez J.-J.* Parallel computation of determinants of matrices with polynomial entries // J. Symbolic Computation. 2004. V. 37. No. 6. P. 749–760.
25. *Almalki S., Alzahrani S., and Alabdullatif A.* New parallel algorithms for finding determinants of $N \times N$ matrices // Proc. World Congress on Computer and Information Technology (WCCIT), 2013. P. 1–6.
26. `www.mathworks.com/help/matlab/ref/det.html` — The MathWorks, Inc., 2019.

REFERENCES

1. *Legendi T. et al.* Megacell machine. Parallel Computing, 1988, vol. 8, no. 1–3, pp. 195–199.
2. *Kramer P. P. G. and van Leeuwen J.* Systolic computation and VLSI. Foundations of Computer Science IV, part 1 (eds. J. W. de Bakker and J. van Leeuven). Amsterdam, 1983, pp. 75–103.
3. *Matushkin I. V. and Zapletina M. A.* Otrazheniye i transponirovaniye dannykh v matritse kletochno-avtomatnogo vychislitelya [Data reflection and transposition in the matrix of a cellular automata computator]. Proc. Universities. Electronics, 2019, vol. 24, no. 1, pp. 51–63. (in Russian)
4. *Matushkin I. V., Zhemerikin A. V., and Zapletina M. A.* Kletochno-avtomatnyye algoritmy sortirovki strok i umnozheniya tselykh chisel po skheme Atrubina [Cellular Automata Algorithms for String Sorting and Integer Multiplication by Atrubin's Scheme]. Proc. Universities. Electronics, 2016, vol. 21, no. 6, pp. 557–565. (in Russian)
5. *Matyushkin I. V. and Kozhevnikov V. S.* Kletochno-avtomatnyye algoritmy permutatsii matrits [Cellular automata algorithms for matrix permutations]. Proc. MIPT, 2019, vol. 11, no. 1, pp. 39–52. (in Russian)
6. *Gergel V. P.* Teoriya i praktika parallel'nykh vychisleniy [Theory and Practice of Parallel Computing]. Moscow, Internet-Universitet Informatsionnykh Tekhnologiy; BINOM Publ., 2007. 423 p. (in Russian)
7. *Kudryavtsev V. B., Podkolzin A. S., and Bolotov A. A.* Osnovy teorii odnorodnykh struktur [Basics of the Theory of Homogeneous Structures]. Moscow, Nauka Publ., 1990. 296 p. (in Russian)
8. *Strassen V.* Gaussian elimination is not optimal. Numer. Math., 1969, vol. 13, no. 4, pp. 354–356.
9. *Cohn H., Kleinberg R., Szegedy B., and Umans C.* Group-theoretic algorithms for matrix multiplication. Proc. Ann. IEEE Symp. FOCS, 2005, pp. 379–388.
10. *Feldman L. P., Nazarova I. A., and Horoshilov A. V.* Parallel'nyye blochnyye algoritmy umnozheniya matrits dlya mul'tikomp'yuterov s raspredelennoy pamyat'yu [Parallel block algorithms of matrix multiplication for computer systems with distributed memories]. Naukovi pratsi Donets'kogo natsional'nogo tekhnichnogo universitetu, seriya "Informatika, kibernetika ta obchislyuval'na tekhnika", 2007, vol. 8, no. 120, pp. 297–308. (in Russian)
11. *Katona E.* Cellular algorithms for binary matrix operations. LNCS, 1981, vol. 111, pp. 203–216.
12. *Stojanovic N. M., Milovanovic I. Z., Stojcev M. K., and Milovanovic E. I.* Matrix-vector multiplication on a fixed size unidirectional systolic array. 8th Intern. Conf. on Telecommunications in Modern Satellite, Cable and Broadcasting Services, Nis, Serbia, 2007, pp. 457–460.

13. *Gentleman W. M.* Some complexity results for matrix computations on parallel processors. J. ACM, 1978, vol. 25, no. 1, pp. 112–115.
14. *Dekel E., Nassimi D., and Sahni S.* Parallel matrix and graph algorithms. SIAM J. Computing, 1981, vol. 10, no. 4, pp. 657–675.
15. *Moraga C.* On a case of symbiosis between systolic arrays. Integration, 1984, vol. 2, no. 3, pp. 243–253.
16. *Umeo H.* Firing squad synchronization algorithms for two-dimensional cellular automata. J. Cellular Automata, 2009, vol. 4, no. 1, pp. 1–20.
17. *Mazoyer J.* A six-state minimal time solution to the firing squad synchronization problem. Theor. Comput. Sci., 1987, vol. 50, no. 2, pp. 183–238.
18. *Beklemishev D. V.* Kurs analiticheskoy geometrii i lineynoy algebry [A course of analytic geometry and linear algebra]. Moscow, Fizmatlit Publ., 2005. 304 p. (in Russian)
19. *Arshon S.* Obobshchennoye pravilo Sarryusa [Generalized Sarrus' rule]. Matematicheskiy Sbornik, 1935, vol. 42, no. 1, pp. 121–128. (in Russian)
20. *Mahajan M. and Vinay V.* Determinant: Combinatorics, Algorithms, and Complexity. Chicago J. Theor. Comput. Sci., 1997.
21. *Csanky L.* Fast parallel inversion algorithms. SIAM J. Computing, 1976, vol. 5, no. 4, pp. 818–823.
22. *Berkowitz S. J.* On computing the determinant in small parallel time using a small number of processors. Inform. Processing Lett., 1984, vol. 18, no. 3, pp. 147–150.
23. *Beliakov G. and Matiyasevich Y.* A parallel algorithm for calculation of determinants and minors using arbitrary precision arithmetic. BIT Numerical Math., 2015, vol. 56, no. 1, pp. 33–50.
24. *Marco A. and Martínez J.-J.* Parallel computation of determinants of matrices with polynomial entries. J. Symbolic Computation, 2004, vol. 37, no. 6, pp. 749–760.
25. *Almalki S., Alzahrani S., and Alabdullatif A.* New parallel algorithms for finding determinants of $N \times N$ matrices. Proc. World Congress on Computer and Information Technology (WCCIT), 2013, pp. 1–6.
26. `www.mathworks.com/help/matlab/ref/det.html` — The MathWorks, Inc., 2019.

УДК 681.142.2

ВЗАИМНО-РЕКУРРЕНТНЫЕ ФОРМУЛЫ ДЛЯ ПЕРЕЧИСЛЕНИЯ РАЗБИЕНИЙ ПРЯМОУГОЛЬНИКА[1]

А. М. Магомедов*, Т. А. Магомедов**, С. А. Лавренченко***

*Дагестанский государственный университет, г. Махачкала, Россия
**Uber, г. Амстердам, Нидерланды
***Российский государственный университет туризма и сервиса, г. Москва, Россия

Задача перечисления разбиений прямоугольника заданных целочисленных размеров $h \times w$ на прямоугольники 1×2 рассматривалась рядом авторов в связи с вопросами термодинамики потоков жидкости и проблемой перечисления совершенных паросочетаний плоского графа за полиномиальное время. В данной работе методами теории графов разработан алгоритм, компьютерное воплощение которого способно генерировать систему взаимно-рекуррентных формул для искомого перечисления разбиений прямоугольника произвольных целочисленных размеров. Решены вопросы инициализации рекуррентных формул; показано, что решение задачи организации вычислений в соответствии с системой формул, сгенерированных компьютером, сводится к топологической сортировке ациклического орграфа; сформулированы открытые задачи.

