МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2016 **№ 3(33)**

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

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Адрес редакции: 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 15.09.2016.
Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 13,4. Уч.-изд. л. 15. Тираж 300 экз. Заказ № 2085.

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 519.1, 519.151

ХАРАКТЕРИЗАЦИЯ МАТРОИДОВ В ТЕРМИНАХ ПОВЕРХНОСТЕЙ[1]

А. В. Ильев*, В. П. Ильев**,***

*Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия,
**Институт математики им. С. Л. Соболева СО РАН, г. Новосибирск, Россия,
***Омский государственный университет, г. Омск, Россия

Изучаются матроиды конечного ранга и конечномерные комбинаторные геометрии. Предложено определение матроида в терминах поверхностей различного ранга, удовлетворяющих заданным аксиомам инцидентности. Доказана эквивалентность этого определения определению матроида в терминах независимых множеств. В случае обыкновенного матроида его характеризация представляет собой эквивалентное определение комбинаторной геометрии.

Ключевые слова: *матроид, поверхность, ранг, комбинаторная геометрия.*

DOI 10.17223/20710410/33/1

A CHARACTERIZATION OF MATROIDS IN TERMS OF SURFACES

A. V. Il'ev*, V. P. Il'ev**,***

*Sobolev Institute of Mathematics, Omsk, Russia,
**Sobolev Institute of Mathematics, Novosibirsk, Russia,
***Omsk State University, Omsk, Russia

E-mail: artyom_iljev@mail.ru, iljev@mail.ru

In the paper, the matroids of finite rank and finite-dimensional combinatorial geometries are studied. A definition of a matroid in terms of different rank surfaces satisfying some incidence axioms is proposed. This definition is equivalent to the definition of a matroid in terms of independent sets. In case of a simple matroid its characterization can be viewed as an equivalent definition of a combinatorial geometry.

Keywords: *matroid, surface, rank, combinatorial geometry.*

Введение

Впервые определение конечного матроида было дано в 1935 г. Х. Уитни [1]. В дальнейшем было предложено множество эквивалентных определений матроида. Например, в [2] приведены тринадцать различных эквивалентных определений матроида

[1]Работа первого автора выполнена при поддержке Программы фундаментальных научных исследований государственных академий наук на 2013–2020 гг., п. I.1.1.3. «Теоретико-модельные и алгебро-геометрические свойства алгебраических систем».

Работа второго автора выполнена при финансовой поддержке РНФ (проект № 15-11-10009).

(не считая их вариантов). Достаточно полный обзор определений матроида содержится также в [3, 4]. Большая часть этих определений может быть отнесена к следующим двум группам.

Первая группа определений. Матроид определяется как булева решётка 2^U всех подмножеств конечного множества U с выделенным семейством подмножеств.

К этой группе относится данное Уитни определение в терминах независимых множеств, где выделено непустое семейство $\mathcal{A} \subseteq 2^U$, обладающее следующими свойствами:

- (A1) если $A \in \mathcal{A}$, $B \subseteq A$, то $B \in \mathcal{A}$ (*наследственность*);

- (A2) для любых $A, B \in \mathcal{A}$, таких, что $|B| = |A| + 1$, существует элемент $b \in B \setminus A$, для которого $A \cup \{b\} \in \mathcal{A}$ (*пополнение*).

Обозначается такой матроид обычно как $M = (U, \mathcal{A})$.

Множества семейства $\mathcal{A}$ называются *независимыми*, а все остальные подмножества U — *зависимыми* множествами матроида M. Максимальные по включению независимые множества матроида M называются *базами*, а минимальные по включению зависимые множества — *циклами* матроида M. Семейства всех баз и всех циклов матроида обозначаются $\mathcal{B}$ и $\mathcal{C}$ соответственно.

Для любого множества $X \subseteq U$ *базой множества* $X \subseteq U$ называется любое его максимальное по включению независимое подмножество. Базы множества U являются базами матроида. Нетрудно показать, что в матроиде все базы любого множества имеют одинаковую мощность.

Ранговой функцией матроида $M = (U, \mathcal{A})$ называется отображение $r : 2^U \to \mathbb{Z}_+$, значением которой $r(X)$ для множества $X \subseteq U$ является мощность любой базы множества X. Ранг множества U называется *рангом матроида.*

К первой группе относятся также хорошо известные определения матроида в терминах баз (выделено семейство $\mathcal{B}$), циклов (выделено семейство $\mathcal{C}$) и другие.

Вторая группа определений. Матроид определяется как булева решетка 2^U всех подмножеств конечного множества U с заданным на 2^U отображением.

Наиболее известными определениями второй группы являются определение в терминах ранговой функции и следующее определение в терминах оператора замыкания.

Матроид — это пара $M = (U, \varphi)$, где U — непустое конечное множество, φ — отображение булевой решётки 2^U всех подмножеств множества U в себя, которое ставит в соответствие любому множеству $X \subseteq U$ его замыкание $\overline{X}$ и обладает следующими свойствами:

- (φ1) $X \subseteq \overline{X}$ для любого $X \subseteq U$ (*направленность*);

- (φ2) для любых $X, Y \subseteq U$ если $X \subseteq Y$, то $\overline{X} \subseteq \overline{Y}$ (*монотонность*);

- (φ3) $\overline{\overline{X}} = \overline{X}$ для любого $X \subseteq U$ (*идемпотентность*);

- (φ4) для любых элементов $u, v \in U$ и любого подмножества $X \subseteq U$ если $u \notin \overline{X}$ и $u \in \overline{X \cup \{v\}}$, то $v \in \overline{X \cup \{u\}}$ (*свойство замены*).

Матроид $M = (U, \varphi)$ называется *обыкновенным*, если он, кроме того, обладает свойством (φ5):

- (φ5) $\overline{\varnothing} = \varnothing$ и $\overline{\{u\}} = \{u\}$ для любого $u \in U$.

Подмножество $X \subseteq U$ называется *замкнутым*, если $X = \overline{X}$. Замкнутые множества матроида $M = (U, \varphi)$ называют его *листами* или *поверхностями.*

Наряду с конечными матроидами изучают также матроиды общего вида, основное множество U в которых может быть бесконечным. Как правило, в таких матроидах накладывают некоторые ограничения на ранг. Например, *матроид конечного ранга* в терминах независимых множеств определяется как булева решётка 2^U всех подмножеств произвольного (возможно, бесконечного) множества U с выделенным семейством $\mathcal{A} \subseteq 2^U$, обладающим свойствами (A1), (A2) и

- (A3) существует такое число $r \in \mathbb{N}$, что для любого $A \in \mathcal{A}$ выполнено $|A| \leqslant r$ (*свойство конечности ранга*).

Заметим, что класс матроидов конечного ранга представляет собой объединение всех классов матроидов фиксированного ранга k по всем $k \in \mathbb{N}$.

В определении матроида конечного ранга в терминах оператора замыкания свойство конечности ранга выглядит следующим образом:

- (φ6) для любого $A \subseteq U$ существует такое $B \subseteq A$, что $|B| < \infty$ и $\overline{B} = \overline{A}$.

Определение в терминах оператора замыкания часто принимают в качестве определения комбинаторной геометрии, отождествляя комбинаторные геометрии и обыкновенные матроиды [5–7], а именно: *комбинаторная геометрия* — это пара $M = (U, \varphi)$, где U — непустое множество; φ — отображение булевой решётки 2^U всех подмножеств множества U в себя, которое ставит в соответствие любому множеству $X \subseteq U$ его замыкание $\overline{X}$ и обладает свойствами (φ1)–(φ6). Если в этом определении не требовать выполнения условия (φ5), то получим определение *комбинаторной предгеометрии.* Таким образом, комбинаторная предгеометрия и матроид — это один и тот же объект.

Известны и другие определения комбинаторной геометрии. Например, в [8] даётся такое определение комбинаторной геометрии в терминах поверхностей (аналогичное определение матроида в терминах поверхностей можно найти в [3, 9]).

Комбинаторная геометрия — это пара $M = (U, \mathcal{F})$, где U — непустое конечное множество *точек*; $\mathcal{F}$ — непустое семейство его подмножеств (*поверхностей*), обладающих следующими свойствами:

- (F1) если $F_1, F_2 \in \mathcal{F}$, то $F_1 \cap F_2 \in \mathcal{F}$;
- (F2) $\varnothing \in \mathcal{F}$, $U \in \mathcal{F}$, а также $\{x\} \in \mathcal{F}$ для любого $x \in U$;
- (F3) для любой поверхности $F \in \mathcal{F}$ ($F \neq U$) поверхности, которые покрывают F, образуют разбиение оставшихся точек.

Здесь «*E покрывает F*» означает, что $F \subset E$, причём $F \subset G \subset E$ не выполнено ни для какой поверхности $G \in \mathcal{F}$.

Как показано в [3, 8, 9], это определение эквивалентно предыдущему и поэтому может рассматриваться как эквивалентное определение обыкновенного матроида в терминах поверхностей. Как видно, оно также относится к первой группе определений.

Весьма естественно выглядело бы определение комбинаторной геометрии как геометрической конфигурации, т. е. системы поверхностей различного ранга, удовлетворяющих заданным аксиомам инцидентности. В наибольшей степени этим требованиям отвечает определение, данное Дж. Мейсоном в [10] (оно приведено в следующем пункте), однако это определение задаёт объекты более широкого класса, чем класс обыкновенных и даже класс матроидов общего вида. Более подробно об этом сказано далее.

Хотя изучению вопросов, связанных с комбинаторными геометриями, посвящена обширная литература (см., например, [9, 11–18]), ни в одной из работ не содержится общего геометрического определения комбинаторной предгеометрии (геометрии), которое было бы эквивалентно определению матроида (обыкновенного матроида).

В настоящей работе предложено геометрическое определение конечномерной комбинаторной предгеометрии, эквивалентное определению матроида общего вида конечного ранга. Дано также аналогичное определение комбинаторной геометрии.

1. Матроиды как геометрические конфигурации: предварительные сведения

По-видимому, первым, кто выявил геометрические свойства матроида, был С. Маклейн. Уже в 1936 г. в работе [19] он предложил геометрическую интерпретацию обыкновенных матроидов как n-мерных схематических геометрических фигур. Его определения этих фигур выглядят следующим образом.

Определение 1. *Схематическая плоская фигура* — это система, состоящая из конечного числа *точек* и некоторых множеств этих точек, называемых *прямыми*, обладающая следующими свойствами:

- (P1) любая пара точек лежит на единственной прямой;
- (P2) каждая прямая содержит по меньшей мере две точки;
- (P3) никакая прямая не содержит все точки;
- (P4) существуют по крайней мере две различные точки.

Определение 2. *Схематическая пространственная фигура* — это система, состоящая из конечного числа *точек* и некоторых множеств этих точек, называемых *прямыми* и *плоскостями*, обладающая свойствами (P1)–(P4) и, кроме того, следующими свойствами:

- (S1) любые три точки, не лежащие на одной прямой, лежат в единственной плоскости;
- (S2) каждая плоскость содержит по меньшей мере три точки, не лежащие на одной прямой;
- (S3) никакая плоскость не содержит все точки;
- (S4) если плоскость содержит две точки прямой, то она содержит все точки данной прямой.

Далее Маклейн указывает, что аналогично можно определить *схематическую n-мерную фигуру*, которая состоит из конечного числа *точек* и *k-мерных плоскостей* для $k = 1, \dots, n-1$, но точного определения не даёт. Маклейн утверждает, что существует взаимно однозначное соответствие между матроидами ранга $n+1$ и схематическими n-мерными фигурами, но не приводит доказательства.

Поскольку в работе [19] в явном виде не сформулировано определение в n-мерном случае, то возник вопрос о существовании полностью геометрического определения комбинаторной геометрии в терминах поверхностей определённого ранга. В литературе удалось найти несколько определений в терминах поверхностей (см. введение), но в явном виде ранг поверхности присутствует только в следующем определении.

Определение 3 [10]. *Комбинаторная геометрия* — это конечное множество поверхностей с соответствующим рангом $k \geqslant 0$, таких, что всякая k-поверхность и 1-поверхность (точка), не лежащая на ней, лежат в единственной $(k+1)$-поверхности.

Однако если поверхностям комбинаторной геометрии в определении Мейсона поставить в соответствие поверхности обыкновенного матроида, а под рангом поверхности понимать значение ранговой функции на множестве всех точек данной поверхности, то данное в работе Мейсона определение комбинаторной геометрии не эквивалентно ранее приведённому определению обыкновенного матроида в терминах поверхностей, что подтверждается следующим примером.

Пример 1. $U = \{1, 2, 3, 4, 5, 6, 7\}$ — множество точек.

Поверхность ранга 0: $\varnothing$.

Поверхности ранга 1: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{6\}$, $\{7\}$ — точки.

Поверхности ранга 2: $\{1,2,3\}$, $\{1,4,5\}$, $\{1,6,7\}$, $\{2,4,7\}$, $\{2,5,6\}$, $\{3,4,6\}$, $\{3,5,7\}$.

Поверхности ранга 3: $\{1,2,3,4,5,6\}$, $\{1,2,3,7\}$, $\{1,2,4,7\}$, $\{1,2,6,7\}$, $\{1,3,5,7\}$, $\{1,3,6,7\}$, $\{1,4,5,7\}$, $\{1,4,6,7\}$, $\{1,5,6,7\}$, $\{2,3,4,7\}$, $\{2,3,5,7\}$, $\{2,4,5,7\}$, $\{2,4,6,7\}$, $\{2,5,6,7\}$, $\{3,4,5,7\}$, $\{3,4,6,7\}$, $\{3,5,6,7\}$.

Поверхность ранга 4: само множество $U = \{1,2,3,4,5,6,7\}$.

Легко видеть, что этот набор поверхностей удовлетворяет определению Мейсона, но не является обыкновенным матроидом, поскольку в нём нарушена аксиома (F1). Например, пересечение поверхностей $\{1,2,3,7\}$ и $\{1,2,4,7\}$ — множество $\{1,2,7\}$ — не является поверхностью.

Естественно возникает задача дополнить определение Мейсона до эквивалентного определению обыкновенного матроида, после чего отделить комбинаторные предгеометрии от геометрий аналогично определениям матроида и обыкновенного матроида в терминах замыкания. Это и сделано в данной работе. Кроме того, в отличие от Маклейна, который ведёт речь только о конечных матроидах фиксированного ранга, мы даём геометрическое определение как конечных, так и бесконечных матроидов конечного ранга.

2. Геометрические свойства матроидов

В общем случае определение матроида конечного ранга в терминах независимых множеств выглядит следующим образом.

Определение 4 [5]. *Матроид* — это пара $M = (U, \mathcal{A})$, где U — непустое (возможно, бесконечное) множество; $\mathcal{A}$ — непустое семейство его подмножеств (называемых *независимыми*), обладающее следующими свойствами:

- (A1) если $A \in \mathcal{A}$, $B \subseteq A$, то $B \in \mathcal{A}$ (наследственность);

- (A2) для любых $A, B \in \mathcal{A}$, таких, что $|B| = |A| + 1$, существует элемент $b \in B \setminus A$, для которого $A \cup \{b\} \in \mathcal{A}$ (пополнение);

- (A3) существует такое число $r \in \mathbb{N}$, что для любого $A \in \mathcal{A}$ выполнено $|A| \leqslant r$ (конечность ранга).

Пусть $M = (U, \mathcal{A})$ — матроид, где $\mathcal{A}$ — семейство его независимых множеств. Фиксируем $A \in \mathcal{A}$ — независимое множество мощности k. Определим *поверхность ранга* k как множество

$$F(A) = A \cup \{u \in U : A \cup u \notin \mathcal{A}\}.$$

Лемма 1. Пусть $A, B \in \mathcal{A}$, $|A| = |B|$, причём $A \neq B$ и $B \subseteq F(A)$. Тогда $F(A) = F(B)$.

Доказательство.

1) Докажем, что $F(A) \subseteq F(B)$. Предположим противное, т.е. существует $a \in F(A) \setminus F(B)$. Это означает, что $B \cup \{a\} \in \mathcal{A}$.

Воспользуемся свойством пополнения (A2): $|B \cup \{a\}| = |A| + 1$, поэтому существует $b \in (B \cup \{a\}) \setminus A$, такой, что $A \cup \{b\} \in \mathcal{A}$. С другой стороны, $A \cup \{b\} \subseteq F(A)$ — противоречие с определением $F(A)$. Следовательно, $F(A) \subseteq F(B)$.

2) Докажем, что $F(B) \subseteq F(A)$. Предположим противное, т.е. существует $b \in F(B) \setminus F(A)$. Это означает, что $B \cup \{b\} \notin \mathcal{A}$ и $A \cup \{b\} \in \mathcal{A}$.

Так как $|A \cup \{b\}| = |B| + 1$, в силу свойства (A2) существует $a \in (A \cup \{b\}) \setminus B$, такой, что $B \cup \{a\} \in \mathcal{A}$. Поскольку $B \cup \{b\} \notin \mathcal{A}$, то $a \neq b$. Кроме того, в п. 1 доказано, что $F(A) \subseteq F(B)$. Поэтому $B \cup \{a\} \subseteq F(B)$, т.е. получаем противоречие с определением $F(B)$. Это означает, что $F(B) \subseteq F(A)$. Таким образом, $F(B) = F(A)$. ■

Следствие 1. Поверхность ранга k однозначно определяется любым независимым множеством мощности k, содержащимся в этой поверхности.

Следствие 1 даёт возможность говорить о поверхности ранга k без указания конкретного независимого множества мощности k, определяющего эту поверхность. Таким образом, семейство поверхностей матроида определяется по следующему правилу:

$$\mathcal{F} = \{F \subseteq U : \exists A \in \mathcal{A}\ (F = F(A))\}. \tag{1}$$

Лемма 2. Пусть $M = (U, \mathcal{A})$ — матроид, $A \subseteq U$ — подмножество его элементов мощности k, не лежащее ни в какой поверхности ранга, меньшего k. Тогда $A \in \mathcal{A}$.

Доказательство. От противного. Пусть $A \notin \mathcal{A}$ — множество мощности k, такое, что для любого независимого множества B мощности, меньшей k, выполнено $A \not\subseteq F(B)$. Но из множества A можно выделить максимальное по включению независимое подмножество A'. Его мощность $|A'| < k$, и $A \subseteq F(A')$ по определению поверхности, т. е. A лежит в поверхности ранга, меньшего k, — противоречие. ■

Теорема 1. Для всякого матроида $M = (U, \mathcal{A})$, поверхности которого определены в соответствии с правилом (1), выполнены свойства (G1)–(G5):

- (G1) поверхность ранга 0 существует;

- (G2) никакая поверхность ранга k не лежит в поверхности ранга $k-1$;

- (G3) всякая поверхность ранга k и точка, не лежащая на ней, лежат в единственной поверхности ранга $k+1$;

- (G4) любые k точек, не лежащие ни в какой поверхности ранга, меньшего k, лежат в единственной поверхности ранга k;

- (G5) существует такое число $r \in \mathbb{N}$, что ранг любой поверхности не превосходит r.

Доказательство. Свойство (G1) следует из независимости $\varnothing$. Поверхностью ранга 0 является $F(\varnothing)$.