Ключевые слова: *перечисление разбиений прямоугольника, рекуррентные формулы.*

DOI 10.17223/20710410/46/9

MUTUALLY-RECURSIVE FORMULAS FOR ENUMERATING PARTITIONS OF THE RECTANGLE

A. M. Magomedov*, T. A. Magomedov**, S. A. Lawrencenko***

*Dagestan State University, Makhachkala, Russia
**Uber, Amsterdam, Netherlands
***Russian State University of Tourism and Service, Moscow, Russia

E-mail: magomedtagir1@yandex.ru, tagir.magomedov@gmail.com, lawrencenko@hotmail.com

The paper deals with the problem of enumerating the (complete) partitions of a given $h \times w$ rectangle into 1×2 rectangles (dominos). An algorithm is given to find a system of mutually recursive formulas enumerating all such partitions of a given rectangle of an arbitrary size. The idea is as follows. In fact, the process of finding all partitions is equivalent to the process of growing a binary tree T with vertices representing partial partitions of the given rectangle. In the former process, a descriptor means the border between the part already partitioned and the remaining part of the rectangle. Let $\varphi(v)$ be the number of extensions of a partial partition v to a complete partition of the rectangle. The tree-growing process terminates as soon as the descriptor of each pendant vertex of T coincides (up to symmetry) with the descriptor

[1]Работа выполнена при финансовой поддержке Дагестанского государственного университета и Отдела математики и информатики ДНЦ РАН.

of some non-pendant vertex of T. The algorithm for generating recurrence relations for calculating the number of complete partitions is based on the following: the sibling vertex y, or z, of a vertex x in the growing tree corresponds to the partial partition obtained by horizontal or vertical placement of a domino at the left upper corner of the not-yet-partitioned part of the partial partition corresponding to x. The desired recurrence relations are written upon the completion of the tree-growing process, on the base of the equation: $\varphi(x) = \varphi(y) + \varphi(z)$. The main result of this paper is an algorithm for computer generation of recurrence relations, using only the operation of integer addition. Almost all previously known formulas for solutions for the problem contain floating-point operations, which require a long computation time and significant computer resources.

Keywords: *enumeration of rectangle partitions, recurrent formulas.*

Введение

Формулировка задачи и известные результаты

В комбинаторике под *перечислением* понимается подсчёт количества или непосредственный перебор и построение всех объектов заданного типа [1].

Прямоугольник M размеров w и h будем обозначать $M(h \times w)$, количество различных способов разбиения $M(h \times w)$ на 1×2-прямоугольники (плитки) — через $f(h, w)$, при этом каждая плитка расположена горизонтально или вертикально. Требуется найти рекуррентные формулы для $f(h, w)$ и выполнить вычисления.

Задача перечисления разбиений прямоугольника заданных размеров на плитки хорошо известна. Она сводится к задаче подсчёта совершенных паросочетаний в планарном сеточном графе и допускает решение за полиномиальное время [2, 3].

Впервые задача была рассмотрена в связи с вопросами термодинамики потоков жидкости в работах [4, 5], где получена формула, которая в обозначениях данной работы имеет вид

$$f(h, w) = 2^{wh/2} \prod_{j=1}^{w} \prod_{k=1}^{h} \left(\cos^2 \frac{\pi j}{w+1} + \cos^2 \frac{\pi k}{h+1} \right)^{1/4}. \qquad (1)$$

В [6, с. 92–93] указано, что формула (1) выведена независимо авторами [4] и [5]. В [7, с. 6] утверждается, что независимо эту же формулу получил Д. Кнут. В [7] рассмотрены рациональные производящие функции для задачи при $w \leqslant 8$. Отметим, что в таблице для $f(h, w)$, $h \leqslant 12$, $w \leqslant 8$, построенной в [7], в качестве значений $f(10, 8)$, $f(11, 8)$ и $f(12, 8)$ указаны 1031151240, 8940739821 и 82741005789 соответственно, тогда как их истинные значения, вычисленные по методу, предложенному в настоящей работе, равны соответственно 1031151241, 8940739824, 82741005829 и совпадают со значениями, приведёнными в [8], а также со значениями, вычисленными компьютерной программой по формуле (1).

В [8] с помощью порождающих функций получены рекуррентные формулы для $f(h, w)$ при $w = 2, 3$ и 4 и утверждается, что не составляет труда с привлечением компьютерной программы (на языке APL) распространить идею вывода формул на случай $w > 4$. При этом сами формулы не приведены, а таблица значений $f(h, w)$ для $h = 2, 3, \ldots, 30$, $w = 0, 1, \ldots, 9$ заполнена лишь частично. Более точно, **ни для какого** $w > 4$ таблица не заполнена полностью **для всех** значений $h = 2, 3, \ldots, 30$. Так, для $w = 7$ все ячейки таблицы ниже строки $h = 21$ пусты, для $w = 9$ оставлены пустыми все ячейки ниже строки, соответствующей $h = 14$ и т. д. Подчеркнём, что все приведённые в таблице значения согласуются с результатами наших вычислений.

Операции с плавающей запятой затрудняют точные вычисления по формуле (1). Так, например, для прямоугольника $M(8 \times 16)$ результатом вычисления по формуле (1), выполненного C#-программой с применением вещественных переменных двойной точности, является значение 540061286536919, тогда как точное значение $f(8,18)$ равно 540061286536921. Этот пример показывает малопригодность формулы (1) для точных вычислений.

Трудности с выполнением точных вычислений послужили мотивацией для построения систем рекуррентных формул, использующих лишь операции сложения целых чисел, более точно — для построения алгоритма компьютерной генерации таких формул.

Различным аспектам рассматриваемой задачи посвящена бóльшая часть главы 7 работ [9, 10]. Для небольших w и h задача была предложена на VI Всероссийской олимпиаде школьников по информатике [11 – 13]. В [13] для решения задачи предложен метод динамического программирования; там же показано (см. также п. 1.1 данной работы), что при $w = 2$ значения $f(h,w)$ задаются числами Фибоначчи:

$$f(1,2) = 1,\ f(2,2) = 2;\ f(h,2) = f(h-1,2) + f(h-2,2) \text{ при } h > 2;$$

рассмотрены также следующие простые случаи: а) $f(h,w) = 0$, если и только если произведение $w \cdot h$ нечётно; б) $f(h,1) = 1$ при чётном h.

При очевидной симметрии $f(w,h) = f(h,w)$ один из параметров w и h удобно принимать за основной. В качестве такового примем w, тогда при каждом фиксированном w искомые рекуррентные формулы будут зависеть лишь от переменной h, которая может принимать при вычислениях произвольно большие значения.