Свойство (G2) доказывается от противного. Предположим, что существуют такие независимые множества A и B, что $|A| = k-1$, $|B| = k$ и $F(B) \subseteq F(A)$. Тогда в силу свойства пополнения (A2) существует элемент $b \in B \setminus A$, такой, что $A \cup \{b\} \in \mathcal{A}$. Но поскольку $A \cup \{b\} \subseteq F(A)$, получаем противоречие с определением $F(A)$.

Докажем свойство (G3). Пусть $F(A)$ — поверхность ранга k и элемент $b \notin F(A)$. Это означает, что $A \cup \{b\} \in \mathcal{A}$. Поэтому в силу следствия 1 $F(A \cup \{b\})$ — единственная поверхность ранга $k+1$, содержащая поверхность $F(A)$ и элемент b.

Докажем свойство (G4). Пусть $A = \{a_1, \ldots, a_k\}$ — произвольное множество, которое не лежит ни в какой поверхности ранга, меньшего k. По лемме 2 оно независимо и, следовательно, лежит в единственной поверхности ранга k в силу следствия 1.

Свойство (G5) следует из свойства (A3) матроида. ■

3. Основные свойства комбинаторных предгеометрий

Определение комбинаторной предгеометрии в конечном случае выглядит так.

Определение 5. *Комбинаторная предгеометрия* — это пара $(U, \mathcal{F})$, где U — непустое конечное множество *точек*; $\mathcal{F}$ — семейство его подмножеств — *поверхностей*, каждой из которых приписан *ранг* $k \in \mathbb{Z}_+$, обладающих следующими свойствами:

- (G1) поверхность ранга 0 существует;

- (G2) никакая поверхность ранга k не лежит в поверхности ранга $k-1$;

- (G3) всякая поверхность ранга k и точка, не лежащая на ней, лежат в единственной поверхности ранга $k+1$;

- (G4) любые k точек, не лежащие ни в какой поверхности ранга, меньшего k, лежат в единственной поверхности ранга k.

Чтобы определить класс конечномерных *комбинаторных предгеометрий*, включающий как конечные, так и бесконечные объекты, в определении 5 нужно потребовать выполнение свойства (G5):

- (G5) существует такое число $r \in \mathbb{N}$, что ранг любой поверхности не превосходит r.

Замечание 1. Несложно проверить, что система аксиом (G1)–(G5) *независима*, т. е. любая из них не является следствием остальных аксиом.

Замечание 2. В любой комбинаторной предгеометрии существует единственная поверхность ранга 0 (это легко доказывается применением аксиом (G1) и (G4) для $\varnothing$).

Определение 6. Множество точек $\{a_1, \ldots, a_k\}$ называется *независимым множеством* точек комбинаторной предгеометрии, если оно не лежит ни в какой поверхности ранга, меньшего k, т. е. семейство $\mathcal{A}$ независимых множеств точек определяется следующим образом:

$$\mathcal{A} = \{A \subseteq U : \forall F \in \mathcal{F}\ ((r(F) < |A|) \Rightarrow (A \not\subseteq F))\}. \tag{2}$$

Заметим, что $\varnothing$ — независимое множество по определению.

Лемма 3. Пусть $A = \{a_1, \ldots, a_k\}$ — независимое множество точек, F — поверхность ранга k, такая, что $A \subseteq F$. Тогда для любой точки $a \notin F$ множество $A \cup \{a\}$ является независимым множеством точек.

Доказательство. В поверхности ранга, меньшего k, множество $A \cup \{a\}$ лежать не может, потому что это противоречило бы независимости множества A. По свойству (G4) F — единственная поверхность ранга k, в которой лежит A. Таким образом, множество $A \cup \{a\}$ не лежит ни в какой поверхности ранга, меньшего $k + 1$, а значит, оно независимо по определению. ■

Теперь докажем основные свойства независимых множеств точек комбинаторной предгеометрии.

Теорема 2. Во всякой комбинаторной предгеометрии семейство $\mathcal{A}$, определённое по правилу (2), обладает следующими свойствами:

1) любое подмножество независимого множества точек само является независимым множеством (наследственность);
2) если A и B — произвольные независимые множества точек, для которых выполнено $|B| = |A| + 1$, то существует точка $b \in B \setminus A$, такая, что $A \cup \{b\}$ — независимое множество (пополнение);
3) существует число $r \in \mathbb{N}$, такое, что для любого $A \in \mathcal{A}$ выполнено $|A| \leqslant r$ (конечность ранга).

Доказательство. Пусть $(U, \mathcal{F})$ — комбинаторная предгеометрия. В силу свойства (G5) для неё существует такое число $r \in \mathbb{N}$, что ранг любой поверхности данной комбинаторной предгеометрии не превосходит r.

1) Докажем свойство наследственности. Пусть $k \in \{1, \ldots, r\}$. Докажем, что любое подмножество независимого множества $A = \{a_1, \ldots, a_k\}$, состоящее из $k - 1$ точек, является независимым.

Предположим противное, т. е. существует точка a_i, такая, что множество точек

$$A_i = A \setminus \{a_i\} = \{a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_k\}$$

не является независимым множеством. В этом случае по определению должна существовать поверхность F ранга $l < k - 1$, в которой лежало бы A_i.

Точка a_i не лежит в F, иначе всё множество A лежало бы в F — поверхности ранга, меньшего k, что противоречит независимости множества A. Тогда по свойству (G3) существует единственная поверхность H ранга $l+1$, в которой F и a_i, а значит, и всё множество A лежат в H. Поскольку $l < k-1$, то $l+1 < k$, и получается, что множество A лежит в поверхности ранга, меньшего k, что противоречит независимости A.

Рассуждая аналогичным образом, можно показать, что любые подмножества множества A, состоящие из $k-2$, $k-3$, $\ldots$, 1 точки, независимы.

2) Докажем свойство пополнения. Пусть $A = \{a_1, \ldots, a_k, c_1, \ldots, c_l\}$ и $B = \{b_1, \ldots, b_k, b_{k+1}, c_1, \ldots, c_l\}$ — независимые множества, где $k, l \in \mathbb{Z}_+$; $k+l+1 \leqslant r$ и $a_i \neq b_j$ для любых $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, k+1\}$. Тогда $A \cap B = \{c_1, \ldots, c_l\}$.

Предположим противное, что для любой точки $b_j \in B \setminus A$ множество $A \cup \{b_j\}$ не является независимым, т. е. существует некоторая поверхность ранга, меньшего $k+l+1$, в которой лежит $A \cup \{b_j\}$.

Поскольку A независимо, оно не лежит ни в какой поверхности ранга, меньшего $k+l$. Однако в силу аксиомы (G4) существует единственная поверхность F ранга $k+l$, которая содержит множество A. С учётом предположения, что $A \cup \{b_j\}$ не является независимым множеством, оно также должно лежать в F.

Таким образом, для любой точки $b_j \in B \setminus A$ множество $A \cup \{b_j\}$ лежит в F, а значит, всё множество B лежит в F — поверхности ранга $k+l$, что противоречит независимости B.

3) Свойство конечности ранга следует из (G3) и (G5). ■

4. Взаимно однозначное соответствие между матроидами и комбинаторными предгеометриями

Эквивалентность определения матроида конечного ранга приведённому в предыдущем пункте определению конечномерной комбинаторной предгеометрии вытекает из следующей теоремы.

Теорема 3 (Основная теорема).

1) Пусть $M = (U, \mathcal{A})$ — матроид, где U — непустое множество его элементов; $\mathcal{A}$ — семейство его независимых множеств. Тогда семейство $\mathcal{F}$, определённое по правилу (1), обладает свойствами (G1)–(G5), причём имеет место равенство (2).

2) Пусть $(U, \mathcal{F})$ — комбинаторная предгеометрия, где U — непустое множество точек, а $\mathcal{F}$ — семейство её поверхностей. Тогда семейство $\mathcal{A}$, определённое по правилу (2), обладает свойствами (A1)–(A3), причём имеет место равенство (1).

Доказательство.

1) Свойства (G1)–(G5) следуют из теоремы 1, а равенство (2) — из леммы 2.

2) Свойства (A1)–(A3) следуют из теоремы 2.

Чтобы доказать равенство (1), сначала покажем, что всякая поверхность ранга k содержит в себе независимое множество точек мощности k. Предположим противное, что для некоторой поверхности F максимальная мощность её независимого подмножества равна $k-1$. Тогда по свойству (G4) существует единственная поверхность H ранга $k-1$, определённая для некоторого независимого множества $A \subseteq F$ мощности $k-1$. Заметим, что F не содержит точек, не лежащих в H, поскольку если существует точка $a \in F \setminus H$, то в силу леммы 3 множество $A \cup \{a\}$ является независимым вопреки максимальности множества A в поверхности F. Таким образом, $F \subseteq H$, что противоречит аксиоме (G2). Аналогично с использованием аксиомы (G3) можно показать, что мощность максимального независимого подмножества поверхности F не может быть меньше $k-1$.

Теперь докажем равенство (1), т. е. тот факт, что любая поверхность F ранга k может быть определена при помощи независимого множества точек $A \subseteq F$ мощности k по правилу

$$F = A \cup \{u \in U : A \cup \{u\} \notin \mathcal{A}\}.$$

В силу свойства (G4) поверхность F задаётся множеством A единственным образом. По лемме 3 все точки $u \in U$, такие, что $A \cup \{u\}$ не является независимым множеством, лежат в F. При этом других точек поверхность F содержать не может, поскольку это противоречит понятию независимого множества точек. ∎

Теперь дадим определения обыкновенного матроида и комбинаторной геометрии.

Определение 7. Обыкновенный матроид — это пара $M = (U, \mathcal{A})$, где U — непустое множество; $\mathcal{A}$ — непустое семейство его независимых подмножеств, обладающее свойствами (A1)–(A3) и, кроме того, следующими свойствами:

- (A4) для любого $u \in U$ выполнено $\{u\} \in \mathcal{A}$;
- (A5) для любых $u, v \in U$ если $u \neq v$, то $\{u, v\} \in \mathcal{A}$.

Определение 8. *Комбинаторная геометрия* — это комбинаторная предгеометрия, для которой выполнено свойство (G6):

- (G6) $\varnothing$ и все точки являются поверхностями.

Замечание 3. В любой комбинаторной геометрии поверхностями ранга 1 являются точки, и только они.

Из следующей теоремы вытекает эквивалентность предложенного определения комбинаторной геометрии определению обыкновенного матроида.

Теорема 4. Существует взаимно однозначное соответствие между обыкновенными матроидами и комбинаторными геометриями.

Доказательство.

1) Пусть $M = (U, \mathcal{A})$ — обыкновенный матроид, где U — непустое множество его элементов; $\mathcal{A}$ — семейство его независимых подмножеств. По теореме 3 семейство $\mathcal{F}$, определённое по правилу (1), обладает свойствами (G1)–(G5), причём имеет место равенство (2).

Докажем свойство (G6). По свойству наследственности $\varnothing \in \mathcal{A}$. Таким образом, поверхность ранга 0 совпадает с пустым множеством в силу свойства (A4). Аналогично, поскольку все одноэлементные множества являются независимыми, для любого из них определяемая им поверхность ранга 1 совпадает с самим одноэлементным множеством в силу свойства (A5).

2) Пусть $(U, \mathcal{F})$ — комбинаторная геометрия, где U — непустое множество точек, а $\mathcal{F}$ — семейство её поверхностей. По теореме 3 семейство $\mathcal{A}$, определённое по правилу (2), обладает свойствами (A1)–(A3), причём имеет место равенство (1).

Докажем свойство (A4). В силу замечания 2 и свойств (G2) и (G4) комбинаторной геометрии $\varnothing$ является единственной поверхностью ранга 0. Поскольку никакая точка не лежит в поверхности ранга 0, любое множество комбинаторной геометрии, состоящее из единственной точки, является независимым. Таким образом, свойство (A4) доказано.

Теперь докажем свойство (A5). Применив аксиомы (G3) и (G4) к $\varnothing$ и точкам, легко сделать вывод, что в комбинаторной геометрии поверхностями ранга 1 являются все точки, и только они. Отсюда следует, что любое множество, состоящее из двух точек, независимо, поскольку оно не лежит ни в какой поверхности ранга, меньшего 2. ∎

Заключение

Соответствие между матроидами конечного ранга и конечномерными комбинаторными предгеометриями, установленное в теореме 3, является взаимно однозначным. Поэтому определение комбинаторной предгеометрии можно рассматривать как эквивалентное определение матроида в терминах поверхностей заданного ранга. Аналогично, из теоремы 4 следует, что определение комбинаторной геометрии является эквивалентным определением обыкновенного матроида в данных терминах.

ЛИТЕРАТУРА

1. *Whitney H.* On the abstract properties of linear dependence // Amer. J. Math. 1935. V. 57. P. 509–533.
2. *Brylawski T.* Appendix of matroid cryptomorphisms // Theory of Matroids. Encyclopedia of Mathematics and its Applications 26 / ed. N. White. Cambridge: Cambridge University Press, 1986. P. 298–312.
3. *Schrijver A.* Combinatorial Optimization. V. B. Berlin, Heidelberg, N. Y.: Springer Verlag, 2003. 1881 p.
4. *Welsh D. J. A.* Matroid Theory. London, N. Y., San Francisco: Academic Press, 1976. 433 p.
5. *Айгнер М.* Комбинаторная теория. М.: Мир, 1982. 558 с.
6. *Рыбников К. А.* Введение в комбинаторный анализ. М.: Изд-во МГУ, 1985. 308 с.
7. *Crapo H. H. and Rota G.-C.* On the Foundations of Combinatorial Theory. II. Combinatorial Geometries. Cambridge, MA: MIT Press, 1970. 293 p.
8. *Van Lint J. H. and Wilson R. M.* A Course in Combinatorics. N. Y.: Cambridge University Press, 2001. 602 p.
9. *Gordon G. and McNulty J.* Matroids: a Geometric Introduction. Cambridge: Cambridge University Press, 2012. 410 p.
10. *Mason J. H.* Matroids as the study of geometrical configurations // Higher Combinatorics / ed. M. Aigner. Dordrecht: Reidel Publishing Company, 1977. P. 133–176.
11. *Bruhn H., Diestel R., Kriesell M., et al.* Axioms for infinite matroids // Adv. Math. 2013. V. 239. P. 18–46.
12. *Delucchi E.* Combinatorial Geometries in Algebra and Topology. Habilitationsschrift. Universität Bremen, 2011. 160 p.
13. *Gelfand I. M., Goresky R. M., McPherson R. D., and Serganova V.* Combinatorial geometries, convex polyhedra and Schubert cells // Adv. Math. 1987. V. 63. P. 301–316.
14. *Oxley J.* Matroid Theory. N. Y.: Oxford University Press, 1992. 532 p.
15. *Pitsoulis L. S.* Topics in Matroid Theory. N. Y.: Springer Verlag, 2014. 127 p.
16. Theory of Matroids. Encyclopedia of Mathematics and its Applications 26 / ed. N. White. Cambridge: Cambridge University Press, 1986. 316 p.
17. Combinatorial Geometries. Encyclopedia of Mathematics and its Applications 29 / ed. N. White. Cambridge: Cambridge University Press, 1987. 212 p.
18. Matroid Applications. Encyclopedia of Mathematics and its Applications 40 / ed. N. White. Cambridge: Cambridge University Press, 1992. 363 p.
19. *Mac Lane S.* Some interpretations of abstract linear dependence in terms of projective geometry // Amer. J. Math. 1936. V. 58. P. 236–240.

REFERENCES

1. *Whitney H.* On the abstract properties of linear dependence. Amer. J. Math., 1935, vol. 57, pp. 509–533.

2. *Brylawski T.* Appendix of matroid cryptomorphisms. Theory of Matroids. Encyclopedia of Mathematics and its Applications 26, ed. N. White. Cambridge, Cambridge University Press, 1986, pp. 298–312.
3. *Schrijver A.* Combinatorial Optimization, vol. B. Berlin, Heidelberg, N. Y., Springer Verlag, 2003. 1881 p.
4. *Welsh D. J. A.* Matroid Theory. London, N. Y., San Francisco, Academic Press, 1976. 433 p.
5. *Aigner M.* Combinatorial Theory. Berlin, Heidelberg, N. Y., Springer Verlag, 1979.
6. *Rybnikov K. A.* Vvedenie v kombinatornyy analiz [Introduction to Combinatorial Analysis]. Moscow, MSU Publ., 1985. (in Russian)
7. *Crapo H. H. and Rota G.-C.* On the Foundations of Combinatorial Theory. II. Combinatorial Geometries. Cambridge, MA, MIT Press, 1970. 293 p.
8. *Van Lint J. H. and Wilson R. M.* A Course in Combinatorics. N. Y., Cambridge University Press, 2001. 602 p.
9. *Gordon G. and McNulty J.* Matroids: a Geometric Introduction. Cambridge, Cambridge University Press, 2012. 410 p.
10. *Mason J. H.* Matroids as the study of geometrical configurations. Higher Combinatorics, ed. M. Aigner. Dordrecht, Reidel Publishing Company, 1977, pp. 133–176.
11. *Bruhn H., Diestel R., Kriesell M., et al.* Axioms for infinite matroids. Adv. Math., 2013, vol. 239, pp. 18–46.
12. *Delucchi E.* Combinatorial Geometries in Algebra and Topology. Habilitationsschrift. Universität Bremen, 2011. 160 p.
13. *Gelfand I. M., Goresky R. M., McPherson R. D., and Serganova V.* Combinatorial geometries, convex polyhedra and Schubert cells. Adv. Math., 1987, vol. 63, pp. 301–316.
14. *Oxley J.* Matroid Theory. N. Y., Oxford University Press, 1992. 532 p.
15. *Pitsoulis L. S.* Topics in Matroid Theory. N. Y., Springer Verlag, 2014. 127 p.
16. Theory of Matroids. Encyclopedia of Mathematics and its Applications 26, ed. N. White. Cambridge, Cambridge University Press, 1986. 316 p.
17. Combinatorial Geometries. Encyclopedia of Mathematics and its Applications 29, ed. N. White. Cambridge, Cambridge University Press, 1987. 212 p.
18. Matroid Applications. Encyclopedia of Mathematics and its Applications 40, ed. N. White. Cambridge, Cambridge University Press, 1992. 363 p.
19. *Mac Lane S.* Some interpretations of abstract linear dependence in terms of projective geometry. Amer. J. Math., 1936, vol. 58, pp. 236–240.

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 519.7

ОТ КРИПТОАНАЛИЗА ШИФРА К КРИПТОГРАФИЧЕСКОМУ СВОЙСТВУ БУЛЕВОЙ ФУНКЦИИ[1]

А. А. Городилова

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

Настоящий обзор посвящён описанию основных криптографических свойств булевых функций, таких, как высокая алгебраическая степень, уравновешенность и совершенная уравновешенность, лавинные характеристики, отсутствие линейных структур, корреляционная иммунность и устойчивость, высокая нелинейность, статистическая независимость, алгебраическая иммунность, уровень аффинности и k-нормальность, дифференциальная равномерность, разложимость в сумму специальных функций, мультипликативная сложность, высокие мощности линеаризационных множеств. Исследуются вопросы формирования данных свойств на основе атак на блочные и поточные шифры, использующих определённые уязвимости булевых функций, являющихся компонентами шифров; приводятся основные идеи данных атак. Кратко рассмотрены базовые теоретические результаты, полученные для каждого из свойств, и сформулированы открытые проблемы в данной области.