В з а и м н о - р е к у р р е н т н ы е ф о р м у л ы

Нам понадобятся некоторые понятия, относящиеся к рекуррентным формулам. Если для каждой из m последовательностей

$$F[i] = \{F_{i1}, F_{i2}, F_{i3}, \ldots\},\ i = 0, 1, \ldots, m-1,$$

заданы формальные (т. е. без анализа корректности) формулы, выражающие F_{ij} в виде линейной комбинации элементов других последовательностей и элементов F_{ik}, $k \neq j$, с коэффициентами, зависящими только от i, то такие формулы будем называть *взаимно-рекуррентными формулами* (*в.р.ф.*).

Под отрезком $[a..b]$, $a < b$, последовательности будем понимать набор её элементов с индексами $a, a+1, \ldots, b$. Пусть q — заданное целое положительное число. Упорядоченную систему в.р.ф. будем называть *согласованной системой взаимно-рекуррентных формул с шагом* q (и обозначать q-в.р.ф.), если j-й элемент F_{ij} каждой последовательности $F[i]$ представлен в виде линейной комбинации элементов отрезка $[j-q+1..j]$ каждой из последовательностей $F[0], \ldots, F[i]$ (подразумевается, что коэффициенты линейной комбинации зависят только от i). Выполнение свойства согласования необходимо для организации вычислений $f(w,h)$ по q-в.р.ф., при этом уменьшение шага q является одним из факторов повышения эффективности процесса вычислений. В контексте, где значение q несущественно, будем применять термин *согласование в.р.ф.* (без указания q).

В [10, гл. 7] приведено подробное обсуждение задачи для случаев $w = 2$ и 3: для случая $w = 2$ с применением производящих функций получена известная рекуррентная формула Фибоначчи, а для $w = 3$ — система в.р.ф., состоящая из двух формул. Однако при небольших значениях w (2 и 3, как в [10], или 4 и 5, как в [14]) из рассмотрения выпадают такие существенные аспекты проблемы, как:

1) согласование в.р.ф.; другими словами, приведение в.р.ф. к виду, допускающему выполнение вычислений: после инициализации некоторого «порогового» количества q начальных элементов последовательности для каждой формулы в.р.ф. найти элементы с номером $q+1$: сначала по первой формуле в.р.ф., затем — по второй формуле и так далее;
2) выяснение зависимости порогового значения q от w.

В данной работе: 1) построен алгоритм генерации в.р.ф. для произвольного w (из соображений обозримости в.р.ф. при компьютерной реализации алгоритма мы ограничились значениями $w \leqslant 13$); 2) в качестве вспомогательной задачи решена задача согласования в интерпретации задачи упорядочения вершин ациклического ориентированного графа; 3) рассмотрены вопросы инициализации в.р.ф.; 4) выполнены вычисления.

Отметим, что после генерации в.р.ф., их согласования и инициализации выполнение вычислений со сверхбольшими целыми числами не представило затруднений, так как язык программирования C# [15], на основе которого подготовлено программное обеспечение, включает соответствующий класс BigInteger.

Вывод в.р.ф. «вручную», как это осуществлено в [10] для малых значений w, представляется малореальным уже при значениях w, близких к 10, и вовсе невыполнимым при бóльших w. Это видно из следующего утверждения, сформулированного с привлечением программного обеспечения (через $Q(w)$ обозначено количество формул в в.р.ф.).

Утверждение 1. Для $w = 2, \ldots, 13$ значения $Q(w)$ задаются табл. 1.

Таблица 1

w	2	3	4	5	6	7	8	9	10	11	12	13
$Q(w)$	1	2	3	8	10	33	38	142	158	653	709	3050

1. Генерация в.р.ф.

1.1. Клетка ветвления и частичное разбиение

Суть решения, предложенного в [10] для случая $w = 2$, заключается в следующем. Закрашенная на рис. 1, *а* клетка (см. ниже определение *клетки ветвления*) разветвляет процесс разбиения на плитки в том смысле, что входит либо в горизонтальную плитку, либо в вертикальную (рис. 1, *б* и *в* соответственно). Поэтому $f(h, 2)$ равно сумме чисел разбиений светлой фигуры рис. 1, *б* и светлой фигуры рис. 1, *в*. Очевидно, что число разбиений светлой фигуры рис. 1, *в* равно числу разбиений светлой фигуры рис. 1, *г*, т. е. $f(h-2, 2)$. Таким образом, $f(h, 2) = f(h-1, 2) + f(h-2, 2)$. Поскольку $f(1, 2) = 1$, $f(2, 2) = 2$, при $w = 2$ получаем рекуррентную формулу Фибоначчи.

Считая *начальной клеткой* горизонтальной плитки её левую клетку, а начальной клеткой вертикальной плитки — верхнюю, заметим, что упорядочение клеток исходного прямоугольника по строкам

$$M_{11}, \ldots, M_{1w}, \ldots, M_{h1}, \ldots, M_{hw} \tag{2}$$

индуцирует упорядочение плиток в каждом *полном разбиении*: из двух плиток предшествующей будем считать плитку, чья начальная клетка располагается раньше в упорядочении по строкам; для соответствующего упорядочения плиток в полном разбиении также будем применять термин «по строкам».

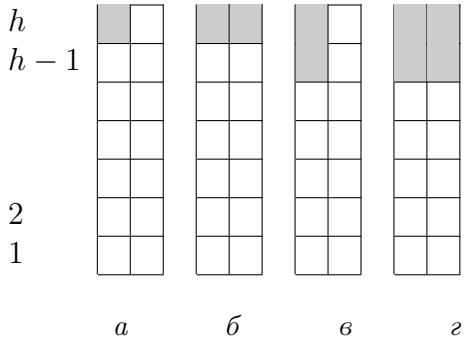


Рис. 1. Рекуррентная формула Фибоначчи, получаемая при $w = 2$ (слева — обозначения высот столбцов)

Понятие *частичного разбиения* (ч.р.) определяется рекурсивно. Пустое покрытие считается частичным разбиением и обозначается μ_{11}; клетку M_{ij} назовем *клеткой ветвления ч.р.*, если выполнены два условия: 1) M_{ij} является первой непокрытой плитками клеткой в последовательности (2); 2) клетки M_{ij}, $M_{i,j+1}$ и $M_{i+1,j}$ существуют и не покрыты плитками.

Если μ_{ij} — ч.р. с клеткой ветвления M_{ij}, то наименьшее разбиение, содержащее μ_{ij} и одну из двух пар клеток — $(M_{ij}, M_{i+1,j})$ либо $(M_{ij}, M_{i,j+1})$, — считается частичным разбиением — потомком ч.р. μ_{ij}, если обладает клеткой ветвления или равно полному разбиению.

Другими словами, первая незакрашенная клетка ч.р. называется клеткой ветвления, если её правая и нижняя соседние клетки не закрашены. Для ч.р. удобно называть занятую плитками часть прямоугольника *закрашенной фигурой*, а оставшуюся часть — *светлой фигурой*. Поскольку в перечислении разбиений без ограничения общности можно применять способ укладки плиток по строкам, в любом ч.р. закрашенной фигурой будет верхняя, а светлой — нижняя.

1.2. Громоздкость метода северо-западной клетки

Полный перебор разбиений прямоугольника $M(h\times w)$ естественно представить в виде построения двоичного ориентированного дерева. Название следующего алгоритма 1 связано с местом, занимаемым клеткой ветвления в светлой фигуре ч.р.

Алгоритм 1. Метод северо-западной клетки

1: Пустому разбиению μ_{11} сопоставить корневую вершину дерева; клеткой ветвления корневой вершины является M_{11}.

2: Осуществить последовательное рассмотрение и нумерацию листьев уровней $0, 1, \ldots$: пока имеется лист дерева, отличный от полного разбиения, для каждого такого листа — ч.р. v — включать клетку ветвления в добавляемую в ч.р. плитку — горизонтальную или вертикальную — и продолжить отдельно каждое из двух полученных разбиений до получения потомка ч.р. v.

Двоичное дерево, генерируемое методом северо-западной клетки, будем называть *полным двоичным деревом*. Рассматривая упорядоченность плиток полного разбиения по строкам в качестве хронологической последовательности укладки плиток, получим следующее утверждение.

Утверждение 2. Набор листьев полного двоичного дерева — суть все полные разбиения.

Построение полного двоичного дерева при больших значениях w и h весьма проблематично из-за громоздкости, в этом смысле утверждение 2 имеет лишь ту ценность, что служит основой для исключения (*элиминации*) полного перебора. Элиминации перебора посвящены п. 1.3 и 1.4.

Замечание 1. Там, где это не приводит к неоднозначности, для ч.р. и соответствующей ему вершины полного двоичного дерева будем использовать одно и то же обозначение.

1.3. Классы эквивалентности

Каждому ч.р. v сопоставим *дескриптор* $(v_1, \ldots, v_w)$, где

$$v_k = h - \max\{i : M_{ik} \in v,\ k = 1, \ldots, w\}$$

и индекс $\mathrm{ind}_v = \max\{v_1, \ldots, v_w\}$. Вектор $(v_k - \min\{v_1, \ldots, v_w\} : k = 1, \ldots, w)$ будем называть *нормализованным дескриптором* v. Другими словами, *дескриптор ч.р.* — вектор, образованный высотами столбцов светлой фигуры ч.р., а *индекс ч.р.* — наибольшее значение этих высот.

Замечание 2. Если придерживаться способа укладки плиток по строкам, то элементы нормализованного дескриптора любого ч.р. принадлежат множеству $\{0, 1, 2\}$.

Для ч.р. v обозначим через $\varphi(v)$ количество способов достройки v до полного разбиения. Если x и y — (непосредственные) потомки соответствующей вершины v, то

$$\varphi(v) = \varphi(x) + \varphi(y).$$

Очевидно, что $\varphi(v)$ не зависит от способа укладки плиток в закрашенной фигуре и всецело определяется конфигурацией светлой фигуры, т. е. нормализованным дескриптором ч.р. v и ind_v.

Рассмотрим двоичное поддерево, полученное в результате выполнения начальных шагов методом северо-западной клетки. Вершины с дескрипторами $(v_1, \ldots, v_w)$, $(v_w, \ldots, v_1)$ и $(v_1 + C, \ldots, v_w + C)$, где $C = \mathrm{const}$, будем называть *эквивалентными*. Другими словами, две вершины дерева назовём эквивалентными, если их нормализованные дескрипторы совпадают с точностью до симметрии. Выполним разбиение множества всех вершин на классы эквивалентности $F[0], F[1], F[2], \ldots$ Количество различных способов достройки ч.р. v (вершины дерева), принадлежащей $F[i]$, до полного разбиения будем называть *перечислением для вершины* v и обозначать $F[i, \mathrm{ind}_v]$ (рис. 2).

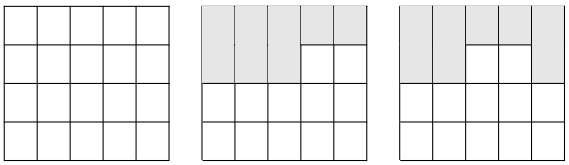


Рис. 2. Обозначение перечисления для вершины дерева — частичного разбиения (включает номер класса эквивалентности и индекс — высоту светлой фигуры)

1.4. Элиминация перебора

Лист v, полученный на некотором шаге метода северо-западной клетки, будем называть *терминальной вершиной*, если в момент создания v уже существует эквивалентная ей вершина v' степени больше единицы; при этом вершину v' будем называть *вторичной терминальной вершиной*. На рис. 3 под каждым ч.р. — вершиной дерева — приведён её дескриптор. Для ч.р., соответствующих вершинам $v_8, v_9, v_{10}, v_{11}, v_{14}$ и v_{15}, сохранены лишь обозначения, рисунки этих ч.р. удалены из соображений экономии места. Для листа v_{12} имеется эквивалентная вершина v_4 степени 3; таким образом, вершина v_{12} является терминальной, а v_4 — вторичной терминальной. Начало поддерева с корнем в вершине v будет таким же, как и начало поддерева с корнем в вершине v'. Это наблюдение служит основой для генерации взаимно-рекуррентных формул.