Ключевые слова: *булева функция, поточный шифр, блочный шифр, алгебраическая степень, уравновешенность, совершенная уравновешенность, лавинные характеристики, линейная структура, корреляционная иммунность, устойчивость, нелинейность, статистическая независимость, алгебраическая иммунность, уровень аффинности, k-нормальность, дифференциальная равномерность, пороговое разбиение, мультипликативная сложность, линеаризационное множество, линейная сложность, корреляционный криптоанализ, быстрая корреляционная атака, линейный криптоанализ, статистический аналог, алгебраический криптоанализ, дифференциальный криптоанализ, атаки по сторонним каналам, линеаризационная атака.*

DOI 10.17223/20710410/33/2

FROM CRYPTANALYSIS TO CRYPTOGRAPHIC PROPERTY OF A BOOLEAN FUNCTION

A. A. Gorodilova

Sobolev Institute of Mathematics, Novosibirsk, Russia

E-mail: gorodilova@math.nsc.ru

The survey is devoted to description of basic, but not all, cryptographic properties of Boolean functions: algebraic degree, balancedness and perfect balancedness, avalanche

[1]Работа поддержана грантом РФФИ № 15-07-01328.

characteristics, non-existence of linear structures, correlation immunity and resiliency, high nonlinearity, statistical independence, algebraic immunity, affinity level and k-normality, differential uniformity, threshold implementation, multiplicative complexity, high cardinality of linearization sets. The questions about these properties formation are studied based on the attacks on stream and block ciphers that exploit the vulnerabilities of Boolean functions used in ciphers as components. The ideas of such attacks are given. We briefly describe the basic theoretical results obtained for each of the properties and formulate open problems in this area.

Keywords: *Boolean function, stream cipher, block cipher, algebraic degree, balancedness, perfect balancedness, avalanche characteristics, linear structure, correlation immunity, resiliency, nonlinearity, statistical independence, algebraic immunity, affinity level, k-normality, differential uniformity, threshold implementation, multiplicative complexity, linearization set, linear complexity, correlation attack, fast correlation attack, linear cryptanalysis, statistical analogue, differential cryptanalysis, side-channel attacks, linearization attack.*

Введение

На протяжении полувека сформировалось достаточно много требований к булевым функциям, использующимся в системах шифрования. Функции, удовлетворяющие данным требованиям, стали называть «криптографическими булевыми функциями».

Основная задача работы — привести исходные связи между различными методами криптоанализа шифров и математическими требованиями, которые накладываются на булевы функции, используемые в этих шифрах, для противодействия этим атакам. Таблица отражает основное содержание настоящей работы. Для более основательного знакомства с теоретическими результатами в области различных криптографических свойств можно рекомендовать работы О. А. Логачева, А. А. Сальникова, С. В. Смышляева, В. В. Ященко [11], Г. П. Агибалова [1], И. А. Панкратовой [14], Ю. В. Таранникова [19], Н. Н. Токаревой [42], C. Carlet [24, 25], T. W. Cusick, P. Stanica [30], A. Braeken [23].

Рассматриваемые криптографические свойства и их назначение

№ п/п	Свойство	Цель
1	Высокая алгебраическая степень	Повышение линейной сложности генерируемой последовательности; повышение степени системы нелинейных уравнений, описывающих шифр
2	Уравновешенность	Улучшение статистических свойств последовательностей, вырабатываемых поточными генераторами
3	Совершенная уравновешенность	
4	Лавинные характеристики	Обеспечение изменения значений большого числа выходных переменных при изменении значений малого числа входных переменных
5	Отсутствие линейных структур	Улучшение нелинейных свойств функций
6	Корреляционная иммунность, устойчивость	Препятствие проведению корреляционной атаки на поточные шифры
7	Высокая нелинейность	Препятствие проведению быстрой корреляционной атаки на поточные шифры и линейного криптоанализа блочных шифров
8	Статистическая независимость	Препятствие проведению статистических методов криптоанализа шифров

Окончание таблицы

№ п/п	Свойство	Цель
9	Алгебраическая иммунность	Препятствие проведению алгебраического криптоанализа шифров
10	Уровень аффинности и k-нормальность	Препятствие методу линеаризации без введения новых переменных для решения булевых уравнений
11	Дифференциальная равномерность	Препятствие проведению дифференциального криптоанализа блочных шифров
12	Разложимость в сумму специальных функций	Маскирование данных с целью защиты от атак по сторонним каналам
13	Мультипликативная сложность	Минимизация размера и стоимости аппаратной реализации криптоалгоритмов
14	Высокие мощности линеаризационных множеств	Препятствие линеаризационной атаке

1. Основные определения и обозначения

1.1. Булевы функции

Введём обозначения: n — натуральное число; $\mathbb{F}_2$ — множество, состоящее из 0 и 1; $x = (x_1, \ldots, x_n)$ — двоичный вектор с координатами из $\mathbb{F}_2$; $\mathbb{F}_2^n$ — множество всех двоичных векторов длины n; $\mathbf{0} = (0, \ldots, 0)$ — нулевой вектор; $\oplus$ — сложение по модулю 2.

Весом Хэмминга $\mathrm{wt}(x)$ двоичного вектора x называется количество единиц, содержащихся в x: $\mathrm{wt}(x) = \sum\limits_{i=1}^{n} x_i$. *Расстоянием Хэмминга* $\mathrm{d}(x, y)$ между двумя векторами x, y называется число позиций, в которых они различаются, или, что эквивалентно, $\mathrm{d}(x, y) = \mathrm{wt}(x \oplus y)$. *Скалярное произведение* $\langle x, y\rangle$ двоичных векторов x, y определяется как $\langle x, y\rangle = x_1y_1 \oplus \ldots \oplus x_ny_n$.

Векторной булевой функцией ((n, m)-функцией) F называется произвольное отображение $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. В случае $m = 1$ говорят, что F — *булева функция* от n переменных. Можно рассматривать (n, m)-функцию как набор из m координатных булевых функций от n переменных: $F = (f_1, \ldots, f_m)$. *Компонентной функцией* называется любая ненулевая линейная комбинация координатных функций, т. е. булева функция $\langle b, F\rangle$, где $b \in \mathbb{F}_2^m$, $b \neq \mathbf{0}$.

Вес функции $\mathrm{wt}(f)$ равен мощности её носителя $\mathrm{supp}(f) = \{x \in \mathbb{F}_2^n : f(x) = 1\}$. *Расстоянием Хэмминга* $\mathrm{d}(f, g)$ между булевыми функциями f и g является расстояние Хэмминга между векторами их значений: $\mathrm{d}(f, g) = |\{x \in \mathbb{F}_2^n : f(x) \neq g(x)\}|$. Пусть $\mathcal{M}_n$ — некоторое множество булевых функций от n переменных. Расстояние от функции g до множества функций $\mathcal{M}_n$ определяется как $\mathrm{d}(g, \mathcal{M}_n) = \min\{d(f, g) : f \in \mathcal{M}_n\}$. Для любой булевой функции f *производная по направлению* a, где $a \in \mathbb{F}_2^n$, определяется следующим образом: $D_a f(x) = f(x) \oplus f(x \oplus a)$.

Любую (n, m)-функцию можно единственным образом записать в виде *полинома Жегалкина*, или *алгебраической нормальной формы* (АНФ): $F(x_1, \ldots, x_n) = \bigoplus\limits_{k=0}^{n} \bigoplus\limits_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} x_{i_1} \ldots x_{i_k} \oplus a_0$, где $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ и $a_{i_1,\ldots,i_k} \in \mathbb{F}_2^m$.

Алгебраической степенью $\deg(F)$ функции F называется количество переменных в самом длинном слагаемом АНФ, при котором коэффициент не равен нулевому вектору. Функция степени не выше 1 называется *аффинной*. В случае $a_0 = \mathbf{0}$ функция *линейна*.

Для каждого $y \in \mathbb{F}_2^n$ *коэффициентом Уолша — Адамара* $W_f(y)$ булевой функции f от n переменных называется величина, определяемая равенством $W_f(y) =$

$= \sum\limits_{x\in\mathbb{F}_2^n}(-1)^{f(x)\oplus\langle x,y\rangle}$. Набор коэффициентов $W_f(y)$ по всем $y\in\mathbb{F}_2^n$ называется *спектром Уолша — Адамара* булевой функции f. Справедливо *равенство Парсеваля*: $\sum\limits_{y\in\mathbb{F}_2^n} W_f^2(y) = 2^{2n}$. Спектр Уолша — Адамара векторной функции состоит из всех коэффициентов Уолша — Адамара всех её компонентных булевых функций: $W_F(u,v) = \sum\limits_{x\in\mathbb{F}_2^n}(-1)^{\langle v,F(x)\rangle\oplus\langle u,x\rangle}$.

1.2. Блочные шифры

Блочный шифр преобразует блок открытого текста (сообщения) длины N в блок шифртекста также длины N с использованием некоторого секретного ключа. Процесс шифрования состоит из нескольких повторяющихся раундов, определяющихся, как правило, одинаковой раундовой функцией, зависящей от раундового подключа K_i, вырабатываемого по специальному правилу из исходного ключа K.

Наиболее распространённые типы блочных шифров, *SP-сеть* и *сеть Фейстеля*, схематично приведены на рис. 1 [20]. Обе эти схемы отражают в себе два принципа построения шифрующих преобразований, которые определил в своей работе Клод Шеннон, — *рассеивание* и *перемешивание*. Приведём объяснение этих принципов по работе [5]: «Качественно можно сказать, что перемешивание усложняет восстановление взаимосвязи статистических и аналитических свойств открытого и шифрованного текстов, а рассеивание распространяет влияние одного знака открытого текста на большое число знаков шифртекста, что позволяет сгладить влияние статистических свойств открытого текста на свойства шифртекста». На примере SP-сети хорошо видно, что P-блок обеспечивает рассеивание, а набор небольших S-блоков — перемешивание. В то время как в качестве P-блока обычно выбирается линейная функция, S-блоки составляют нелинейные преобразования шифра.

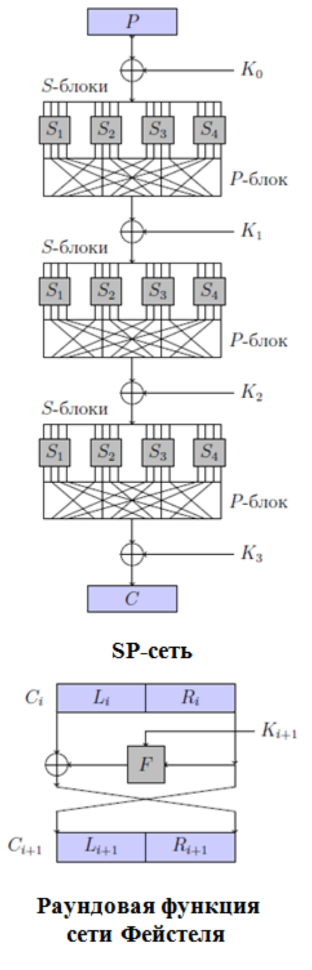


Рис. 1

По сути, S-блок — это векторная (n,m)-функция, причём значения n и m небольшие, например 4, 6, 8 битов. Несмотря на столь небольшой размер, найти такой S-блок с «хорошими» криптографическими свойствами достаточно трудно. Для наглядности заметим, что всевозможных отображений из $\mathbb{F}_2^8$ в себя существует 2^{2048}, что в настоящее время не поддаётся полному перебору! При этом даже аналитические рассуждения пока не могут привести к ответу на некоторые важные для криптографических приложений вопросы. Например, не известно, существуют ли взаимно однозначные почти совершенно нелинейные отображения из $\mathbb{F}_2^n$ в себя при чётных $n \geqslant 8$, подробнее об этом сказано в п. 12.

1.3. Поточные шифры

Приведём широко используемую модель поточного шифра — шифр гаммирования [5]. В основе таких систем лежит метод «наложения» (например, сложение по модулю 2) ключевой последовательности (гаммы) на открытый текст (шифруемое сообщение). Из работ Клода Шеннона следует, что если ключ имеет такую же длину, как и сообщение, выбирается случайно и равновероятно и при этом используется только один раз, то данная система шифрования является абсолютно стойкой к атакам на основе шифртекста. Поскольку такая модель неприменима для широкого распространения в силу того, что затруднительно генерировать такой же объём ключа, как и сообщения, а тем более его передавать, перед разработчиками стоит задача получать из короткой случайной последовательности битов ключа некоторую длинную последовательность (гамму), которая будет близка к случайной. Заметим, что именно от свойств данной последовательности зависит криптографическая стойкость шифра.

Часто в качестве компонент поточного шифра используются регистры сдвига с обратной связью. Общая схема их работы приведена на рис. 2, где f — булева функция от n переменных, являющаяся *функцией обратной связи* [20].

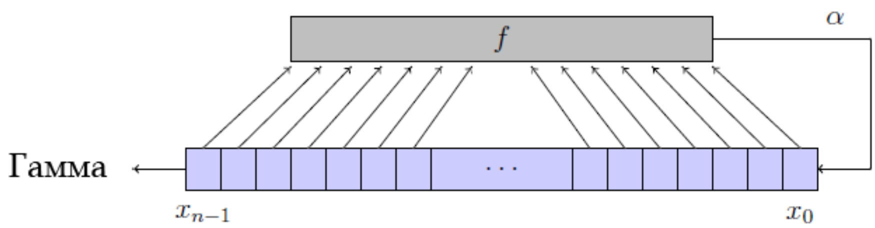


Рис. 2

Перед стартом работы происходит начальное заполнение состояния регистра некоторыми n битами. Далее на каждом такте работы вычисляется очередное значение $\alpha = f(x_{n-1}, \ldots, x_0)$, затем все биты регистра сдвигаются влево, при этом в крайний правый бит записывается значение α, а крайний левый бит становится очередным битом выходной последовательности $u = \{u_0, u_1, u_2 \ldots\}$.

Наибольшее распространение получили регистры сдвига с линейной обратной связью (LFSR), т. е. те, в которых f линейна, скажем, $f(x_{n-1}, \ldots, x_0) = \langle c, x\rangle$, где $c \in \mathbb{F}_2^n$. Заметим, что последовательность, порождаемая любым регистром с обратной связью, всегда периодическая. Легко видеть, что на самом деле любую периодическую последовательность можно породить LFSR подходящей длины. При этом *линейной сложностью* $\mathcal{L}^u$ последовательности u называют минимальную длину LFSR, который её порождает. Линейная сложность последовательности — это основной параметр, характеризующий сложность её аналитического строения [5].

Напомним, что для генерации «хорошей» гаммы не используется лишь один LFSR. Действительно, если мы знаем функцию обратной связи $f(x) = \langle c, x\rangle$, $c \in \mathbb{F}_2^n$, то достаточно лишь n подряд идущих битов последовательности для того, чтобы восстановить начальное состояние регистра путём решения системы линейных уравнений. А начальное состояние, как правило, и является секретным ключом шифра.

Кроме того, известен более сильный результат, который позволяет найти линейную функцию обратной связи для любой периодической последовательности. Допустим, что мы перехватили достаточно длинный отрезок некоторой периодической последовательности. Тогда с помощью широко известного алгоритма Берлекэмпа — Месси можно за полиномиальное от длины конечной последовательности время найти закон рекурсии, её порождающий, что эквивалентно нахождению линейного регистра, который вырабатывает данный отрезок последовательности. При этом если длина последовательности не меньше $2\mathcal{L}$, где $\mathcal{L}$ — её линейная сложность, то найденный LFSR вырабатывает и всю бесконечную последовательность, отрезок которой нам известен.

С другой стороны, линейная сложность вырабатываемой последовательности должна быть высокой при почти любом начальном состоянии регистра — неизвестном ключе. В силу этого используют некоторые усложнения. Выделяют две основные модели генераторов, построенных на основе регистров сдвига с линейной обратной связью [20] (рис. 3). Булева функция h называется соответственно *комбинирующей* и *фильтрующей*.

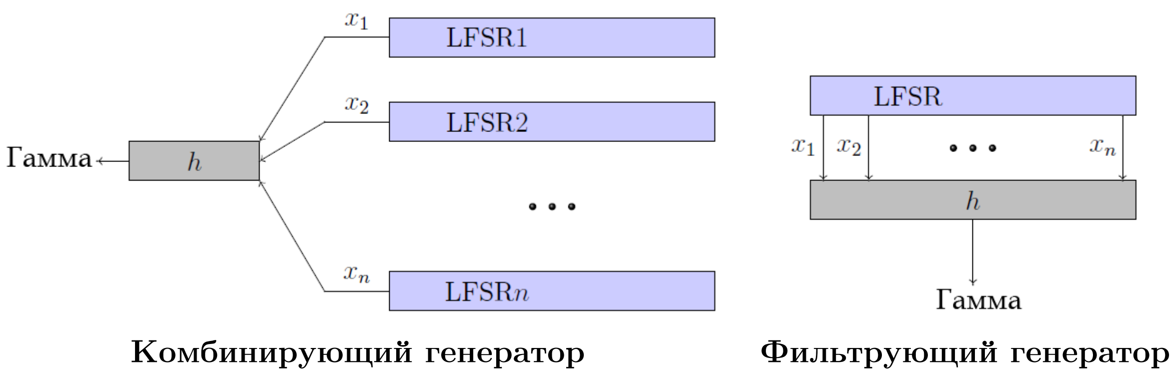


Рис. 3

2. Высокая алгебраическая степень

Опишем, опираясь на обзор C. Carlet [24], теоретические результаты, которые показывают, что в качестве комбинирующей и фильтрующей функции h следует выбирать те, чья алгебраическая степень не мала. Рассмотрим сначала комбинирующий генератор, состоящий из n регистров с линейной обратной связью длин $L_1, \ldots, L_n$. Пусть комбинирующая функция h задана в виде алгебраической нормальной формы $h(x_1, \ldots, x_n) = \bigoplus\limits_{k=1}^{n} \bigoplus\limits_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} x_{i_1} \ldots x_{i_k}$, где $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ и $a_{i_1,\ldots,i_k} \in \mathbb{F}_2$. Тогда линейная сложность $\mathcal{L}$ вырабатываемой последовательности оценивается сверху через длины регистров следующим образом:

$$\mathcal{L} \leqslant \sum_{k=1}^{n} \sum_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} L_{i_1} \ldots L_{i_k},$$

при этом известны условия, при которых данная оценка достигается: если L_i совпадает с линейной сложностью генерируемой LFSR$_i$ последовательности для любого i и числа $L_1, \ldots, L_n$ попарно взаимно просты. Можно видеть, что чем выше степень АНФ, тем большую линейную сложность можно получить.

В случае фильтрующей модели аналогичного точного результата нет, хотя также известна оценка линейной сложности $\mathcal{L}$ генерируемой последовательности через степень фильтрующей функции $\deg(h)$, а именно: $\mathcal{L} \leqslant \sum\limits_{i=0}^{\deg(h)} C_L^i$, где L — длина регистра, а C_L^i — биномиальный коэффициент. Кроме того, если L — простое число, то верна оценка снизу: $\mathcal{L} \geqslant C_L^{\deg(h)}$.

В блочных шифрах следует также выбирать функции с достаточно большой степенью. «Параметр $\deg(F)$ булевой функции должен быть большим. Для блочных шифров это условие накладывается, как правило, для того, чтобы система уравнений на биты ключа, построенная путём анализа структуры шифра — в том числе функции F, использующейся в качестве его компоненты, — имела высокую степень. Чем выше степень системы, тем сложнее её решить, а значит, определить ключ» [20].