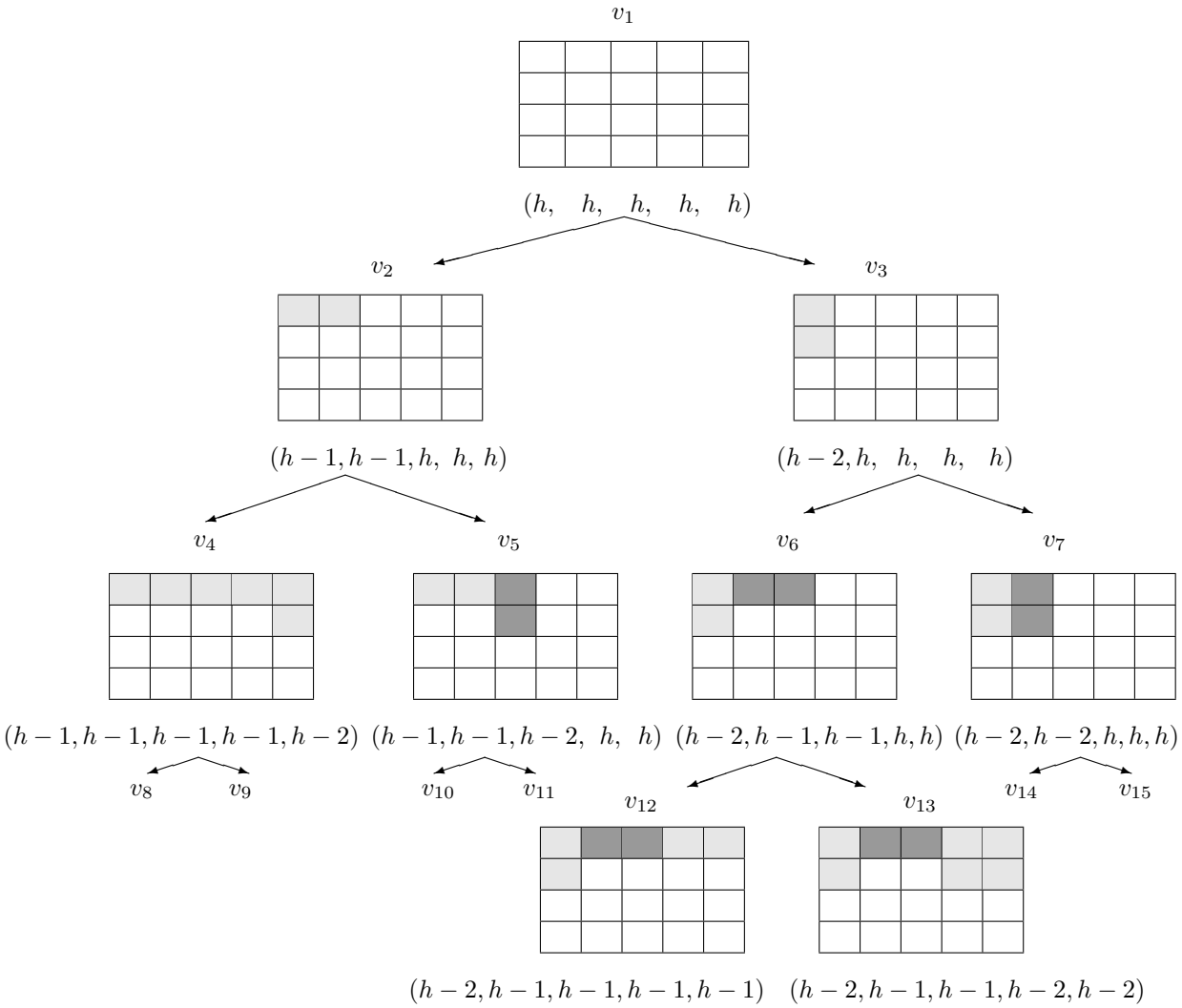


Рис. 3

В качестве критерия прерывания метода северо-западной клетки выберем достижение условия «все листья построенной части дерева являются терминальными вершинами», а построенное до прерывания дерево будем называть *элиминированным двоичным деревом*. Эффективность построения элиминированного двоичного дерева видна из следующего утверждения.

Утверждение 3. Для любой вершины v i-го уровня ($w \leqslant i$) двоичного дерева найдется такой уровень j, что $i - w \leqslant j \leqslant i + w$ и j-й уровень содержит вершину, эквивалентную вершине v.

Доказательство. В самом деле, если вдоль границы (по одну сторону от неё) ч.р. вертикально уложить w плиток, получим ч.р. с тем же нормализованным дескриптором, что и у исходного ч.р. ■

Пусть v — вторичная терминальная вершина элиминированного двоичного дерева. Двоичное поддерево T_v с корнем в вершине v будем называть *терминальным*, если среди его вершин терминальными являются только корень и листья.

Утверждение 4. Перечисление для вторичной терминальной вершины v равно сумме перечислений для листьев терминального поддерева T_v.

Справедливость утверждения 4 вытекает из формулы (2).

Приведём алгоритм 2 перечисления разбиений по терминальным поддеревьям.

Алгоритм 2. Перечисление по терминальным поддеревьям

1: Из каждого класса эквивалентности $F[i]$ выбрать по одной вторичной терминальной вершине v.
2: Построить терминальное поддерево T_v и представить перечисление $F[i, \mathrm{ind}_v]$ в виде суммы перечислений для листьев терминального поддерева T_v.

Утверждение 5. Алгоритм 2 формирует в.р.ф. для решения задачи перечисления разбиений $M(h \times w)$ за время $\mathrm{O}(2^w)$.

Доказательство. По определению терминальной вершины каждый класс эквивалентности $F[i]$ содержит не менее двух вершин; среди них в точности одна вершина терминальная, следовательно, в $F[i]$ всегда найдётся вторичная терминальная вершина. Из утверждения 4 следует, что алгоритм 2 формирует в.р.ф. для решения задачи перечисления разбиений $M(h \times w)$.

Так как построение элиминированного двоичного дерева завершается, как только в каждом ч.р. — листе дерева — все клетки верхних двух строк прямоугольника покрыты плитками, время построения в.р.ф. не превышает $\mathrm{O}(2^w)$. ■

2. Вычислительные аспекты

2.1. Структура программного обеспечения

Пусть элиминированное двоичное дерево построено (напомним, что все его листья являются терминальными вершинами). Обозначим искомое множество в.р.ф. через R, а множество терминальных вершин, включающее точно одного произвольного выбранного представителя из каждого класса эквивалентности, — через N. Сначала каждой вершине из N присвоим строковые метки вида

$$F[\text{номер класса эквивалентности, индекс}],$$

затем по алгоритму «Перечисление по терминальным поддеревьям» сгенерируем искомое множество R.

Программное сопровождение рассматриваемой задачи состоит из трёх модулей. В первом реализован алгоритм «Перечисление по терминальным поддеревьям», а также сконструированы нормализованные дескрипторы вершин множества N для передачи второму модулю. Второй модуль выполняет инициализацию начальных значений

в.р.ф. для каждого из нормализованных дескрипторов, вычисленных первым модулем. Третий модуль предназначен для выполнения расчётов в соответствии с в.р.ф., сгенерированными первым модулем, и значениями их начальных элементов, вычисленных вторым модулем; отметим, что расчёты выполняются с использованием арифметики больших целых чисел.

Приведём систему в.р.ф., сгенерированных компьютерной программой для случая $w = 8$:

$$
\begin{aligned}
F[00,H] =& F[01,H] + F[15,H] + F[05,H] + F[29,H] + F[19,H] + F[36,H-1] + F[06,H-1] + \\
&+ F[00,H-2];\\
F[01,H] =& F[02,H] + F[14,H] + F[11,H] + F[07,H-1] + F[26,H-1] + F[06,H-1];\\
F[02,H] =& F[06,H] + F[04,H-1] + F[10,H-1] + F[02,H-1];\\
F[03,H] =& F[32,H] + F[01,H-1] + F[20,H-1] + F[21,H-1] + F[14,H-1] + F[03,H-2];\\
F[04,H] =& F[17,H] + F[18,H] + F[01,H-1] + F[20,H-1] + F[02,H-1] + F[04,H-2];\\
F[05,H] =& F[13,H] + F[17,H-1] + F[08,H-1] + F[00,H-2] + F[10,H-2];\\
F[06,H] =& F[00,H-1] + F[01,H-1];\\
F[07,H] =& F[36,H] + F[01,H-1] + F[20,H-1];\\
F[08,H] =& F[06,H] + F[11,H];\\
F[09,H] =& F[06,H] + F[22,H-1] + F[16,H-2] + F[03,H-2] + F[09,H-2];\\
F[10,H] =& F[02,H] + F[05,H];\\
F[11,H] =& F[10,H-1] + F[08,H-1];\\
F[12,H] =& F[04,H-1] + F[17,H-1];\\
F[13,H] =& F[01,H-1] + F[00,H-2] + 2F[10,H-2] + F[16,H-2] + F[26,H-2];\\
F[14,H] =& F[03,H-1] + F[16,H-1] + F[09,H-1];\\
F[15,H] =& F[23,H] + F[12,H] + F[25,H-1] + F[07,H-1] + F[36,H-1];\\
F[16,H] =& F[09,H] + F[13,H] + F[14,H-1] + F[16,H-2];\\
F[17,H] =& F[36,H] + F[10,H-1] + F[04,H-2] + F[17,H-2];\\
F[18,H] =& F[04,H-1] + F[04,H-2] + F[18,H-2];\\
F[19,H] =& F[02,H-1] + F[00,H-2] + F[10,H-2];\\
F[20,H] =& F[04,H-1] + F[03,H-1] + F[07,H-1];\\
F[21,H] =& F[03,H-1] + F[28,H-2] + F[21,H-2];\\
F[22,H] =& F[14,H] + F[23,H] + F[29,H];\\
F[23,H] =& F[24,H-1] + F[03,H-1] + F[32,H-1];\\
F[24,H] =& F[32,H] + F[30,H] + F[15,H-1] + F[27,H-1] + F[31,H-1] + F[23,H-1] + \\
&+ F[24,H-2];\\
F[25,H] =& F[36,H] + F[15,H-1] + F[27,H-1];\\
F[26,H] =& F[06,H] + F[13,H];\\
F[27,H] =& F[24,H-1] + F[25,H-1];\\
F[28,H] =& F[21,H] + F[31,H] + F[32,H-1] + F[28,H-2];\\
F[29,H] =& F[32,H-1] + F[09,H-1] + F[00,H-2] + F[22,H-2];\\
F[30,H] =& F[33,H-1] + F[35,H-1] + F[24,H-2] + F[30,H-2];\\
F[31,H] =& F[24,H-1] + F[28,H-2] + F[33,H-2] + F[31,H-2];
\end{aligned}
$$