3. Уравновешенность

Определение 1. Булева функция f от n переменных называется *уравновешенной*, если её вес равен 2^{n-1}, т. е. функция принимает значения 0 и 1 одинаково часто.

Это, пожалуй, одно из самых естественных необходимых свойств, накладываемых на булевы функции, использующиеся в поточных шифрах. Если булева функция уравновешена, то вероятность того, что она примет значение 0 или 1, одинакова и равна 1/2. Это позволяет ослабить статистические зависимости между входом функции и её выходом. В противном случае у криптоаналитика есть возможность, используя вероятностное соотношение, провести криптоанализ шифра.

Данное определение обобщается на векторный случай.

Определение 2. Векторная (n, m)-функция F называется *уравновешенной*, если $|F^{-1}(y)| = |\{x \in \mathbb{F}_2^n : F(x) = y\}| = 2^{n-m}$ для любого $y \in \mathbb{F}_2^m$.

При этом справедливо следующее

Утверждение 1. Векторная (n, m)-функция F уравновешена тогда и только тогда, когда уравновешены все её компонентные функции $\langle v, F\rangle$, $v \in \mathbb{F}_2^m$, $v \neq \mathbf{0}$.

Заметим, что при $n = m$ класс уравновешенных векторных функций совпадает с классом взаимно однозначных функций. Как правило, именно они представляют наибольший интерес для использования в блочных шифрах в качестве S-блоков для обеспечения однозначного расшифрования.

4. Совершенная уравновешенность

Свойство совершенной уравновешенности булевой функции является естественным обобщением обычной уравновешенности, когда данная функция выступает, например, в качестве фильтрующей функции генератора. Данное свойство было формализовано С. Н. Сумароковым [18] .

Пусть f — фильтрующая функция генератора от n переменных, $x = (x_1, \ldots, x_{\ell+n-1})$ — отрезок входной последовательности длины $\ell + n - 1$, где ℓ — некоторое натуральное число. Тогда генератор выработает по нему отрезок последовательности $u = (u_1, \ldots, u_\ell)$ длины ℓ, где $u_i = f(x_i, x_{i+1}, \ldots, x_{i+n-1})$, $i = 1, \ldots, \ell$. Определим для функции f и числа ℓ векторную $(\ell + n - 1, \ell)$-функцию f_ℓ, сопоставляющую вектору x вектор u по описанному выше правилу.

Определение 3. Булева функция f называется *совершенно уравновешенной*, если для любого натурального числа ℓ функция f_ℓ уравновешена.

В частности, если функция совершенно уравновешена, то она уравновешена и в обычном смысле. Обратное неверно.

Запретом булевой функции f называется такой вектор $u = (u_1, \ldots, u_\ell)$ для некоторого ℓ, для которого множество прообразов $f_\ell^{-1}(u)$ пусто. Следующая теорема отражает критерий совершенной уравновешенности в терминах запретов функции.

Теорема 1. Булева функция совершенно уравновешена тогда и только тогда, когда она является функцией без запрета.

Интуитивно понятно, что наличие запрета у фильтрующей функции генератора делает её «слабее» с точки зрения порождения последовательностей с хорошими статистическими свойствами. Однако следует быть осторожными, поскольку совершенно уравновешенная фильтрующая функция в том или ином виде переносит свойства входной последовательности в свойства генерируемой последовательности [11]. Например, С. В. Смышляевым в работе [17] установлен новый критерий, который идейно говорит следующее: «фильтрующая функция сохраняет запреты (в соответствующем смысле) тогда и только тогда, когда она совершенно уравновешена». Соответственно если на вход функции поступает «далёкая» от случайной последовательность, то и на выходе её статистические свойства будут плохие.

Заметим, что если фильтрующая функция линейна по своей первой и/или последней существенной переменной, то она совершенно уравновешена. Но в обратную сторону это неверно, поскольку найдены конструкции совершенно уравновешенных функций, нелинейно зависящих от своих крайних переменных (ссылки можно найти в [11]). Актуальность поиска широкого класса таких функций подтверждается тем, что известна так называемая инверсионная атака на фильтрующие генераторы, использующие линейные по первой или последней существенной переменной функции в качестве фильтрующих. Данную атаку предложил J. Dj. Golić [34].

5. Лавинные характеристики

Концепция лавинных характеристик булевой функции отражает один из принципов Шеннона построения шифрующих преобразований, сформулированных в п. 1.2, а именно принцип рассеивания. Следующее определение ввели A. F. Webster, S. E. Tavares в работе [43].

Определение 4. Булева функция f от n переменных удовлетворяет *строгому лавинному критерию* (SAC), если для любого направления $a \in \mathbb{F}_2^n$, где $\mathrm{wt}(a) = 1$, производная $D_a(f)$ уравновешена.

Если все координатные функции векторной (n, m)-функции удовлетворяют SAC, то при изменении одного входного бита с вероятностью $1/2$ изменится каждый из выходных битов. Следовательно, можно ожидать, что примерно половина выходных битов изменится.

Обобщением данного критерия является следующий, который стали рассматривать B. Preneel и др. [39].

Определение 5. Булева функция f от n переменных удовлетворяет *критерию распространения степени* k (PC(k)), если для любого направления $a \in \mathbb{F}_2^n$, где $1 \leqslant \mathrm{wt}(a) \leqslant k$, производная $D_a(f)$ уравновешена.

По определению PC(1) совпадает с SAC. Забегая вперёд, можно отметить, что функции, удовлетворяющие PC(n), — это в точности бент-функции (см. определение 11, п. 8 и теорему 16, п. 12). Если функция удовлетворяет данным критериям, то это означает, что изменение входного вектора в нескольких битах меняет значение функции с вероятностью $1/2$.

Как отмечается в [11], строгий лавинный критерий и его обобщения «явились в конечном счёте индикаторами локальных свойств для исследуемых криптографических функций». Более правильно требовать, чтобы в среднем у функции были «хорошие» лавинные характеристики, которые выражаются в том, что модуль функции автокорреляции $\Delta_f(a) = \sum\limits_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus f(x \oplus a)}$ был равен или близок к нулю для большинства векторов $a \in \mathbb{F}_2^n$. Такой подход предложили X.-M. Zhang и Y. Zheng в работе [45].

Определение 6. *Глобальными лавинными характеристиками* (GAC) булевой функции f от n переменных называются числа $\sigma_f = \sum\limits_{a \in \mathbb{F}_2^n} \Delta_f^2(a)$ и $\Delta_f = \max\limits_{a \in \mathbb{F}_2^n, a \neq \mathbf{0}} \Delta_f(a)$.

Понятно, что чем меньше данные величины, тем лучше функция для использования в шифре, поскольку GAC отражают лавинные показатели в среднем. В [11] можно найти основные свойства GAC произвольной функции, а также их некоторые связи с другими криптографическими свойствами, в частности с нелинейностью и порядком устойчивости.

6. Линейные структуры

Определение 7. Векторная (n, m)-функция *обладает линейной структурой*, если существует вектор $a \in \mathbb{F}_2^n$, $a \neq \mathbf{0}$, такой, что $D_a(F) \equiv \mathrm{const}$.

В качестве компонент шифра следует выбирать функции, которые не обладают линейной структурой [32], несмотря на то, что, как отмечается в [24], к настоящему моменту данные слабости не были использованы в атаках. Действительно, наличие линейной структуры у функции свидетельствует о её «похожести» на линейную функцию в том смысле, что она линейно эквивалентна функции, у которой есть переменная, от которой она зависит линейно или фиктивно. Как уже отмечалось, близость функций в различных смыслах к линейным неприемлема для использования в криптографических системах.

7. Корреляционная иммунность и устойчивость

Рассматриваемые здесь свойства возникли из разных прикладных задач, но, как оказалось, тесно связаны. Термин корреляционной иммунности ввёл T. Siegenthaler в работе [40]. Он показал, что функции с высоким порядком корреляционной иммунности, используемые в качестве комбинирующей функции генератора в поточном шифре, делают шифр стойким к корреляционной атаке. Суть атаки состоит в поиске подмножества переменных комбинирующей функции, о значениях которых можно получить информацию, зная значение функции. Устойчивые функции были введены в открытой литературе также в 1980-х годах, но, как отмечается в [11], «были связаны с такими областями исследований, как распределённые вычисления, устойчивые относительно ошибок, и выработка общих ключей для квантово-криптографических каналов связи». Из работы К. Н. Панкова [13] известно, что в СССР аналогичные функции исследовались Л. В. Ларионовым в 1970-х годах.

В качестве комбинирующей функции генератора в поточном шифре необходимо использовать функции, обладающие свойством устойчивости. При этом чем выше порядок устойчивости, тем выше стойкость шифра к корреляционному криптоанализу.

Приведём формальные определения данных свойств на языке комбинаторики. Определим сначала понятие подфункции. *Подфункцией* булевой функции f от переменных $x_1, \ldots, x_n$ называется булева функция, полученная из f подстановкой вместо переменных $x_{i_1}, \ldots, x_{i_k}$ конкретных констант $a_1, \ldots, a_k$, принимающих значения 0 или 1. Такая подфункция обозначается $f_{i_1,\ldots,i_k}^{a_1,\ldots,a_k}$.

Определение 8. Булева функция f называется *корреляционно-иммунной порядка k*, если вес подфункций $f_{i_1,\ldots,i_k}^{a_1,\ldots,a_k}$ удовлетворяет соотношению $\mathrm{wt}(f_{i_1,\ldots,i_k}^{a_1,\ldots,a_k}) = \mathrm{wt}(f)/2^k$ для любого набора индексов $1 \leqslant i_1 < \ldots < i_k \leqslant n$ и любых значений $a_1, \ldots, a_k \in \mathbb{F}_2$.

Другими словами, булева функция f называется корреляционно-иммунной порядка k, если $\mathsf{P}[f = 1] = \mathsf{P}[f_{i_1,\ldots,i_k}^{a_1,\ldots,a_k} = 1]$, где P — функция вероятности, т. е. знание некоторых входных битов не даёт статистической информации о значении функции.

Определение 9. Булева функция f называется *k-устойчивой* (*k-эластичной*), если любая её подфункция, полученная фиксацией не более k переменных, является уравновешенной.

Нетрудно убедиться, что булева функция f является k-устойчивой тогда и только тогда, когда она уравновешена и корреляционно-иммунна порядка k.

7.1. Идея корреляционной атаки

Рассмотрим общую идею данного криптоанализа, следуя [14].

Пусть f — комбинирующая функция генератора, LFSR_1, …, LFSR_n — его регистры сдвига с линейной обратной связью длин $L_1, \ldots, L_n$ соответственно, а $u = u_0, u_1, u_2, \ldots$ — выходная последовательность регистра. Трудоёмкость криптоанализа «грубой силой», т. е. полного перебора всех начальных состояний регистров, оценивается как $2^{L_1+\ldots+L_n}$. Если регистр построен «правильно», то последовательность u очень близка к случайной, поэтому можно считать, что $\mathsf{P}[u_i = 0] \approx 1/2$. Следовательно, если $z = z_0, z_1, z_2, \ldots$ — произвольная не зависящая от u последовательность, то можно считать, что $\mathsf{P}[u_i = z_i] \approx 1/2$, поскольку $\mathsf{P}[u_i = z_i] = \mathsf{P}[u_i=0]\,\mathsf{P}[z_i=0] + \mathsf{P}[u_i = 1]\,\mathsf{P}[z_i = 1] \approx 1/2\,(\mathsf{P}[z_i = 0] + \mathsf{P}[z_i = 1]) = 1/2$.

Предположим, что функция f *коррелирует* с функцией $\ell(x_1, \ldots, x_n) = x_1$, что означает $\mathsf{P}[f = \ell] = 1/2 + \varepsilon \neq 1/2$. Тогда утверждается, что можно восстановить неизвестное начальное состояние регистра LFSR_1. Для этого будем перебирать все возможные начальные состояния первого регистра (их 2^{L_1}), для каждого из них генерировать выходную последовательность данного регистра $z = z_0, z_1, z_2, \ldots$ и считать, сколько раз выполнено $z_i = u_i$. Тогда если начальное состояние было предположено неправильно, то $\mathsf{P}[z_i = u_i] \approx 1/2$, а если правильно, то $\mathsf{P}[z_i = u_i] \approx 1/2 + \varepsilon$. Таким образом, чем больше значение корреляции $|\varepsilon|$, тем с большей вероятностью мы найдём правильное состояние регистра. Тем самым мы уменьшили сложность перебора до $2^{L_1} + 2^{L_2+\ldots+L_n}$. Если при этом есть корреляция f и других переменных, то сложность можно ещё понижать. Если же у f нет корреляции с функциями $\ell(x) = x_i$, можно искать корреляции с другими линейными функциями $\langle c, x\rangle$, у которых $\mathrm{wt}(c) = k$ мал, скажем, $c = (1, \ldots, 1, 0, \ldots, 0)$. Тогда сложность перебора уменьшается до $2^{L_1+\ldots+L_k}+2^{L_{k+1}+\ldots+L_n}$. Но если k достаточно большое, то особого выигрыша для криптоаналитика может и не быть.

Заметим, что в [11] приводятся также известные результаты о том, что фильтрующий генератор с функцией f можно свести к специально построенному комбинирующему генератору с той же функцией f, выступающей уже в качестве комбинирующей. При этом новый генератор вырабатывает ту же последовательность при определённом начальном заполнении состояний его регистров. Следовательно, корреляционную атаку можно обобщать и на случай фильтрующих генераторов.

7.2. Основные теоремы и связь с нелинейностью

Утверждение 2. Корреляционно-иммунная порядка k булева функция является также корреляционно-иммунной порядка ℓ для всех $\ell < k$.

В силу этого утверждения естественно ввести определение *порядка* корреляционной иммунности $\mathrm{cor}(f)$ функции f как

$$\mathrm{cor}(f) = \max\{0 \leqslant k \leqslant n : f \text{ — корреляционно-иммунная порядка } k\}.$$

Следующая широко известная теорема даёт спектральную характеризацию корреляционно-иммунных и устойчивых функций.

Теорема 2 (спектральная характеризация). Пусть f — булева функция от n переменных. Справедливо $\mathrm{cor}(f) = k$ тогда и только тогда, когда $W_f(y) = 0$ для всех векторов y, таких, что $1 \leqslant \mathrm{wt}(y) \leqslant k$. Кроме того, f является уравновешенной тогда и только тогда, когда $W_f(\mathbf{0}) = 0$.

Данная теорема наглядно связывает определение корреляционно-иммунных порядка k функций и способность противостоять корреляционной атаке при использовании их в качестве комбинирующих функций поточного генератора. Действительно, для проведения атаки необходимо найти линейную функцию $\ell(x) = \langle c, x\rangle$, с которой есть корреляция у комбинирующей функции f, т. е. $\mathsf{P}[f = \ell] \neq 1/2$. Так как $\mathsf{P}[f = \ell] = (2^n - \mathrm{d}(f, \ell))/2^n = 1/2 + (2^{n-1} - \mathrm{d}(f, \ell))/2^n \neq 1/2$, это эквивалентно тому, что $\mathrm{d}(f, \ell) \neq 2^{n-1}$. Кроме того, легко получить, что расстояние между f и линейной функцией $\ell(x) = \langle c, x\rangle$ выражается как $\mathrm{d}(f, \ell) = 2^{n-1} - W_f(c)/2$. Таким образом, $\mathrm{d}(f, \ell) \neq 2^{n-1}$ тогда и только тогда, когда $W_f(c) \neq 0$. Следовательно, если порядок устойчивости комбинирующей функции достаточно высокий, то корреляционную атаку на данный шифр провести будет сложно.

Связь порядка $\mathrm{cor}(f)$ и степени функции $\deg(f)$ отражается в следующей теореме.

Теорема 3 (Siegenthaler). Пусть f — булева функция от n переменных.

1. Если $\mathrm{cor}(f) = k$, то выполняется $\deg(f) + k \leqslant n$.
2. Если $\mathrm{cor}(f) = k$, f уравновешена и $k \leqslant n - 2$, то выполняется $\deg(f) + k \leqslant n - 1$.

Из теоремы следует, что чем выше степень функции, тем меньше порядок её корреляционной иммунности, и наоборот. Но, как мы видели, оба этих параметра должны быть высокими для функций усложнения криптографических генераторов.

Следствие 1. Пусть f — булева функция от n переменных. Если $\mathrm{cor}(f) = n$, то $f \equiv \mathrm{const}$. Если $\mathrm{cor}(f) = n - 1$, то $f(x) = x_1 \oplus \ldots \oplus x_n \oplus \mathrm{const}$.

Известна следующая оценка $\mathrm{cor}(f)$, полученная Д. Г. Фон-дер-Флаассом [33].

Теорема 4 (Фон-дер-Флаасс). Пусть f — неуравновешенная булева функция от n переменных. Тогда $\mathrm{cor}(f) \leqslant (2n/3) - 1$.

Здесь естественно также упомянуть про нелинейность N_f (см. определение 10, п. 8) корреляционно-иммунных функций.

Теорема 5 (связь $\mathrm{cor}(f)$ и N_f). Пусть f — булева функция от n переменных.

1. Если $\mathrm{cor}(f) = k$, $k \leqslant n - 1$, то выполняется $N_f \leqslant 2^{n-1} - 2^k$.
2. Если $\mathrm{cor}(f) = k$, f уравновешена и $k \leqslant n - 2$, то выполняется $N_f \leqslant 2^{n-1} - 2^{k+1}$.

Как видно из теоремы, нелинейность функции с ростом порядка устойчивости падает. Это интуитивно понятно из того, что с ростом $\mathrm{cor}(f)$ становится всё больше нулевых коэффициентов Уолша — Адамара функции f, что ведёт к увеличению максимального значения $|W_f(a)|$ в силу равенства Парсеваля, а следовательно, к снижению нелинейности. При этом интересен и актуален вопрос о достижимости оценок из теоремы 5. Известно, что если оценка для k-устойчивых функций достигается, то $(n-3)/2 \leqslant k \leqslant n - 2$, но примеров для всех таких возможных параметров пока не найдено. Ю. В. Таранниковым [41] разработан и обобщён метод, который в настоящий момент позволяет строить k-устойчивые функции с нелинейностью $2^{n-1} - 2^{k+1}$ для всех $k \geqslant cn(1 + \mathrm{o}(1))$, где $c = 0{,}5789\ldots$

8. Высокая нелинейность

Определение 10. *Нелинейностью* булевой функции f от n переменных называется величина N_f, равная расстоянию Хэмминга от f до множества $\mathcal{A}_n$ всех аффинных функций от n переменных.

В п. 7.2 мы уже упоминали связь расстояния между произвольной функцией и линейной функцией $\mathrm{d}(f, \langle c, x\rangle) = 2^{n-1} - W_f(c)/2$. Основываясь на ней, легко получить, что нелинейность f выражается через её коэффициенты Уолша — Адамара следующим образом: $N_f = \mathrm{d}(f, \mathcal{A}_n) = 2^{n-1} - \max\limits_{c \in \mathbb{F}_2^n} |W_f(c)|/2$. Более того, из равенства Парсеваля можно найти оценку снизу: $\max\limits_{c\in\mathbb{F}_2^n} |W_f(c)| \geqslant 2^{n/2}$. Таким образом, нелинейность функции всегда удовлетворяет неравенству $N_f \leqslant 2^{n-1} - 2^{n/2-1}$.

Определение 11. *Максимально нелинейной* называется функция, нелинейность которой достигает максимально возможного значения. В случае чётного числа переменных максимально нелинейные функции также называются *бент-функциями*.

Вопрос о том, кто первым начал изучение максимально нелинейных функций, остаётся открытым. Признанный авторитет в ответе на этот вопрос имеет Oscar S. Rothaus. В 1960-х годах он работал математиком в Институте оборонного анализа США и в то же время написал свою первую работу о бент-функциях, которая появилась в открытой печати лишь в 1976 г. Однако с недавнего времени стало известно [8], что бент-функции также изучались в Советском Союзе в 1960-х годах. Среди первых исследователей — В. А. Елисеев и О. П. Степченков, но их работы по-прежнему засекречены. Известно, что они называли бент-функции *минимальными функциями* и получили ряд утверждений о их свойствах, а также предложили аналог известной конструкции Майорана — МакФарланда. Подробно с историей изучения максимально нелинейных функций, их связи с различными комбинаторными объектами, применении в криптографии, известными конструкциями и открытыми вопросами можно познакомиться по работе Н. Н. Токаревой [42], посвящённой бент-функциям.

Чем выше значение нелинейности булевой функции, тем предпочтительнее её использовать как в поточных, так и в блочных шифрах. Приведём далее две атаки на различные виды шифров.

8.1. Идея быстрой корреляционной атаки

Данный вид атаки на комбинирующий генератор появился вскоре после простой корреляционной атаки (рассмотренной в п. 7.1). Опишем её основную идею, не вдаваясь в подробности теории кодов, исправляющих ошибки [24].

Будем оперировать с тем же комбинирующим генератором, что был описан в п. 7.1. Как и для корреляционной атаки, для быстрой корреляционной атаки необходимо найти линейную функцию $\ell(x) = \langle c, x\rangle$, с которой у f есть корреляция, т. е. $\mathsf{P}[f = \ell] = 1/2 + \varepsilon \neq 1/2$. Отличие в том, что нам теперь не важно, каково значение $\mathrm{wt}(c)$, а важно лишь, чтобы значение корреляции $|\varepsilon|$ было как можно больше.

Будем считать далее, что $\varepsilon > 0$ (иначе вместо ℓ рассмотрим функцию $\ell \oplus 1$). Пусть $u = u_0, u_1, u_2, \ldots$ — последовательность, вырабатываемая генератором. Тогда можно представить, что эта «правильная» последовательность (т. е. которую мы можем действительно наблюдать) является результатом внесения помех в «неправильную» последовательность $z = z_0, z_1, z_2, \ldots$ с вероятностью ошибки $1/2 - \varepsilon$, где z получена тем же генератором, но с комбинирующей функцией ℓ вместо f. Так как мы выбрали ε достаточно большим, вероятность ошибки будет маленькой.

Допустим, мы можем наблюдать отрезок последовательности $u_k, \ldots, u_{k+N-1}$. Множество всевозможных значений $z_k, \ldots, z_{k+N-1}$ является линейным кодом длины N. Тогда, наблюдая фрагмент последовательности u, путём исправления ошибок можно восстановить последовательность z. А это даёт выигрыш в том, что линейная сложность z гораздо ниже линейной сложности u, и достаточно отрезка последовательности гораздо меньшей длины, чтобы восстановить закон рекурсии и начальное состояние регистра с помощью алгоритма Берлекэмпа — Месси.

Отметим здесь сразу, что значение корреляции ε функции f с аффинными функциями можно оценить снизу через её нелинейность: $|\varepsilon| \leqslant 2^n - 2N_f$. Следовательно, чем выше нелинейность, тем меньше максимальная корреляция, а значит, больше вероятность ошибки в моделируемом канале с шумом, что делает менее возможным проведение быстрой корреляционной атаки на комбинирующий генератор с комбинирующей функцией f.

8.2. Идея линейного криптоанализа

В 1993 г. японский криптограф M. Matsui предложил статистический метод анализа шифра DES, названный *линейным криптоанализом.* Опишем идею самой простой его модификации по работе [20] (алгоритм 1).

Пусть P, C, K — блоки открытого текста, шифртекста и ключа некоторого блочного шифра. *Линейным приближением* шифра называется соотношение $\langle \alpha, P\rangle \oplus \langle \beta, C\rangle = \langle \gamma, K\rangle$, выполняющее с некоторой вероятностью $1/2+\varepsilon$, $\varepsilon \neq 0$, где α, β, γ — некоторые двоичные вектора соответствующих длин.

Алгоритм 1. Линейный криптоанализ

1: Находим линейное приближение шифра, для которого $|\varepsilon|$ как можно больше.
2: При фиксированном неизвестном ключе K набираем выборку из N пар (P, C).
3: Для каждой пары выборки вычисляем значение левой части выбранного на шаге 1 соотношения. Пусть N_0 — количество полученных нулей, а N_1 — единиц, $N_0 + N_1{=}N$.
4: Полагаем $\langle \gamma, K\rangle = 0$, если $(N_0 - N_1)\varepsilon > 0$, и $\langle \gamma, K\rangle = 1$ иначе.

В результате находим одно линейное соотношение на биты неизвестного ключа, следовательно, можем сократить полный перебор с 2^k до 2^{k-1}, где k — количество бит ключа K. Существуют более сильные модификации линейного криптоанализа, которые позволяют находить сразу группу неизвестных битов ключа, но суть остаётся прежней — поиск линейного приближения, но уже не всего шифра, а его части.

Основная сложность метода в том, как находить линейное приближение. На практике поступают так: анализируют соотношения, которые выполняются для S-блоков, а затем расширяют их на несколько раундов и на большую часть битов P, C, K.

Пусть S-блок шифра задан векторной (n, m)-функцией F. Требуется найти соотношение вида $\langle a, x\rangle \oplus \langle b, F(x)\rangle = 0$, выполняющееся с некоторой вероятностью $p = 1/2+\varepsilon$, где $\varepsilon \neq 0$. Распишем $p = \mathsf{P}[\langle a, x\rangle = \langle b, F(x)\rangle] = 1/2 + \left(2^{n-1} - \mathrm{d}(\langle a, x\rangle, \langle b, F(x)\rangle)\right)/2^n$. Опять, как и для поточного шифра, для того чтобы успешно провести атаку, т. е. максимизировать $|\varepsilon|$, необходимо минимизировать расстояние $\mathrm{d}(\langle a, x\rangle, \langle b, F(x)\rangle)$ по всем возможным ненулевым $a \in \mathbb{F}_2^n$, $b \in \mathbb{F}_2^m$. Соответственно противодействием данной атаке является выбор в качестве S-блоков таких векторных функций, у которых минимальное расстояние $\mathrm{d}(\langle a, x\rangle, \langle b, F(x)\rangle)$ по всем возможным ненулевым a, b как можно больше. Заметим, что это эквивалентно рассмотрению нелинейности компонентных функций $\langle b, F\rangle$ функции F.

8.3. Теоретические результаты и открытые вопросы

Приведём некоторые факты, опираясь на работы [42, 25]. Пусть f — булева функция от n переменных. Ранее мы получили оценку нелинейности $N_f \leqslant 2^{n-1} - 2^{n/2-1}$, которая достигается при чётном n для бент-функций. Для нечётного числа переменных точного значения максимальной нелинейности в общем случае не известно. Например, установлено, что при $n = 1, 3, 5, 7$ для f от n переменных $N_f \leqslant 2^{n-1} - 2^{(n-1)/2}$, и данная оценка достигается для квадратичных функций; но при нечётных $n > 7$ существуют функции, нелинейность которых строго больше $2^{n-1} - 2^{(n-1)/2}$.

Теорема 6 (нелинейность случайной функции). Существует константа c, такая, что для почти всех булевых функций от n переменных $N_f \geqslant 2^{n-1} - 2c\sqrt{n}2^{n/2-1}$.

Данный результат говорит о том, что нелинейность произвольной функции близка к верхней границе, но, как это часто бывает, нахождение конкретных функций с высокой нелинейностью является нетривиальной задачей.

В следующей теореме приведём некоторые известные факты о бент-функциях.

Теорема 7.

1. Бент-функции существенно зависят от всех своих переменных.
2. Для бент-функции f от n переменных верно $\mathrm{wt}(f) = 2^{n-1} \pm 2^{n/2-1}$.
3. Для бент-функции f от n переменных верно $\deg(f) \leqslant n/2$.
4. Конструкция Мэйорана — МакФарланда. Пусть π — любая перестановка на множестве $\mathbb{F}_2^{n/2}$; h — произвольная булева функция от $n/2$ переменных. Тогда $f(x', x'') = \langle x', \pi(x'')\rangle \oplus h(x'')$ является бент-функцией от n переменных.

Из утверждений 3 и 4 теоремы 7 можно получить оценки мощности класса $\mathcal{B}_n$ бент-функций от n переменных.

Утверждение 3 (оценки $|\mathcal{B}_n|$). Справедливо: $2^{2^{n/2}}(2^{n/2})! \leqslant |\mathcal{B}_n| \leqslant 2^{2^{n-1}+C_n^{n/2}/2}$.

Известны некоторые улучшения данных оценок, но на качественном уровне они остаются такими же. Видно, что с ростом n разрыв в оценках становится очень большим. Самым насущным открытым вопросом в этой области является установление точного количества бент-функций или хотя бы нахождение более приемлемых оценок мощности класса бент-функций. А это, в свою очередь, связано с поиском новых конструкций. Более подробно об этом можно найти в [42].

Рассмотрим теперь случай векторной булевой (n, m)-функции F. Как мы видели, для того чтобы противостоять линейному криптоанализу, следует выбирать в качестве S-блоков функции, нелинейность компонентных функций которых высока. Поэтому нелинейность векторной функции определяется как $N_F = \min\limits_{v \in \mathbb{F}_2^m, v \neq \mathbf{0}} N_{\langle v, F\rangle}$. Следовательно, также справедлива оценка $N_F \leqslant 2^{n-1} - 2^{n/2-1}$. Функции, нелинейность которых достигает данной оценки, также называются *векторными бент-функциями*. Вопрос состоит в том, а всегда ли они существуют?

Теорема 8 (существование бент-функций). Векторные бент-функции из $\mathbb{F}_2^n$ в $\mathbb{F}_2^m$ существуют только при $m \leqslant n/2$, где n чётно.

Теорема 9 (Сидельников). При $m \geqslant n-1$ для нелинейности произвольной векторной (n, m)-функции F верна оценка $N_F \leqslant 2^{n-1} - \dfrac{1}{2}\sqrt{3 \cdot 2^n - 2 - 2\dfrac{(2^n-1)(2^{n-1}-1)}{2^m-1}}$.

При $n = m$ из оценки Сидельникова следует, что $N_F \leqslant 2^{n-1} - 2^{(n-1)/2}$. При этом известно, что оценка точная. Максимально нелинейные (n, n)-функции существуют при нечётных n и называются *почти бент-функциями* (AB-функциями). Но до сих пор остаётся открытым вопрос о максимальной нелинейности в случаях, если:

а) n нечётное и $m < n-1$;

б) n чётное и $n/2 < m < n-1$.

9. Статистическая независимость

Понятие статистической независимости введено в [4] в связи с рассмотрением статистических аналогов функций.

Определение 12. Булева функция f от n переменных *статистически не зависит* от подмножества своих переменных $U = \{x_{i_1}, \dots, x_{i_k}\}$, если выполнено: $\mathsf{P}[f^{a_1,\dots,a_k}_{i_1,\dots,i_k} = 0] = \mathsf{P}[f = 0]$ для любых значений $a_1, \dots, a_k \in \mathbb{F}_2$.

Справедлив конструктивный тест на проверку статистической независимости [4].

Теорема 10 (критерий статистической независимости). Булева функция $f(x, y)$ от $n + m$ переменных, где $x \in \mathbb{F}_2^n$, $y \in \mathbb{F}_2^m$, статистически не зависит от переменных в x, если и только если $W_f(u, \mathbf{0}) = 0$ для любого $u \in \mathbb{F}_2^n$.

9.1. Статистические аналоги

Пусть $F : \mathbb{F}_2^n \times \mathbb{F}_2^r \to \mathbb{F}_2^m$. Данная функция может представлять собой, например, преобразование блока открытого текста длины n с помощью ключа длины r в блок шифртекста длины m.

Определение 13. *Статистическим аналогом* функции F называется уравнение $\varphi(x, y, k) = 0$, где $x \in \mathbb{F}_2^n$, $y \in \mathbb{F}_2^m$, $k \in \mathbb{F}_2^r$ связаны соотношением $y = F(x, k)$ и $\varphi : \mathbb{F}_2^n \times \mathbb{F}_2^m \times \mathbb{F}_2^r \to \mathbb{F}_2$ такая, что функция $\varphi_F(x, k) = \varphi(x, F(x, k), k)$ статистически не зависит от переменных в x. Число $p = \mathsf{P}[\varphi_F = 0]$ называется *вероятностью* данного статистического аналога, при $p \neq 1/2$ аналог *эффективен*.

Эффективные статистические аналоги рассматриваются с той же целью, что и линейные приближения шифра, рассмотренные в п. 8.2, — смоделировать шифр некоторыми упрощёнными уравнениями, связывающими биты открытого текста, шифртекста и ключа, выполняющимися с некоторой вероятностью. Но существенное их отличие заключается в том, что ассоциированные с данными уравнениями функции статистически не зависят от переменных символов открытого текста, что гарантирует сохранение вероятности выполнения данного уравнения при подстановке в них любых значений открытых текстов и соответствующих шифртекстов.

В работе [4] рассматриваются также вопросы построения статистических аналогов аналогично методу «от простого с сложному», т. е. как строить статистические аналоги всего шифра, отталкиваясь от статистических аналогов его простых компонент. Например, «для суперпозиции двух дискретных функций определяется суперпозиция одной из них (внутренней) и статистического аналога другой (внешней) и показывается, что в случае аддитивности внутренней функции полученная суперпозиция является функцией статистического аналога для первой суперпозиции с вероятностью статистического аналога её внешней функции» [4]. Кроме того, приводятся алгоритмы криптоанализа блочных шифров путём решения систем линейных и нелинейных статистических аналогов функций шифрования методом максимального правдоподобия, которые подкрепляются примерами для шифра DES.

10. Алгебраическая иммунность

Пусть задана булева функция f от n переменных. Булева функция g от n переменных называется *аннулятором* функции f, если выполнено равенство $fg = 0$.

Определение 14. *Алгебраической иммунностью* $\mathrm{AI}(f)$ функции f называется такое наименьшее число d, что существует аннулятор g степени d, не тождественно равный нулю, либо для функции f, либо для $f \oplus 1$.

10.1. Идея алгебраической атаки

Перед тем как давать пояснение определению алгебраической иммунности булевой функции, необходимо сказать следующее. Вообще говоря, любой шифр, как поточный, так и блочный, можно описать в виде системы булевых уравнений, в которой участвуют биты ключа, открытого текста и шифртекста. Таким образом, если мы знаем несколько пар открытого текста и шифртекста, полученных на одном неизвестном ключе, то, подставив их в данную систему, можно попытаться её решить и найти ключ. Отличительная особенность такой системы в том, что она гарантированно совместна, но для реальных шифров её решение представляется затруднительным. Известно, что задача решения систем нелинейных булевых уравнений в общем случае NP-трудна.

Существуют некоторые подходы к решению систем нелинейных булевых уравнений (подробно с ними можно познакомиться по работе Г. П. Агибалова [2]). Опишем один из них. Поскольку для решения линейных булевых систем существует эффективный метод Гаусса, то естественной идеей в общем случае является попытка сделать нелинейную систему линейной, но уже от большего числа переменных. Такой метод называется методом *линеаризации*. Однако чем выше алгебраическая степень системы, тем больше переменных придётся вводить в общем случае.

Соответственно промежуточным вопросом является следующий: а можно ли сначала уменьшить степень системы, не потеряв её решений, а уже затем применять метод линеаризации? В 2003 г. N. Courtois и W. Meier [29] предложили алгебраический криптоанализ фильтрующего генератора, основанный на понижении степени системы уравнений. Позже данный подход был обобщён и для комбинирующих генераторов и блочных шифров, а также сформировано итоговое понятие алгебраической иммунности булевой функции. Опишем идею алгебраического криптоанализа фильтрующего генератора по пособию [14].

Рассмотрим фильтрующий генератор с функцией h от n переменных. Если $f(x) = \langle c, x\rangle$ — закон рекурсии использующегося LFSR, где $c \in \mathbb{F}_2^n$, то на очередном i-м такте работы, $i = 1, 2, \ldots$, на вход фильтрующей функции подаётся значение векторной линейной функции $L^i(K)$, где $L(x_{n-1}, \ldots, x_0) = (x_{n-2}, \ldots, x_0, f(x_{n-1}, \ldots, x_0))$ и $K = (k_{n-1}, \ldots, k_0)$ — начальное состояние регистра.

Если $u_0, u_1, u_2, \ldots$ — выходная последовательность генератора, то

$$\begin{cases} u_0 = & h(k_{n-1}, \ldots, k_0), \\ u_1 = & h\big(L(k_{n-1}, \ldots, k_0)\big), \\ \vdots & \\ u_i = & h\big(L^i(k_{n-1}, \ldots, k_0)\big), \\ \vdots & \end{cases}$$

Из этих равенств строится нелинейная система булевых уравнений от неизвестных битов ключа K, если известен фрагмент последовательности $\{u_i\}$. Попытаемся теперь понизить степень уравнений данной системы. Предположим, что выполнены одно или оба из следующих условий:

1) существует функция g, такая, что $h(x)g(x) = t(x) \not\equiv 0$ и степень t мала;
2) существует функция $g \not\equiv 0$ малой степени, такая, что $h(x)g(x) \equiv 0$.

Тогда можем понизить степени уравнений системы следующим образом:

— если $u_i = 0$, то вместо $h\big(L^i(k_{n-1}, \ldots, k_0)\big) = 0$ рассмотрим уравнение $t\big(L^i(k_{n-1}, \ldots, k_0)\big) = 0$ при выполнении условия 1;
— если $u_i = 1$, то вместо $h\big(L^i(k_{n-1}, \ldots, k_0)\big) = 1$ рассмотрим уравнение $g\big(L^i(k_{n-1}, \ldots, k_0)\big) = 0$ при выполнении условия 2.

Обратим внимание на условие 1. Имеем $hg = t$; домножая равенство на h, получаем $hg = th$. Следовательно, $t = th$, или $(h \oplus 1)t = 0$. Перефразируя условие 1, получаем:

1') существует функция $t \not\equiv 0$ малой степени, такая, что $(h(x) \oplus 1)t(x) \equiv 0$.

Таким образом, чтобы препятствовать проведению алгебраического криптоанализа фильтрующего генератора с функцией h, необходимо, чтобы для всех функций g, таких, что $hg = 0$ или $(h \oplus 1)g = 0$, степень $\deg(g)$ была достаточно большой. Алгебраической иммунностью функции h назвали минимальную степень такой функции g.

10.2. Базовые результаты и связь с нелинейностью

Приведём некоторые известные факты, следуя обзору [24].

Легко видеть, что степень $\deg(f)$ служит естественной верхней оценкой алгебраической иммунности функции. Кроме того, справедлива следующая верхняя оценка алгебраической иммунности, зависящая только от числа переменных n.