$$F[32,H] = F[36,H] + F[28,H-1] + F[22,H-1] + F[03,H-2] + F[24,H-2] + F[32,H-2];$$
$$F[33,H] = F[31,H] + F[34,H-1] + F[30,H-1] + F[33,H-2];$$
$$F[34,H] = F[33,H-1] + F[34,H-2];$$
$$F[35,H] = F[23,H] + F[30,H-1] + F[37,H-1] + F[35,H-2];$$
$$F[36,H] = F[00,H-1] + F[20,H-1] + F[27,H-1] + F[17,H-2] + F[32,H-2] + F[36,H-2];$$
$$F[37,H] = F[35,H-1] + F[03,H-2] + F[37,H-2].$$

Из алгоритма 2 следует, что в формуле с левой частью $F[i,H]$ второй индекс каждого слагаемого в правой части $\leqslant H$.

Замечание 3. Компьютерные эксперименты дают основания для уточняющего предположения: «вторые индексы суть H, $H-1$ или $H-2$», однако доказать справедливость данного предположения не удалось.

Если в в.р.ф. R присутствует формула с левой частью $F[i,H]$, содержащая в правой части слагаемое $F[j,H]$ с тем же вторым индексом, то будем говорить, что *формула j предшествует формуле i*. Для организации вычислений по в.р.ф. R «сверху вниз» требуется упорядочить R таким образом, чтобы соблюдалось условие: если для пары формул i и j известно, что j предшествует i, то в упорядочении R формула j встречается раньше, нежели формула i. Задачу такого упорядочения будем называть *задачей согласования в.р.ф.* Объясним на примере.

Из фрагмента

$$\begin{aligned} F[00,H] = {} & F[01,H] + F[15,H] + F[05,H] + F[29,H] + F[19,H] + F[36,H-1] + \\ & + F[06,H-1] + F[00,H-2]; \\ F[26,H] = {} & F[06,H] + F[13,H]; \\ F[37,H] = {} & F[35,H-1] + F[03,H-2] + F[37,H-2] \end{aligned}$$

видно, что формуле 0 должны предшествовать формулы 1, 5, 15, 19 и 29; у формулы 37 нет предшествующих формул; у формулы 26 две предшествующие формулы — 6 и 13.

Полный список отношений частичного предшествования для в.р.ф. R для случая $w = 8$ приведён в табл. 2. Каждая ячейка начинается с её номера $i = 0, 1, \ldots, 37$; номера всех формул, предшествующих в в.р.ф. R формуле i (если таковые имеются), перечислены в i-й ячейке после разделительного знака «:».

Таблица 2

Полный список частичных предшествований для случая $w = 8$

0: 1,15,5,29,19	6:	12:	18:	24: 32,30	31:
1: 2,14,11	7: 36	13:	19:	25: 36	32: 36:
2: 6	8: 6,11	14:	20:	26: 6,13	33: 31:
3: 32	9: 6	15: 23,12	21:	27:	34:
4: 17,18	10 :2,5	16: 9,13	22: 14,23,29	28: 21,31	35: 23
5: 13	11:	17: 36	23:	29:	36:
				30:	37:

Утверждение 6. В в.р.ф. R отсутствуют циклические ссылки.

Доказательство. Каждое слагаемое в этих формулах (линейных комбинациях) является целым положительным числом, поэтому наличие формулы вида

$$F[H,j] = \ldots + F[H,k] + \ldots$$

исключает наличие формулы вида

$$F[H,k] = \ldots + F[H,j] + \ldots$$

Утверждение доказано. ■

Напомним известное утверждение о топологической сортировке [16, с. 95].

Утверждение 7. Вершины ациклического ориентированного графа G на n вершинах можно таким образом пометить числами из множества $\{0, 1, \ldots, n-1\}$, что если в графе G имеется дуга (i, j), то $i < j$.

Алгоритм топологической сортировки [16, с. 96] состоит в следующем. Сначала произвольная вершина с нулевой полустепенью исхода помечается числом $n-1$ и удаляется из графа вместе с инцидентными дугами. Затем в оставшемся графе произвольная вершина с нулевой полустепенью исхода помечается числом $n-2$. Описанная процедура повторяется, пока не пометим все вершины. Об одном применении топологической сортировки для решения задачи о существовании расписаний специального типа см. в [17].

Утверждение 8. Согласование в.р.ф. R всегда выполнимо.

Доказательство. Рассмотрим орграф $G(R) = (V, E)$, вершины $v_0, v_1, \ldots, v_{n-1}$ которого соответствуют в.р.ф. R, дуги — отношению предшествования формул: дуга из вершины v_i в вершину v_j проведена тогда и только тогда, когда i-я формула предшествует j-й. По утверждениям 6 и 7 упорядочение в.р.ф. R в соответствии с топологической сортировкой вершин графа $G(R)$ обеспечивает согласование R. ■

Поскольку топологическая сортировка выполнима за линейное время, получаем следующее

Следствие 1. Согласование в.р.ф. R выполнимо за время $O(2^w)$.

Замечание 4. Задача топологической сортировки текстуально близка к задаче поиска ориентированного гамильтонова пути в каждой компоненте ориентированного графа. В общем случае задача о гамильтоновом пути NP-полна [18, с. 249], но для ациклических ориентированных графов задача об ориентированном гамильтоновом пути разрешима за полиномиальное время [19]. Заметим также, что ориентированный граф имеет цикл в том и только в том случае, когда алгоритм поиска в глубину [20] находит обратную дугу.

Замечание 5 (о сложности процедуры построения согласованных в.р.ф. R).

Процедура состоит в последовательном выполнении алгоритма 2 и алгоритма согласования. Из утверждения 5 и следствия 1 получим, что вычислительная сложность процедуры построения согласованных в.р.ф. R не превышает $O(2^w)$.

2.2. Инициализация в.р.ф.