Теорема 11 (верхняя оценка AI). Для произвольной булевой функции f от n переменных выполнено $\mathrm{AI}(f) \leqslant \lceil n/2 \rceil$, где $\lceil k \rceil$ — целая часть сверху числа k.

При этом известно, что данная оценка достижима, что подтверждается примерами в следующей теореме.

Теорема 12 (функции с максимальной AI). Следующие функции от n переменных имеют максимальную алгебраическую иммунность $\lceil n/2 \rceil$:

1) для нечётного n: $$f(x) = \begin{cases} 0, & \text{если } \mathrm{wt}(x) < \lceil n/2 \rceil, \\ 1, & \text{если } \mathrm{wt}(x) \geqslant \lceil n/2 \rceil; \end{cases}$$

2) для чётного n: $$f(x) = \begin{cases} 0, & \text{если } \mathrm{wt}(x) < n/2, \\ b \in \{0,1\}, & \text{если } \mathrm{wt}(x) = n/2, \\ 1, & \text{если } \mathrm{wt}(x) > n/2. \end{cases}$$

Хотя существуют примеры функций с максимальной алгебраической иммунностью, известно про этот класс функций очень мало. Любопытным также является следующий факт, который показывает, что алгебраическая иммунность произвольной функции достаточно высока.

Теорема 13 (AI случайной функции). Для любого $a < 1$ и для почти всех булевых функций f от n переменных выполнено $\mathrm{AI}(f) > n/2 - \sqrt{n/2 \cdot \ln\bigl(n/(2a \ln 2)\bigr)}$.

С обобщениями понятия алгебраической иммунности на случай (n,m)-функций можно познакомиться по работе [25].

Приведём также известную точную нижнюю оценку нелинейности функции через её алгебраическую иммунность, полученную М. С. Лобановым [10].

Теорема 14 (связь AI и N_f). Для булевой функции f от n переменных справедлива оценка $N_f \geqslant 2 \sum\limits_{i=0}^{\mathrm{AI}(f)-2} C_{n-1}^i$.

Несмотря на то, что по этой оценке нелинейность и порядок алгебраической иммунности функции «не противоречат» друг другу, высокая алгебраическая иммунность совсем не гарантирует высокой нелинейности. Действительно, при оптимальной алгебраической иммунности $\lceil n/2 \rceil$ данная оценка принимает следующий вид: $N_f \geqslant 2^{n-1} - C_{n-1}^{(n-1)/2}$ при нечётном n и $N_f \geqslant 2^{n-1} - C_n^{n/2}$ при чётном n. Как мы видели в п. 8.3, данная оценка далека от максимального значения нелинейности функции $2^{n-1} - 2^{n/2-1}$ и, более того, далека и от нелинейности случайной функции (теорема 6).

11. Уровень аффинности и k-нормальность

Определение 15. Булева функция f от n переменных называется *k-аффинной*, $0 \leqslant k \leqslant n-1$, если существует набор индексов $1 \leqslant i_1 < \ldots < i_k \leqslant n$ и значения $a_1, \ldots, a_k \in \mathbb{F}_2$, такие, что подфункция $f_{i_1,\ldots,i_k}^{a_1,\ldots,a_k}$ является аффинной.

Определение 16. *Уровнем аффинности* $\mathrm{la}f$ булевой функции f называется минимальное неотрицательное целое число k, для которого f является k-аффинной.

Данный параметр булевой функции стал рассматриваться в связи с предложенной О. А. Логачевым, А. А. Сальниковым, В. В. Ященко атакой на комбинирующий генератор [12]. Он связан с возможностью применения метода линеаризации без введения новых переменных для системы булевых уравнений, описывающей работу генератора. Чем ниже уровень аффинности комбинирующей функции, тем эффективнее криптоанализ. Подробное изучение $\mathrm{la}f$ представлено в работе М. Л. Бурякова [6], где, в частности, исследована связь данного параметра с другими основными криптографическими свойствами. Отметим, что известно асимптотическое поведение уровня аффинности, которое показывает, что он высокий.

Теорема 15 ($\mathrm{la}f$ случайной функции). При $n \to \infty$ для почти всех булевых функций f от n переменных верно $n - \lfloor \log_2 n \rfloor \leqslant \mathrm{la}f \leqslant n - \lceil \log_2 n \rceil + 1$.

В зарубежной литературе [27, 31] введено схожее понятие — k-нормальность.

Определение 17. Булева функция f от n переменных *k-нормальна*, если существует аффинное подпространство размерности k, на котором функция f постоянна.

Известен один пример успешного криптоанализа поточного шифра Grain [36], который как раз основан на том, что используемая в нём фильтрующая функция от 5 переменных имеет низкий порядок нормальности, а именно она 2-нормальная.

12. Дифференциальная равномерность

Принято считать, что определение дифференциально равномерных функций появилось в работах K. Nyberg [38] в начале 1990-х годов в связи с появлением дифференциального криптоанализа блочных шифров, предложенного E. Biham, A. Shamir для шифра DES в 1990 г. Однако необходимо отметить, что подход, который заложен в данном понятии, на самом деле рассматривался уже в 1960-х годах в Советском Союзе, как отмечается в [7].

Определение 18. Векторная булева (n,m)-функция F называется *дифференциально δ-равномерной*, если для любых $a \neq \mathbf{0}$, b уравнение $F(x) \oplus F(x \oplus a) = b$ имеет не более δ решений.

Легко видеть, что минимальное такое δ равно 2^{n-m}.

Определение 19. Векторная булева (n,m)-функция F называется *совершенно нелинейной* (PN-функцией), если она дифференциально 2^{n-m}-равномерна.

Эквивалентным определением PN-функций является следующее:

Определение 19'. F — PN-функция, если её производные D_aF уравновешены по всем ненулевым направлениям $a \in \mathbb{F}_2^n$, т. е. $|\{x \in \mathbb{F}_2^n : D_aF(x) = y\}| = 2^{n-m}$ для всех $y \in \mathbb{F}_2^m$.

Заметим, что при $m = n$ PN-функций не существует, поскольку если x — решение уравнения $F(x) \oplus F(x \oplus a) = b$, то и $x \oplus a$ также является его решением.

Определение 20. Векторная булева (n,n)-функция F называется *почти совершенно нелинейной* (APN-функцией), если она дифференциально 2-равномерна.

12.1. Идея дифференциального криптоанализа

Опишем идею дифференциального криптоанализа блочного шифра, следуя работе [20] (алгоритм 2). Рассмотрим итеративный блочный шифр, состоящий из r раундов. Обозначим открытые тексты P, P', промежуточные шифртексты после i-го раунда — C_i, C_i' и итоговые шифртексты — C, C' соответственно. Пара векторов $(a,b)_i$ называется i-м дифференциалом шифра, если существуют открытые тексты P, P', такие, что $P \oplus P' = a$ и $C_i \oplus C_i' = b$. При этом вероятностью i-го дифференциала $(a,b)_i$ является величина $\mathsf{P}[C_i \oplus C_i' = b \mid P \oplus P' = a]$.

Алгоритм 2. Дифференциальный криптоанализ

1: Выбираем наиболее вероятный $(r-1)$-дифференциал шифра.
2: При фиксированном неизвестном ключе K набираем выборку из N четвёрок $\{P, P', C, C'\}$, таких, что $P \oplus P' = a$.
3: Перебирая раундовые подключи K_r, расшифровываем каждую из N пар C, C' до C_{r-1}, C_{r-1}' и проверяем, выполнено ли $C_{r-1} \oplus C_{r-1}' = b$.
4: Ключ, для которого равенство из п. 3 выполняется чаще всего, полагаем за верный.

Поскольку искать наиболее вероятные дифференциалы для всего шифра сразу, т. е. для функций, переводящих, например, 64 бита в 64 бита, представляется очень трудным, поступают так же, как и в линейном криптоанализе: действуют методом «от простого к сложному», а именно: анализируют сначала S-блоки, а затем расширяют дифференциалы на весь шифр. Таким образом, выбирая S-блоки, дифференциалы которых почти равновероятны, можно сделать весь шифр устойчивым к дифференциальному криптоанализу. Следовательно, интересен вопрос построения дифференциально δ-равномерных функций с как можно меньшим значением δ.

12.2. Связь PN- и бент-функций, открытые вопросы о APN-функциях

Отметим в первую очередь тесную связь совершенно нелинейных функций с максимально нелинейными, рассмотренными в п. 8.

Во-первых, само название «совершенно нелинейные» отражает то, что такие функции тоже сильно отличаются в некотором смысле от самых простых — линейных. Действительно, если L — линейная (n,m)-функция, то уравнение $L(x) \oplus L(x \oplus a) = b$ всегда имеет 2^n решений при $b = L(a)$, что далеко от оптимального значения 2^{n-m}.

Во-вторых, как оказалось, класс PN-функций совпадает с классом бент-функций! Согласно обзору [22], понятие совершенной нелинейности ввели W. Meier, O. Staffelbach для булевой функции. Вскоре было обнаружено, что совершенно нелинейные функции полностью совпадают с бент-функциями, которые ввёл уже в 1970-х годах O. S. Rothaus, что отражено в следующей теореме.

Теорема 16 (бент-функции и её производные). Булева функция f от n переменных — бент-функция тогда и только тогда, когда производные $D_a f$ уравновешены по всем ненулевым направлениям $a \in \mathbb{F}_2^n$.

Из определения векторной бент-функции, утверждения 1 (см. п. 3) и теоремы 16 следует, что классы PN-функций и векторных бент-функций совпадают. Следовательно, PN-функции существуют только при $m \leqslant n/2$.

Таким образом, PN- или бент-функции обладают наилучшей стойкостью как к линейному, так и к дифференциальному криптоанализу. Однако на практике более интересным является случай, когда $n = m$. Как уже отмечалось, при данных параметрах оптимальными с точки зрения дифференциальных характеристик являются APN-функции, т. е. дифференциально 2-равномерные. В связи с тем, что APN-функции представляют большой интерес для использования в шифрах, их изучению уделяется огромное внимание. Несмотря на это, APN-функции остаются «загадочными», поскольку о них больше открытых вопросов, чем известных фактов. Прежде всего, это построение новых APN-функций, описание класса APN-функций, поиск оценок количества APN-функций от произвольного числа переменных и др. Все известные конструкции почти совершенно нелинейных функций получены с помощью алгебраического представления, при котором функция рассматривается как функция над конечным полем. Однако таких конструкций крайне мало. В данном небольшом обзоре упомянем лишь некоторые проблемы.

Интересным открытым вопросом является следующий: верно ли, что нелинейность APN-функций так же высока? Известно следующее утверждение.

Утверждение 4. Функция F — APN тогда и только тогда, когда её коэффициенты Уолша — Адамара удовлетворяют тождеству $\sum\limits_{u,v\in\mathbb{Z}_2^n} \big(W_F(u,v)\big)^4 = 3\cdot 2^{4n} - 2\cdot 2^{3n}$.

Данное тождество не позволяет получить оценок нелинейности APN-функции, но из него следует, что любая AB-функция является APN. Таким образом, AB-функции, так же как и бент, являются оптимальными с точки зрения двух криптографических критериев. Но большой минус AB-функций в том, что они существуют только для нечётного числа переменных, что неудобно для применения на практике. Про нелинейность APN-функций можно наверняка сказать только то, что она не равна нулю. Все известные примеры APN-функций показывают, что их нелинейность достаточно высока, но её нетривиальных оценок снизу или сверху нет даже для случая квадратичных функций.

Стоит отметить про случай нечётного числа переменных ещё и то, что существуют APN-функции, которые не являются AB. Известным примером такой функции является взаимно однозначная функция обращения элемента в конечном поле $\mathbb{F}_{2^n}$: $F(x) = x^{2^n-2}$. Как упоминается в обзоре М. М. Глухова [7], оптимальное свойство данной функции было указано В. А. Башевым и исследовано Б. А. Егоровым ещё в 1968 г. Б. А. Егоровым показано также, что для чётного n соответствующая подстановка не является APN-функцией, но дифференциально 4-равномерна. Именно эта функция от восьми переменных используется в качестве S-блока известного шифра AES.

Таким образом, мы подошли ко второй насущной проблеме в области APN-функций: существованию APN-подстановок при чётном числе переменных. Вычислительно было проверено для функций от двух и четырёх переменных, что ответ на вопрос о существовании APN-подстановок отрицательный, даже высказывалось предположение, что такой ответ верен и в общем случае. Можно представить, каким было удивление, когда в 2009 г. на криптографической конференции J. F. Dillon и др. предъявили взаимно однозначную APN-функцию от шести переменных. «The discovery in 2009 of an APN permutation in a field of characteristic 2 and even dimension has brought new motivation and new ideas to this field of research» [22]. В настоящее время все усилия направлены на поиск ответа на этот вопрос, в частности, для восьми переменных — случая, который является принципиальным для криптографии. Отметим, что в работе [15] В. Н. Сачков развивает новый комбинаторный подход к исследованию и методам построения взаимно однозначных APN-функций.

Упомянем ещё один любопытный открытый вопрос. В работе [26] для (n,n)-функции F определена ассоциированная булева функция γ_F от $2n$ переменных по следующему правилу: $\gamma_F(a,b) = 1$, где $a, b \in \mathbb{F}_2^n$, если $a \neq \mathbf{0}$ и уравнение $F(x) \oplus F(x \oplus a) = b$ имеет решение, и $\gamma_F(a,b) = 0$ иначе. Там же установлена следующая связь.

Утверждение 5. Пусть F — (n,n)-функция. Справедливы утверждения:
1) F — APN-функция тогда и только тогда, когда $\mathrm{wt}(\gamma_F) = 2^{2n-1} - 2^{n-1}$;
2) F — AB-функция тогда и только тогда, когда γ_F — бент-функция.

Функции F и G называются *дифференциально эквивалентными* [35], если $\gamma_F = \gamma_G$. Как оказалось, описать классы дифференциальной эквивалентности APN-функций — очень непростая задача. Изучение данного вопроса начато автором в работах [9, 35], где, в частности, установлено на примере известных APN-функций Голда, что существуют APN-функции, классы дифференциальной эквивалентности которых нетривиальные, а именно: полностью описаны аффинные функции, прибавление которых к APN-функциям Голда не выводит за рамки их классов дифференциальной эквивалентности. Получены вычислительные результаты о таких аффинных функциях для известных квадратичных APN-функций от малого числа переменных 2, …, 8. Отметим, что полное решение вопроса об описании дифференциально эквивалентных APN-функций может потенциально привести к новым конструкциям APN-функций.

13. Разложимость в сумму специальных функций

Такое свойство векторных булевых функций связано с решением задачи *маскирования данных*, которая возникла в связи с появлением атак по сторонним каналам. Криптографические алгоритмы оказались уязвимы к таким атакам, направленным на слабости в практической реализации алгоритма. Криптоаналитик исследует специфические для данной реализации параметры, например такие, как потребляемая мощность, время выполнения операций, электромагнитное излучение. Сравнив их на разных входных данных и набрав некоторую статистику, он может получить информацию о секретном ключе, выполняемых в устройстве операциях и их параметрах. В качестве мер противодействия используются методы, маскирующие входные данные так, чтобы вычисления не зависели от них в явном виде.

Рассмотрим одну идею достаточно нового метода реализации векторных булевых функций с целью маскирования данных. Данный метод порогового разбиения (threshold implementation) получил своё начало в работе бельгийских авторов [37].

Определение 21. Пороговым разбиением (n,m)-функции S на r частей называется такой набор (nr,m)-функций S_j, $j = 1,\ldots,r$, для которого выполняются следующие свойства:

Корректность. Для всех $x = (x_1,\ldots,x_n)$ верно: $S(x_1,\ldots,x_n) = \bigoplus_{j=1}^{r} S_j(\mathbf{x}^1,\ldots,\mathbf{x}^r)$ для любых $\mathbf{x}^i = (x_1^i,\ldots,x_n^i) \in \mathbb{F}_2^n$, удовлетворяющих условию $x = \mathbf{x}^1 \oplus \ldots \oplus \mathbf{x}^r$.

Неполнота. Каждая функция S_j, $j = 1,\ldots,r$, не зависит от переменных $\mathbf{x}^j$.

Равномерность. Для любых векторов $y, y^1,\ldots,y^r \in \mathbb{F}_2^m$, таких, что $\bigoplus_{j=1}^{r} y^j = y$, справедливо $|S_j^{-1}(y^j)| = 2^{(r-1)(m-n)}|S^{-1}(y)|$ для любого $j = 1,\ldots,r$.

Теоретически доказано, что если вместо векторной функции реализовать её пороговое разбиение в криптосистеме, то это сделает систему стойкой к дифференциальной атаке по энергопотреблению первого порядка (first-order differential power analysis) даже при наличии импульсных помех (glitches). Интуитивно, пороговое разбиение функции на r частей представляет собой некоторую схему разделения секрета — входного значения функции x — между r игроками — векторами $\mathbf{x}^j$, $j = 1,\ldots,r$. При этом никакие $r-1$ игроков не могут восстановить секрет, что обеспечивается требованием неполноты разбиения: каждая из S_j не зависит от переменных $\mathbf{x}^j$, следовательно, даже узнав её, мы не сможем восстановить исходное входное значение x. Требование корректности естественно: пороговое разбиение реализует то же преобразование, что и исходная функция. Требование равномерности необходимо для того, чтобы обеспечивать «правильный переход» между раундами, т. е. чтобы на вход очередного раунда преобразования поступало равномерно распределённое значение.

Естественным вопросом является следующий: каково минимальное значение r для функции S, для которого существует пороговое разбиение S на r частей? Вопрос актуален с практической точки зрения, поскольку напрямую связан с размерами и стоимостью реализации всего криптоалгоритма. Известно, что для функций алгебраической степени d минимальное такое r не меньше $d+1$. При этом всегда легко построить такое пороговое разбиение, удовлетворяющее только свойствам корректности и неполноты, сложнее обеспечить равномерность. Более того, как показали исследования для взаимно однозначных функций от малого числа переменных [21], не для всех функций степени d можно построить пороговое разбиение на $d+1$ частей (назовём его *минимальным*). Следовательно, формулируется открытая математическая задача: можно ли выделить некоторую легко проверяемую отличительную особенность тех функций, для которых минимальное пороговое разбиение существует? И если не существует минимального, то какое существует?

Поскольку эти вопросы пока открыты, разработчики данного метода пошли по пути, что называется, наименьшего сопротивления: не можем найти минимальное разбиение — будем искать пороговое разбиение на большее число частей. Так, например, вычислительно показано, что любую взаимно однозначную функцию от не более чем пяти переменных можно представить пороговым разбиением на 5 частей. Но необходимо находить компромисс между размерами и стоимостью реализации. Поэтому получила распространение и другая идея: те функции S, для которых не получается найти минимальное разбиение, представлять в виде композиции двух или более других функций, например $S = F \circ G$, которые, во-первых, меньшей степени, а во-вторых, для них легко строятся минимальные пороговые разбиения.

14. Мультипликативная сложность

Коротко упомянем ещё один аспект, связанный с использованием булевых функций в качестве криптографических элементов.

Определение 22. *Мультипликативной (конъюнктивной) сложностью* $MC(F)$ векторной булевой (n, m)-функции F называется минимальное число умножений в $\mathbb{F}_2$, необходимое и достаточное для вычисления $F(x)$ для любого $x \in \mathbb{F}_2^n$ в базисе $\{\cdot, \oplus, 1\}$.

Мультипликативная сложность функций связана с размерами и стоимостью аппаратной реализации шифров, которые их используют. Чем меньше мультипликативная сложность функции, тем проще её схемная реализация. Особенно это актуально для так называемой легковесной (или малоресурсной) криптографии.

В булевом случае ($m = 1$) данная сложность изучается ещё с 1960-х годов, позднее целый ряд работ был посвящён исследованию мультипликативной сложности различных классов функций: квадратичных, симметрических и др. (см., например, [16]). В векторном случае особый интерес представляют взаимно однозначные (n, n)-функции. Однако для произвольного n результатов известно мало. В работе [44] исследована мультипликативная сложность для малых значений n равных 3 и 4.

Как уже упоминалось в п. 13, в современных аппаратных реализациях требуется маскировать производимые вычисления с целью защиты от атак по сторонним каналам, причём важно накладывать различные маски именно на выполняемые нелинейные преобразования. Соответственно чем их меньше, тем дешевле это сделать. Хотя, как отмечается в [44], сложность реализаций взаимно однозначных (n, n)-функций в данном случае рассматривается обычно как число умножений над полем $\mathbb{F}_{2^n}$, а не $\mathbb{F}_2$, остаётся открытым вопрос, а не эффективнее ли в данном случае использовать именно мультипликативную сложность.

Отметим также, что существует предположение [28], что низкая мультипликативная сложность преобразования, реализующего весь шифр, может, наоборот, свидетельствовать о его слабости, в частности, приводить к возможности применения алгебраического криптоанализа шифра.

15. Линеаризационные множества

Рассматриваемый подход предложен Г. П. Агибаловым в работе [3].

Пусть имеется некоторая система S булевых уравнений.

Определение 23. Подмножество X переменных системы S называется *линеаризационным*, если при любой фиксации значений этих переменных в системе последняя превращается в линейную систему уравнений.

Заметим, что мощность линеаризационного множества переменных системы можно понизить путём введения вспомогательных переменных.

Пусть $x_1, \ldots, x_k$ — переменные системы S. Если набор значений $a_1, \ldots, a_k$ является полным решением системы S, то любой поднабор $a_{j_1}, \ldots, a_{j_i}$ значений переменных $x_{j_1}, \ldots, x_{j_i}$ соответственно, где $1 \leqslant j_1 \leqslant \ldots \leqslant j_i \leqslant k$, называется *частичным решением* системы. Частичное решение *квазиполное*, если его подстановка в систему делает её линейной.

15.1. Идея линеаризационной атаки

Работу генератора ключевого потока можно описать в виде системы булевых уравнений от неизвестных битов ключа, если известен некоторый конечный отрезок выходной последовательности. Суть линеаризационной атаки [3] состоит в поиске линеаризационного множества переменных как можно меньшей мощности для данной системы и соответствующего ему квазиполного решения системы. Действительно, осуществляя перебор значений переменных из некоторого линеаризационного множества и проверяя, будет ли совместной линейная система уравнений, полученная при подстановке данных значений, можно найти квазиполное решение системы, а следовательно, и полное, решив с полиномиальной сложностью полученную линейную систему уравнений. При этом сложность атаки оценивается величиной $2^{|X|}$, где $|X|$ — мощность линеаризационного множества переменных системы.

В качестве иллюстрации работы данной атаки приведём комбинирующий генератор Geffe, который состоит из трёх LFSR максимального периода длин L_1, L_2, L_3 соответственно и комбинирующей функции $f(x, y, z) = xy \oplus yz \oplus z$. Система булевых уравнений, описывающая работу генератора, следующая:

$$\begin{cases} u_t = & x_t^1 x_t^2 \oplus x_t^2 x_t^3 \oplus x_t^3,\ t = 0, 1, \ldots, m-1, \\ x_{L_1+t}^1 = & \bigoplus\limits_{j=0}^{L_1-1} c_j^1 x_{t+j}^1,\ t = 0, 1, \ldots, m-1-L_1, \\ x_{L_2+t}^2 = & \bigoplus\limits_{j=0}^{L_2-1} c_j^2 x_{t+j}^2,\ t = 0, 1, \ldots, m-1-L_2, \\ x_{L_3+t}^3 = & \bigoplus\limits_{j=0}^{L_3-1} c_j^3 x_{t+j}^3,\ t = 0, 1, \ldots, m-1-L_3, \end{cases}$$

где $m \geqslant \max\{L_1, L_2, L_3\}$; $u_0, u_1, \ldots, u_{m-1}$ — известный начальный отрезок выходной последовательности; c_j^i — некоторые заданные константы. Переменными неизвестных битов ключа являются $x_0^1, \ldots, x_{L_1-1}^1$, $x_0^2, \ldots, x_{L_2-1}^2$, $x_0^3, \ldots, x_{L_3-1}^3$. При этом множество $X = \{x_0^2, \ldots, x_{L_2-1}^2\}$ образует линеаризационное множество переменных системы. Следовательно, линеаризационную атаку на данный генератор можно реализовать со сложностью не выше 2^{L_2} путём опробования наборов значений переменных в X. Сложность такой атаки оказывается ниже, чем корреляционной (см. п. 7.1), сложность которой $2^{L_1} + 2^{L_2} + 2^{L_3}$.

Аналогичные результаты справедливы и для генератора с альтернативным управлением, мультиплексорного генератора и генератора скалярного умножения [3].

Заключение

В качестве заключения хотелось бы отметить красоту математики, которая скрыта в криптографических булевых функциях, особенно в нелинейных в разных смыслах. Казалось бы, определения бент-функций или APN-функций может понять и школьник, а полностью описать их классы (даже найти приемлемые оценки мощностей данных классов) не представляется возможным известным математикам! Вполне вероятно, что ответы на эти вопросы (а они рано или поздно все равно появятся) окажутся не такими важными с прикладной точки зрения, однако они будут очень любопытными для математики как таковой.

В данном обзоре мы очень мало осветили связи между различными криптографическими свойствами. Практика показывает, что в качестве компонент шифра необходимо выбирать «хорошие со всех сторон» функции, что на самом деле очень непростая задача, поскольку многие свойства порой противоречат друг другу. Хотя, как мы видели, теоретические результаты показывают, что у случайной функции многие криптографические параметры близки к оптимальным. Вопрос в том, а как её выбрать, случайную?

И наконец, хотелось бы отметить другую важную сторону, касающуюся исследований в области математической криптографии: историю изучения различных свойств булевых функций. К сожалению, многие работы советских математиков-криптографов до сих пор остаются опубликованными лишь в специальных сборниках, в то время как аналогичные работы, полученные за рубежом, известны и считаются основополагающими. И лишь постепенно начинают появляться упоминания о первых авторах-исследователях криптографических булевых функций в статьях и докладах конференций в России, посвящённых криптографической тематике.

Автор выражает благодарность рецензентам за ценные замечания и дополнения.

ЛИТЕРАТУРА

1. *Агибалов Г. П.* Избранные теоремы начального курса криптографии: учеб. пособие. Томск: Изд-во НТЛ, 2005. 116 с.
2. *Агибалов Г. П.* Методы решения систем полиномиальных уравнений над конечным полем // Вестник Томского государственного университета. Приложение. 2006. № 17. С. 4–9.
3. *Агибалов Г. П.* Логические уравнения в криптоанализе генераторов ключевого потока // Вестник Томского государственного университета. Приложение. 2003. № 6. С. 31–41.
4. *Агибалов Г. П., Панкратова И. А.* Элементы теории статистических аналогов дискретных функций с применением в криптоанализе итеративных блочных шифров // Прикладная дискретная математика. 2010. № 3. С. 51–68.
5. *Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В.* Основы криптографии. М.: Гелиос АРВ, 2002. 480 с.
6. *Буряков М. Л.* Алгебраические, комбинаторные и криптографические свойства параметров аффинных ограничений булевых функций: дис. ... канд. физ.-мат. наук. М., 2007.
7. *Глухов М. М.* О совершенно и почти совершенно нелинейных функциях // Математические вопросы криптографии. 2016. (в печати)
8. *Глухов М. М.* Планарные отображения конечных полей и их обобщения. Презентация для конф. «Алгебра и логика, теория и приложения». Красноярск, 21–27 июля, 2013.
9. *Городилова А. А.* Характеризация почти совершенно нелинейных функций через подфункции // Дискретная математика. 2015. Т. 27. Вып. 3. С. 3–16.
10. *Лобанов М. С.* Точное соотношение между нелинейностью и алгебраической иммунностью // Дискретная математика. 2006. Т. 18. Вып. 3. С. 152–159.
11. *Логачев О. А., Сальников А. А., Смышляев С. В., Ященко В. В.* Булевы функции в теории кодирования и криптологии. 2-е изд. М.: МЦНМО, 2012. 584 с.
12. *Логачев О. А., Сальников А. А., Ященко В. В.* Корреляционная иммунность и реальная секретность // Математика и безопасность информационных технологий. Материалы конф. в МГУ 23–24 октября 2003 г. М.: МЦНМО, 2004. С. 165–171.
13. *Панков К. Н.* Асимптотические оценки для чисел двоичных отображений с заданными криптографическими свойствами // Математические вопросы криптографии. 2014. Т. 5. Вып. 4. С. 73–97.
14. *Панкратова И. А.* Булевы функции в криптографии: учеб. пособие. Томск: Издательский Дом Томского государственного университета, 2014. 88 с.
15. *Сачков В. Н.* Комбинаторные свойства дифференциально 2-равномерных подстановок // Математические вопросы криптографии. 2015. Т. 6. Вып. 1. С. 159–179.
16. *Селезнева С. Н.* Мультипликативная сложность некоторых функций алгебры логики // Дискретная математика. 2014. Т. 26. Вып. 4. С. 100–109.
17. *Смышляев С. В.* О криптографических слабостях некоторых классов преобразований двоичных последовательностей // Прикладная дискретная математика. 2010. № 1. С. 5–15.
18. *Сумароков С. Н.* Запреты двоичных функций и обратимость для одного класса кодирующих устройств // Обозрение прикладной и промышленной математики. 1994. Т. 1. Вып. 1. С. 33–55.
19. *Таранников Ю. В.* О корреляционно-иммунных и устойчивых булевых функциях // Математич. вопросы кибернетики. 2002. Вып. 11. С. 91–148.
20. *Токарева Н. Н.* Симметричная криптография. Краткий курс: учеб. пособие. Новосибирск: Новосибирский государственный университет, 2012. 232 с.
21. *Bilgin B., Nikova S., Nikov V., et al.* Threshold implementations of small S-boxes // Cryptography and Communications. 2015. V. 7. No. 1. P. 3–33.

22. *Blondeau C. and Nyberg K.* Perfect nonlinear functions and cryptography // Finite Fields and their Applications. 2015. V. 32. P. 120–147.
23. *Braeken A.* Cryptographic Properties of Boolean Functions and S-boxes. PhD Thesis, Katholieke Universiteit Leuven, 2006.
24. *Carlet C.* Boolean functions for cryptography and error correcting codes // Ch. 8 of the Monograph "Boolean Methods and Models in Mathematics, Computer Science, and Engineering", Cambridge Univ. Press, 2010. P. 257–397.
25. *Carlet C.* Vectorial Boolean functions for cryptography // Ch. 9 of the Monograph "Boolean Methods and Models in Mathematics, Computer Science, and Engineering", Cambridge Univ. Press, 2010. P. 398–472.
26. *Carlet C., Charpin P., and Zinoviev V.* Codes, bent functions and permutations suitable for DES-like cryptosystems // Des. Codes Cryptogr. 1998. V. 15. P. 125–156.
27. *Charpin P.* Normal Boolean functions // J. Complexity. 2004. V. 20. P. 245–265.
28. *Courtois N., Hulme D., and Mourouzis T.* Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis. Cryptology ePrint Archive. Report 2011/475 (2011).
29. *Courtois N. and Meier W.* Algebraic attack on stream ciphers with linear feedback // LNCS. 2003. V. 2656. P. 345–359.
30. *Cusick T. W. and Stănică P.* Cryptographic Boolean Functions and Applications. Acad. Press. Elsevier, 2009. 245 p.
31. *Dobbertin H.* Construction of bent functions and balanced Boolean functions with high nonlinearity // FSE'95. LNCS. 1995. V. 1008. P. 61–74.
32. *Evertse J. H.* Linear structures in block ciphers // EUROCRYPT'87. LNCS. 1988. V. 304. P. 249–266.
33. *Fon-Der-Flaass D. G.* A bound on correlation immunity // Siberian Elektron. Mat. Izv. 2007. No. 4. P. 133–135.
34. *Golić J. Dj.* On the security of nonlinear filter generators // FSE'96. LNCS. 1996. V. 1039. P. 173–188.
35. *Gorodilova A.* On a Remarkable Property of APN Gold Functions. Cryptology ePrint Archive. Report 2016/286 (2016).
36. *Mihaljevic M., Gangopadhyay S., Paul G., and Imai H.* An algorithm for the internal state recovery of Grain-v1 // Proc. CECC'2011. Debrecen, Hungary, June 30–July 2, 2011. P. 7–20.
37. *Nikova S., Rechberger C., and Rijmen V.* Threshold implementations against side-channel attacks and glitches // LNCS. 2006. V. 4307. P. 529–545.
38. *Nyberg K.* Differentially uniform mappings for cryptography // Eurocrypt'93. LNCS. 1994. V. 765. P. 55–64.
39. *Preneel B., Van Leekwijck W., Van Linden L., et al.* Propagation characteristics of Boolean functions // Eurocrypt'90. LNCS. 1991. V. 473. P. 161–173.
40. *Siegenthaler T.* Correlation-immunity of nonlinear combining functions for cryptographic applications // IEEE Trans. Inform. Theory. 1984. V. 30. No. 5. P. 776–780.
41. *Tarannikov Y. V.* Generalized proper matrices and constructing of m-resilient Boolean functions with maximal nonlinearity for expanded range of parameters // Siberian Elektron. Mat. Izv. 2014. No. 11. P. 229–245.
42. *Tokareva N.* Bent Functions: Results and Applications to Cryptography. Acad. Press. Elsevier, 2015. 220 p.
43. *Webster A. F. and Tavares S. E.* On the design of S-boxes // Crypto'85. LNCS. 1986. V. 218. P. 523–534.

44. *Zajac P. and Jokay M.* Multiplicative complexity of bijective 4×4 S-boxes // Cryptography and Communications. 2014. V. 6. No. 3. P. 255–277.
45. *Zhang X.-M. and Zheng Y.* GAC — the criterion for Global Avalanche Characteristics of cryptographic functions // J. Universal Computer Science. 1995. V. 1. No. 5. P. 320–337.

REFERENCES

1. *Agibalov G. P.* Izbrannye teoremy nachal'nogo kursa kriptografii: ucheb. posobie [Selected Theorems of Basic Cryptography Course: Tutorial]. Tomsk, NTL Publ., 2005. (in Russian)
2. *Agibalov G. P.* Metody resheniya sistem polinomial'nykh uravneniy nad konechnym polem [Methods for solving systems of polynomial equations over a finite field]. Vestnik Tomskogo Gosudarstvennogo Universiteta. Prilozhenie, 2006, no. 17, pp. 4–9. (in Russian)
3. *Agibalov G. P.* Logicheskie uravneniya v kriptoanalize generatorov klyuchevogo potoka [Logical equations in cryptanalysis of key stream generators]. Vestnik Tomskogo Gosudarstvennogo Universiteta. Prilozhenie, 2003, no. 6, pp. 31–41. (in Russian)
4. *Agibalov G. P. and Pankratova I. A.* Elementy teorii statisticheskikh analogov diskretnykh funktsiy s primeneniem v kriptoanalize iterativnykh blochnykh shifrov [Statistical approximation theory for discrete functions with application in cryptanalysis of iterative block ciphers]. Prikladnaya Diskretnaya Matematika, 2010, no. 3, pp. 51–68. (in Russian)
5. *Alferov A. P., Zubov A. Yu., Kuz'min A. S., and Cheremushkin A. V.* Osnovy Kriptografii [Basics of Cryptography]. Moscow, Gelios ARV Publ., 2002. (in Russian)
6. *Buryakov M. L.* Algebraicheskie, kombinatornye i kriptograficheskie svoystva parametrov affinnykh ogranicheniy bulevykh funktsiy [Algebraic, Combinatorial, and Cryptographic properties of Parameters of Boolean Functions Affine Restrictions]. PhD Thesis, Moscow, 2007. (in Russian)
7. *Glukhov M. M.* O sovershenno i pochti sovershenno nelineynykh funktsiyakh [About perfectly and almost perfectly non-linear functions]. Matematicheskie Voprosy Kriptografii, 2016. (to be published) (in Russian)
8. *Glukhov M. M.* Planarnye otobrazheniya konechnykh poley i ikh obobshcheniya [On planar maps and their generalisation to finite fields]. Pres. conf. “Algebra and Logic, Theory and Applications”, Krasnoyarsk, 21–27 July 2013.
9. *Gorodilova A. A.* Kharakterizatsiya pochti sovershenno nelineynykh funktsiy cherez podfunktsii [Characteristics of almost perfectly non-linear functions by subfunctions]. Diskr. Mat., 2015, vol. 27, no. 3, pp. 3–16. (in Russian)
10. *Lobanov M. S.* Tochnoe sootnoshenie mezhdu nelineynost'yu i algebraicheskoy immunnost'yu [Exact relation between nonlinearity and algebraic immunity]. Diskr. Mat., 2006, vol. 18, no. 3, pp. 152–159. (in Russian)
11. *Logachev O. A., Sal'nikov A. A., Smyshlyaev S. V., and Yashchenko V. V.* Bulevy funktsii v teorii kodirovaniya i kriptologii [Boolean Functions in Coding Theory and Cryptology]. Moscow, MCCME Publ., 2012. (in Russian)
12. *Logachev O. A., Sal'nikov A. A., and Yashchenko V. V.* Korrelyatsionnaya immunnost' i real'naya sekretnost' [Correlation immunity and real privacy]. Proc. conf. “Mathematics and Security of Information Technologies”, Moscow, MCCME Publ., 2004, pp. 165–171. (in Russian)
13. *Pankov K. N.* Asimptoticheskie otsenki dlya chisel dvoichnykh otobrazheniy s zadannymi kriptograficheskimi svoystvami [Asymptotic estimates for numbers of Boolean mappings with given cryptographic properties]. Mat. Vopr. Kriptogr., 2014, vol. 5, iss. 4, pp. 73–97. (in Russian)
14. *Pankratova I. A.* Bulevy funktsii v kriptografii: ucheb. posobie [Boolean Functions in Cryptography: Tutorial]. Tomsk, TSU Publ., 2014. (in Russian)