Как отмечено в п. 2.1, модуль 1 генерирует нормализованные дескрипторы каждой вершины множества N для передачи модулю 2. Модуль 2 вычисляет начальные

значения для всех в.р.ф., составляющих R. Нетривиальным является вопрос о количестве подлежащих вычислению начальных элементов каждой из (согласованных) в.р.ф. Нетрудно видеть связь данного вопроса со значением шага в.р.ф. Компьютерные преобразования подтверждают следующую гипотезу (проверка выполнена для $w \leqslant 13$), но доказать её не удалось.

Гипотеза 1 (о шаге в.р.ф.). Сгенерированные алгоритмом «Перечисление по терминальным поддеревьям» и согласованные алгоритмом топологической сортировки в.р.ф. являются 3-согласованными взаимно-рекуррентными формулами.

Из гипотезы 1 следует, что для организации вычислений по сгенерированным согласованным в.р.ф. достаточно инициализировать по три начальных значения каждого из $F[i]$. Приведём примеры вычисленных значений $f(h, w)$:

1) $f(3, 2) = 3$;
2) $f(20, 3) = 413403$ [10, с. 379];
3) $f(2, 4) = 5$;
4) $f(100, 5) = 4995246427425596587926101947511568142197556312989986399$;
5) $f(8, 6) = 167089$ (в [13, с. 53] допущена опечатка: «$f(8, 6) = 4213133$»);
6) $f(30, 7) = 744382189686310539093281$;
7) $f(20, 8) = 3547073578562247994$ [13, с. 54];
8) $f(100, 8) = 82480872701819841011582029499055748502616126613824529802178700733106822468932478950831981372929$.

ЛИТЕРАТУРА

1. *Stanley R. P.* Enumerative Combinatorics. V. 2. Cambridge: Cambridge University Press, 1999. 228 p.
2. *Montroll E. W.* Lattice statistics // Applied Combinatorial Mathematics / ed. E. F. Beckenbach. N.Y.: John Wiley and Sons, 1964. P. 96–143.
3. *Караваев А. М., Перепечко С. Н.* Задача о димерах на цилиндрах: рекуррентные соотношения и производящие функции // Математическое моделирование. 2014. Т. 26. № 11. С. 18–22.
4. *Kateleyn P. W.* The statistic of dimers on a lattice I: The number of dimer arrangements on quadratic lattice // Physica. 1961. V. 27. P. 1209–1225.
5. *Temperley H. N. V. and Fisher M. E.* Dimer problem in statistical mechanics — an exact result // Phil. Mag. 1961. V. 6. P. 1061–1063.
6. *Matoušek J.* Thirty-three Miniatures: Mathematical and Algorithmic Applications of Linear Algebra: Amer. Math. Soc., 2010. 182 p.
7. *Klarner D. and Pollack J.* Domino tilings of rectangles with fixed width // Discr. Math. 1980. V. 32. P. 45–52.
8. *Read R. C.* A note on tiling rectangles with dominoes // Fib. Q. 1980. V. 18. No. 1. P. 24–27.
9. *Graham R. L., Knuth D. E., and Patashnik O.* Concrete Mathematics. Massachusetts: Addison-Wesley, 1994. 657 p.
10. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики: пер. с англ. М.: Мир, 1998. 703 с.
11. `http://neerc.ifmo.ru/school/archive/1993-1994/ru-olymp-roi-1994-problems.html` — Олимпиады по информатике. 1994.
12. *Кирюхин В. М.* VI Всероссийская олимпиада школьников по информатике // Информатика и образование. 1994. № 3. С. 47–50.
13. *Волченков С. Г.* Задача «Паркет» // Информатика и образование. 1994. № 3. С. 52–54.

14. *Магомедов А. М., Магомедов Т. А.* Компьютерный вывод рекуррентных формул разбиения прямоугольника // Тез. докл. X Белорусской матем. конф., 3–7 ноября 2008 г. Ч. 4. Минск: Институт математики НАН Беларуси, 2008. С. 44.
15. *Albahari J. and Albahari B.* C# 5.0 in a Nutshell. The Definitive Reference. 5th ed. O'Reilly Media, 2012. 1064 p.
16. *Swamy M. N. and Thulasiraman K.* Graphs, Networks and Algorithms. N.Y.: Wiley-Interscience, 1981. 590 p.
17. *Магомедов А. М.* Цепочечные структуры в задачах о расписаниях // Прикладная дискретная математика. 2016. № 3(33). С. 67–77.
18. *Garey M. R. and Jonson D. S.* Computers and Intractability. San Francisco: Freeman and Company, 1979. 347 p.
19. *Lawler E. L.* Combinatorial Optimization: Networks and Matroids. N.Y.: Holt, Rinehart, and Winston, 1976. 384 p.
20. *Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.* Лекции по теории графов. М.: Книжный дом «Либроком», 2009. 392 с.

REFERENCES

1. *Stanley R. P.* Enumerative Combinatorics, vol. 2. Cambridge, Cambridge University Press, 1999. 228 p.
2. *Montroll E. W.* Lattice statistics. Applied Combinatorial Mathematics (ed. E. F. Beckenbach). N.Y., John Wiley and Sons, 1964, pp. 96–143.
3. *Karavaev A. M. and Perepechko S. N.* Zadacha o dimerakh na tsilindrakh: rekurrentnye sootnosheniya i proizvodyashchie funktsii [The problem of dimers on a cylinder: recurrence relations and generating functions]. Matematicheskoe Modelirovanie, 2014, vol. 26, no. 11, pp. 18–22. (in Russian)
4. *Kateleyn P. W.* The statistic of dimers on a lattice I: The number of dimer arrangements on quadratic lattice. Physica, 1961, vol. 27, pp. 1209–1225.
5. *Temperley H. N. V. and Fisher M. E.* Dimer problem in statistical mechanics — an exact result. Phil. Mag., 1961, vol. 6, pp. 1061–1063.
6. *Matoušek J.* Thirty-three Miniatures: Mathematical and Algorithmic Applications of Linear Algebra. Amer. Math. Soc., 2010. 182 p.
7. *Klarner D. and Pollack J.* Domino tilings of rectangles with fixed width. Discr. Math., 1980, vol. 32, pp. 45–52.
8. *Read R. C.* A note on tiling rectangles with dominoes. Fib. Q., 1980, vol. 18, no. 1, pp. 24–27.
9. *Graham R. L., Knuth D. E., and Patashnik O.* Concrete Mathematics. Massachusetts, Addison-Wesley, 1994. 657 p.
10. *Graham R., Knut D., Patashnik O.* Konkretnaja Matematika. Osnovanie Informatiki [Concrete Mathematics]. Moscow, Mir Publ., 1998. 703 p. (in Russian)
11. `http://neerc.ifmo.ru/school/archive/1993-1994/ru-olymp-roi-1994-problems.html` — Olympiads in Informatics, 1994.
12. *Kirjuhin V. M.* VI Vserossijskaja olimpiada shkol'nikov po informatike [VI all-Russian Olympiad in Informatics]. Informatika i Obrazovanie, 1994, no. 3, pp. 47–50. (in Russian)
13. *Volchenkov S. G.* Zadacha «Parket» [Task «Tiling»]. Informatika i Obrazovanie, 1994, no. 3, pp. 52–54. (in Russian)
14. *Magomedov A. M. and Magomedov T. A.* Komp'juternyj vyvod rekurrentnyh formul razbienija prjamougol'nika [Computer construction of recurrence formulas for partitions of the rectangle]. Proc. X Belarusian Math. Conf., November 3–7, 2008. Part 4. Minsk, Institut of Mathematics NAS of Belarus, 2008, p. 44. (in Russian)

15. *Albahari J. and Albahari B.* C# 5.0 in a Nutshell. The Definitive Reference. 5th Ed. O'Reilly Media, 2012. 1064 p.
16. *Swamy M. N. and Thulasiraman K.* Graphs, Networks and Algorithms. N.Y., Wiley-Interscience, 1981. 590 p.
17. *Магомедов А. М.* Cepochechnye struktury v zadachah o raspisanijah [Chain structures in scheduling tasks]. Prikladnaja Diskretnaja Matematika, 2016, no. 3(33), pp. 67–77. (in Russian)
18. *Garey M. R. and Jonson D. S.* Computers and Intractability. San Francisco, Freeman and Company, 1979. 347 p.
19. *Lawler E. L.* Combinatorial Optimization: Networks and Matroids. N.Y., Holt, Rinehart and Winston, 1976. 384 p.
20. *Emelichev V. A., Mel'nikov O. I., Sarvanov V. I., and Tyshkevich R. I.* Lekcii po teorii grafov [Lectures on Graph Theory]. Moscow, Librokom Publ., 2009. 392 p. (in Russian)

ПИСЬМО В РЕДАКЦИЮ

В моей статье «Оценка стойкости AE-криптосистем типа GCM», опубликованной в 2016 г. в № 2 (32) Вашего журнала, допущена ошибка. Она содержится в последнем абзаце доказательства утверждения 4: «Так как $q_1 + q_2 = q$, $q_1 \leqslant q - 1$ и функция $q_1(q_1 + 1)(1 - q_1)$ принимает максимальное значение при $q_1 = q - 1$, получаем отсюда, что вероятность успеха атаки не превосходит $\varepsilon q(q-1)/2$, что и требуется доказать». Во-первых, вместо функции $q_1(q_1+1)(1-q_1)$ в этом утверждении должна быть функция $q_1(q_1 + 1)(q - q_1)$. Во-вторых, оценка $\varepsilon q(q-1)/2$ неверна и её следует заменить оценкой $\varepsilon(2q+3)q^2/27$, которая достигается не при $q_1 = q - 1$, а при $q_1 = 2q/3$.

В связи с этим утверждение 4 и теорема 2 (которая использует это утверждение) должны быть сформулированы в следующем виде.

Утверждение 4. Пусть G — это εU-семейство функций. Тогда при использовании не более чем q запросов к оракулу проверки метки и к оракулу генерации метки система $FH[G]$ является $\varepsilon(2q+3)q^2/27$-стойкой.

Теорема 2. Пусть $\mathcal{C}$ — противник, имеющий преимущество $\mathcal{C}_{\text{GCM'}}$ в атаке различения семейства функций, реализуемого GCM', или в активной атаке против GCM', при числе запросов к оракулам, не превосходящем q. Пусть для каждого запроса (iv, A, P) выполняются условия $l(A)+l(C) \leqslant l$ и $l(iv) = n-32$. Тогда существует различитель $\mathcal{B}$ базового шифра E, имеющий преимущество $\mathcal{B}_E$, где

$$\mathcal{C}_{\text{GCM'}} \leqslant \mathcal{B}_E + \frac{1}{27}\left\lceil \frac{l}{n} + 1 \right\rceil q^2(2q+3)/2^{\tau} + q(q+1)/2^{n+1}.$$

Требует также исправления приведённый в конце статьи числовой пример. В нём следует заменить число 2^{32} на 2^{24} и число $1{,}1 \cdot 10^{-8}$ на $4{,}42 \cdot 10^{-8}$.

Приношу свои извинения читателям и редакции журнала.

А. Ю. Зубов

СВЕДЕНИЯ ОБ АВТОРАХ

АГИБАЛОВ Геннадий Петрович — доктор технических наук, профессор, главный научный сотрудник лаборатории компьютерной криптографии Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **agibalov@mail.tsu.ru**

ДЕ ЛА КРУС ХИМЕНЕС Рейнер Антонио — научный сотрудник ООО «Центр сертификационных исследований», г. Москва. E-mail: **djr.antonio537@gmail.com**

КИСЕЛЕВА Наталья Максимовна — студентка Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **kiselyov-natalya@mail.ru**

КОЖЕВНИКОВ Владислав Сергеевич — студент Московского физико-технического института (Национального исследовательского университета), г. Долгопрудный.
E-mail: **vladislavkozhevnikov@gmail.com**

КОЙ ПУЭНТЕ Оливер — научный сотрудник ООО «Центр сертификационных исследований», г. Москва. E-mail: **o.coypuente@gmail.com**

ЛАВРЕНЧЕНКО Сергей Александрович — кандидат физико-математических наук, доцент кафедры организации бизнес-процессов в сфере туризма и сервиса Института туризма и гостеприимства Российского государственного университета туризма и сервиса, г. Москва. E-mail: **lawrencenko@hotmail.com**

ЛИПАТОВА Екатерина Сергеевна — студентка Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **katrinelipatova@gmail.com**

МАГОМЕДОВ Абдулкарим Магомедович — доктор физико-математических наук, профессор, заведующий кафедрой дискретной математики и информатики Дагестанского государственного университета, г. Махачкала.
E-mail: **magomedtagir1@yandex.ru**

МАГОМЕДОВ Тагир Абдулкаримович — программист компании «Uber», г. Амстердам. E-mail: **tagir.magomedov@gmail.com**

МАТЮШКИН Игорь Валерьевич — кандидат физико-математических наук, доцент, старший научный сотрудник Национального исследовательского университета «Московский институт электронной техники», г. Москва.
E-mail: **imatyushkin@niime.ru**

НЕПОМНЯЩАЯ Анна Шмилевна — кандидат физико-математических наук, старший научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **anep@ssd.sscc.ru**

ПАНКРАТОВА Ирина Анатольевна — кандидат физико-математических наук, доцент, заведующая лабораторией компьютерной криптографии Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **pank@mail.tsu.ru**

ПОПКОВ Кирилл Андреевич — кандидат физико-математических наук, научный сотрудник Института прикладной математики им. М. В. Келдыша РАН, г. Москва.
E-mail: **kirill-formulist@mail.ru**

РЫБАЛОВ Александр Николаевич — кандидат физико-математических наук, старший научный сотрудник лаборатории комбинаторных и вычислительных методов алгебры и логики Института математики им. С. Л. Соболева СО РАН, г. Омск.
E-mail: **alexander.rybalov@gmail.com**

СНЫТНИКОВА Татьяна Валентиновна — младший научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **snytnikovat@ssd.sscc.ru**

ТРИФОНОВА Елизавета Евгеньевна — студентка Национального исследовательского Томского государственного университета, г. Томск.
E-mail: **lizatrif@gmail.com**

ЭРНАНДЕС ПИЛОТО Даниэль Умберто — научный сотрудник ООО «Центр сертификационных исследований», г. Москва.
E-mail: **dhhernandez2410@gmail.com**