15. *Sachkov V. N.* Kombinatornye svoystva differentsial'no 2-ravnomernykh podstanovok [Combinatorial properties of differentially 2-uniform substitutions]. Mat. Vopr. Kriptogr., 2015, vol. 6, iss. 1, pp. 159–179. (in Russian)
16. *Selezneva S. N.* Mul'tiplikativnaya slozhnost' nekotorykh funktsiy algebry logiki [Multiplicative complexity of some Boolean functions]. Diskr. Mat., 2014, vol. 26, iss. 4, pp. 100–109. (in Russian)
17. *Smyshlyaev S. V.* O kriptograficheskikh slabostyakh nekotorykh klassov preobrazovaniy dvoichnykh posledovatel'nostey [On cryptographic weaknesses of some classes of binary sequence transformations]. Prikladnaya Diskretnaya Matematika, 2010, no. 1, pp. 5–15. (in Russian)
18. *Sumarokov S. N.* Zaprety dvoichnykh funktsiy i obratimost' dlya odnogo klassa kodiruyushchikh ustroystv [Prohibitions of binary functions and reversibility for a class of encoders]. Obozrenie Prikladnoy i Promyshlennoy Matematiki, 1994, vol. 1, no. 1, pp. 33–55. (in Russian)
19. *Tarannikov Yu. V.* O korrelyatsionno-immunnykh i ustoychivykh bulevykh funktsiyakh [On correlation-immune and resilient Boolean functions]. Mat. Voprosy Kibernetiki, 2002, vol. 11, pp. 91–148. (in Russian)
20. *Tokareva N. N.* Simmetrichnaya kriptografiya. Kratkiy kurs: ucheb. posobie. [Symmetric Cryptography. Short Course: Tutorial]. Novosibirsk, NSU Publ., 2012. (in Russian)
21. *Bilgin B., Nikova S., Nikov V., et al.* Threshold implementations of small S-boxes. Cryptography and Communications, 2015, vol. 7, no. 1, pp. 3–33.
22. *Blondeau C. and Nyberg K.* Perfect nonlinear functions and cryptography. Finite Fields and their Applications, 2015, vol. 32, pp. 120–147.
23. *Braeken A.* Cryptographic Properties of Boolean Functions and S-boxes. PhD Thesis, Katholieke Universiteit Leuven, 2006.
24. *Carlet C.* Boolean functions for cryptography and error correcting codes. Ch. 8 of the Monograph "Boolean Methods and Models in Mathematics, Computer Science, and Engineering", Cambridge Univ. Press, 2010, pp. 257–397.
25. *Carlet C.* Vectorial Boolean functions for cryptography. Ch. 9 of the Monograph "Boolean Methods and Models in Mathematics, Computer Science, and Engineering", Cambridge Univ. Press, 2010, pp. 398–472.
26. *Carlet C., Charpin P., and Zinoviev V.* Codes, bent functions and permutations suitable for DES-like cryptosystems. Des. Codes Cryptogr., 1998, vol. 15, pp. 125–156.
27. *Charpin P.* Normal Boolean functions. J. Complexity, 2004, vol. 20, pp. 245–265.
28. *Courtois N., Hulme D., and Mourouzis T.* Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis. Cryptology ePrint Archive. Report 2011/475 (2011).
29. *Courtois N. and Meier W.* Algebraic attack on stream ciphers with linear feedback. LNCS, 2003, vol. 2656, pp. 345–359.
30. *Cusick T. W. and Stănică P.* Cryptographic Boolean Functions and Applications. Acad. Press. Elsevier, 2009. 245 p.
31. *Dobbertin H.* Construction of bent functions and balanced Boolean functions with high nonlinearity. FSE'95, LNCS, 1995, vol. 1008, pp. 61–74.
32. *Evertse J. H.* Linear structures in block ciphers. EUROCRYPT'87, LNCS, 1988, vol. 304, pp. 249–266.
33. *Fon-Der-Flaass D. G.* A bound on correlation immunity. Siberian Elektron. Mat. Izv., 2007, no. 4, pp. 133–135.
34. *Golić J. Dj.* On the security of nonlinear filter generators. FSE'96, LNCS, 1996, vol. 1039, pp. 173–188.

35. *Gorodilova A.* On a Remarkable Property of APN Gold Functions. Cryptology ePrint Archive. Report 2016/286 (2016).
36. *Mihaljevic M., Gangopadhyay S., Paul G., and Imai H.* An algorithm for the internal state recovery of Grain-v1. Proc. CECC'2011 Debrecen, Hungary, June 30–July 2, 2011, pp. 7–20.
37. *Nikova S., Rechberger C., and Rijmen V.* Threshold implementations against side-channel attacks and glitches. LNCS, 2006, vol. 4307, pp. 529–545.
38. *Nyberg K.* Differentially uniform mappings for cryptography. Eurocrypt'93, LNCS, 1994, vol. 765, pp. 55–64.
39. *Preneel B., Van Leekwijck W., Van Linden L., et al.* Propagation characteristics of Boolean functions. Eurocrypt'90, LNCS, 1991, vol. 473, pp. 161–173.
40. *Siegenthaler T.* Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. Inform. Theory, 1984, V. 30, no. 5, pp. 776–780.
41. *Tarannikov Y. V.* Generalized proper matrices and constructing of m-resilient Boolean functions with maximal nonlinearity for expanded range of parameters. Siberian Elektron. Mat. Izv., 2014, no. 11, pp. 229–245.
42. *Tokareva N.* Bent Functions: Results and Applications to Cryptography. Acad. Press. Elsevier, 2015. 220 p.
43. *Webster A. F. and Tavares S. E.* On the design of S-boxes. Crypto'85, LNCS, 1986, vol. 218, pp. 523–534.
44. *Zajac P. and Jokay M.* Multiplicative complexity of bijective 4×4 S-boxes. Cryptography and Communications, 2014, vol. 6, no. 3, pp. 255–277.
45. *Zhang X.-M. and Zheng Y.* GAC — the criterion for Global Avalanche Characteristics of cryptographic functions. J. Universal Computer Science, 1995, vol. 1, no. 5, pp. 320–337.

УДК 519.7

О ПОНЯТИИ ε-СОВЕРШЕННОГО ШИФРА

А. Ю. Зубов

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

Обсуждаются обобщения понятия совершенного шифра. Шифр называется ε-совершенным, если максимальное значение модуля разности апостериорной и априорной вероятностей открытого текста не превосходит ε. Изучаются две конструкции шифров, которые являются ε-совершенными для любого множества открытых текстов, частотные характеристики которых удовлетворяют незначительному ограничению. Понятие ε-совершенного шифра является одним из возможных приближений к понятию совершенного шифра. Приводятся результаты сравнения изучаемых конструкций шифров по степени близости различных таких приближений, свидетельствующие в пользу понятия ε-совершенности и её аналогов.

Ключевые слова: *совершенный шифр, ε-совершенный шифр.*

DOI 10.17223/20710410/33/3

ON THE CONCEPT OF A ε-PERFECT CIPHER

A. Yu. Zubov

Lomonosov Moscow State University, Moscow, Russia

E-mail: Zubovanatoly@yandex.ru

The generalizations of the perfect cipher concept are discussed. A cipher is called ε-perfect if the maximum absolute value of the difference between the posterior and prior probabilities of a plaintext does not exceed ε. Two constructions of ε-perfect ciphers for a multitude of plaintexts with a minor limitation of their frequency characteristics are studied. The notion of ε-perfect cipher is one of the possible approximations to the notion of a perfect cipher. For studied constructions of ciphers, it is shown that, in comparison with the other such approximations, ε-perfectness and its analogues have much better proximity to perfectness.

Keywords: *perfect cipher, ε-perfect cipher.*

Введение

В [1] введено понятие ε-совершенного шифра (который является совершенным шифром при $\varepsilon = 0$) и построены примеры таких шифров. Интерес к ним вызван следующими причинами. Основным недостатком совершенного шифра является чрезмерно большое число ключей, что делает такой шифр непрактичным. Замена жёсткого условия $\varepsilon = 0$ условием $\varepsilon > 0$ для достаточно малого ε незначительно понижает стойкость шифра (оставляя его в классе теоретически стойких шифров), но допускает возможность использования небольшого числа ключей, что делает шифр более практичным.

Вместе с тем свойство совершенности шифра не зависит от распределения вероятностей на множестве открытых текстов [2], тогда как примеры ε-совершенных шифров из [1] построены лишь при условии, что открытые тексты выбираются случайно и равновероятно. Это ограничивает область применения таких шифров и делает их, в свою очередь, менее практичными, чем совершенные шифры. В данной работе показано, что на самом деле предложенные в [1] шифры остаются ε-совершенными и для любого распределения на множестве открытых текстов, удовлетворяющего некоторому ограничению. При этом параметр ε может принимать достаточно малые значения.

Помимо предложенного в [1] определения ε-совершенности известны и другие подобные определения. Так, в [3] введены шифры, «близкие» к совершенным, с позиции статистического расстояния между вероятностными распределениями. Интересен вопрос о сопоставлении мер «близости» к совершенным шифрам, а также их адекватности. Такой анализ проводится в данной работе на основе предложенной в [1] конструкции шифра.

1. Определения и конструкции

Пусть $\mathcal{S}, \mathcal{K}, \mathcal{M}$ — множества открытых текстов, ключей и шифрованных текстов шифра Σ и $\{E_k : k \in \mathcal{K}\}$ — множество функций зашифрования, представляющих собой инъективные отображения $\mathcal{S} \to \mathcal{M}$. Пусть S, K, M — случайные величины, принимающие значения из $\mathcal{S}, \mathcal{K}, \mathcal{M}$ в соответствии с распределениями вероятностей $\mathcal{P}_S = (p_S(s), s \in \mathcal{S})$, $\mathcal{P}_K = (p_K(k), k \in \mathcal{K})$, $\mathcal{P}_M = (p_M(m), m \in \mathcal{M})$, где

$$p_M(m) = \sum_{k \in \mathcal{K}(m)} p_K(k)\, p_S\left(E_k^{-1}(m)\right), \tag{1}$$

и $\mathcal{K}(m) = \{k \in \mathcal{K} : \exists s \in \mathcal{S}\, (E_k(s) = m)\}$. При этом случайные величины S, K полагаются независимыми.

Пусть $\mathcal{P}_{S,M}$, $\mathcal{P}_{S|M}$, $\mathcal{P}_{M|S}$ — совместное и условные распределения, для которых вероятность $p_{M|S}(m|s)$ вычисляется по формуле

$$p_{M|S}(m|s) = \sum_{k \in \mathcal{K}(s,m)} p_K(k), \tag{2}$$

где $\mathcal{K}(s, m) = \{k \in \mathcal{K} : E_k(s) = m\}$, а вероятность $p_{S|M}(s|m)$ — по формуле

$$p_{S|M}(s|m) = \frac{p_{M|S}(m|s)\, p_S(s)}{p_M(m)}. \tag{3}$$

Шифр Σ называется *совершенным*, если для любых $s \in \mathcal{S}$, $m \in \mathcal{M}$ выполняется равенство

$$p_{S|M}(s|m) = p_S(s). \tag{4}$$

Ослабим условие (4) и введём понятие ε-совершенного шифра.

Используя обозначения

$$\Delta(s, m) = \left|p_{S|M}(s|m) - p_S(s)\right|, \quad \nabla(s, m) = \left|p_{M|S}(m|s) - p_M(m)\right|,$$

запишем (4) в виде равенства

$$\max_{s,m} \Delta(s, m) = 0.$$

Заметим, что, согласно (3), величины $\Delta(s, m)$ и $\nabla(s, m)$ связаны соотношением

$$\begin{aligned} \Delta(s,m) = \left|\frac{p_{M|S}(m|s)\, p_S(s)}{p_M(m)} - p_S(s)\right| = \\ = \frac{p_S(s)}{p_M(m)} \left|p_{M|S}(m|s) - p_M(m)\right| = \frac{p_S(s)}{p_M(m)} \nabla(s, m). \end{aligned} \tag{5}$$

В [1] шифр Σ назван ε-совершенным, если для любых $s \in \mathcal{S}$, $m \in \mathcal{M}$

$$\max_{s,m} \Delta(s,m) \leqslant \varepsilon. \tag{6}$$

Там же предложены конструкции ε-совершенных шифров Σ_1 и Σ_2 с равномерными распределениями $\mathcal{P}_S$, $\mathcal{P}_K$. Для шифра Σ_1

$$\mathcal{S} = (F_q)^r,\ \mathcal{K} = (F_q)^2,\ \mathcal{M} = (F_q)^{r+1},$$

где F_q — поле характеристики 2, состоящее из q элементов (этот случай удобен для эффективной реализации), и r — натуральное число. Функция зашифрования строки $s = (s_1, \ldots, s_r)$ на ключе $k = (a, b)$ определяется формулой

$$E_k(s) = \left(u_1, \ldots, u_r, a + u_1 \cdot b^1 + \ldots + u_r \cdot b^r\right),$$

где

$$u_i = s_i + c_i \cdot a + d_i \cdot b, \quad i = 1, \ldots, r,$$

а $c_1, \ldots, c_r, d_1, \ldots, d_r$ — произвольные ненулевые константы из F_q, такие, что $c_i \cdot d_j \neq$ $\neq c_j \cdot d_i$ при $i \neq j$. В [1] показано, что шифр Σ_1 является ε-совершенным для $\varepsilon < q^{-1}$.

Для шифра Σ_2

$$\mathcal{S} = (F_Q)^{2^t+1},\ \mathcal{K} = (F_Q)^2 \times F_q,\ \mathcal{M} = (F_Q)^{2^t+2},$$

где $q = 2^r$; $Q = 2^{r+t}$; r, t — натуральные числа, такие, что $2t \geqslant r$. Функция зашифрования строки $s = (s_{2^t}, \ldots, s_1, s_0)$ на ключе $k = (a, b, c)$ определяется формулой

$$E_k(s) = \left(u_{2^t}, \ldots, u_0, c + \left[b\left(u_{2^t} \cdot a^{2^t} + \ldots + u_1 \cdot a + u_0\right)\right]_q\right),$$

где

$$u_i = s_i + h_i \cdot a + g_i \cdot b, \quad i = 0, 1, \ldots, 2^t,$$

а $h_0, \ldots, h_{2^t}$, $g_0, \ldots, g_{2^t}$ — произвольные константы из F_Q, такие, что $h_i \cdot g_j \neq h_j \cdot g_i$ при $j \neq i$. Запись $[\alpha]_q$ означает приведение элемента $\alpha \in F_Q$ по модулю q. В [1] показано, что шифр Σ_2 является ε-совершенным для $\varepsilon = 2^{-(r+2t)} - 2^{-(r+1)\left(2^t+1\right)}$.

В следующем утверждении обобщаются результаты работы [1].

Теорема 1. Пусть для шифров Σ_1 и Σ_2 распределение $\mathcal{P}_K$ — равномерное, а $\mathcal{P}_S$ — любое такое распределение, что для некоторых действительных чисел α, β, δ, удовлетворяющих неравенствам $0 < \alpha$, $\beta < 1 \leqslant \delta$, выполняются условия

$$\alpha \leqslant p_S(s) \leqslant \beta, \quad \beta/\alpha \leqslant \delta \tag{7}$$

для каждого $s \in \mathcal{S}$. Тогда шифр Σ_1 является ε-совершенным для $\varepsilon < \delta q^{-1}$, а шифр Σ_2 — ε-совершенным для $\varepsilon < \delta \cdot 2^{-(r+2t)}$.

Доказательство. Согласно (1) и (2), для равномерного распределения $\mathcal{P}_K$ имеем

$$\nabla(s,m) = \left|\sum_{k \in \mathcal{K}(s,m)} p_K(k) - \sum_{k \in \mathcal{K}(m)} p_K(k)\, p_S\left(E_k^{-1}(m)\right)\right| = \frac{1}{|\mathcal{K}|}\left||\mathcal{K}(s,m)| - \sum_{k \in \mathcal{K}(m)} p_S\left(E_k^{-1}(m)\right)\right|.$$

В [1] показано, что для шифра Σ_1 при любых $s \in \mathcal{S}$, $m \in \mathcal{M}$

$$|\mathcal{K}(m)| = q, \quad |\mathcal{K}(s,m)| \leqslant 1.$$

В частности, при $k \neq k'$ невозможно равенство $E_k^{-1}(m) = E_{k'}^{-1}(m)$. Поэтому сумма $\sigma = \sum\limits_{k \in \mathcal{K}(m)} p_S\left(E_k^{-1}(m)\right)$ заключена в пределах $0 < \sigma < 1$, откуда получаем неравенство

$$\nabla(s,m) \leqslant \frac{1}{|\mathcal{K}|} \max\{\sigma, |1-\sigma|\} < \frac{1}{|\mathcal{K}|}. \tag{8}$$

Теперь из (1), (5), (7), (8) следует искомое неравенство для шифра Σ_1:

$$\Delta(s,m) < \frac{p_S(s)}{p_M(m)\,|\mathcal{K}|} \leqslant \frac{\beta\,|\mathcal{K}|}{q\alpha\,|\mathcal{K}|} = \frac{\beta}{\alpha q} \leqslant \delta q^{-1}.$$

В [1] показано также, что для шифра Σ_2 при любых $s \in \mathcal{S}$, $m \in \mathcal{M}$

$$|\mathcal{K}(m)| = 2^{r+2t}, \quad |\mathcal{K}(s,m)| \leqslant 1. \tag{9}$$

Поэтому справедлива оценка (8) и, с учётом (9), отсюда следует искомое неравенство для шифра Σ_2: $\quad \Delta(s,m) < \delta \cdot 2^{-(r+2t)}$. ■

Замечание 1. Условие (7) вводит ограничения, при которых шифры Σ_1 и Σ_2 являются ε-совершенными. Основное ограничение связано с величиной δ. Что можно о ней сказать? При достаточно больших q, например $q = 2^{128}$, частотные характеристики открытых текстов для шифра Σ_1 «распределяются» среди строк длины $128r$ битов. Возможна ли при этом ситуация, когда некоторые тексты встречаются, например, в $\delta = 2^{64}$ раз чаще, чем другие? Если это и возможно, то лишь для текстов с «экзотической» частотной характеристикой. Но даже и в этом случае из теоремы 1 мы получаем для шифра Σ_1 оценку $\varepsilon < 2^{64}$. Для шифра Σ_2 и значительно большие значения δ дают малые значения ε. Это даёт основание полагать, что ограничения (7) не являются слишком «стеснительными» в реальных условиях.

Замечание 2. Конструкция шифра Σ_1 использует константы c_i, d_i, удовлетворяющие соотношениям $c_i \cdot d_j \neq c_j \cdot d_i$ при $i \neq j$. Можно предложить следующий способ выбора таких констант в случае, когда $q = 2^n$, $r < n$. Элемент α поля F_{2^n} представляется многочленом $\alpha_{n-1}x^{n-1} + \ldots + \alpha_1 x^1 + \alpha_0$ с коэффициентами из F_2. Пусть константа c_i представлена многочленом $f_i(x) = c_{n-1}^{(i)}x^{n-1} + \ldots + c_1^{(i)}x$, а d_i — многочленом $g_i(x) = c_{n-1}^{(i)}x^{n-1} + \ldots + c_1^{(i)}x + 1$. Пусть $c_i \neq c_j$, тогда и $f_i(x) \neq f_j(x)$. Произведение $c_i \cdot d_j$ представляется многочленом $f_i(x)g_j(x) = f_i(x)\left(f_j(x)+1\right)$, а произведение $c_j \cdot d_i$ — многочленом $f_j(x)g_i(x) = f_j(x)\left(f_i(x)+1\right)$. Отсюда следует, что для выбранных вариантов констант $c_i \cdot d_j \neq c_j \cdot d_i$. Выбирая произвольно различные двоичные векторы $\left(c_{n-1}^{(i)}, \ldots, c_1^{(i)}\right)$, получаем искомый набор констант.

Замечание 3. В [1] шифры Σ_1 и Σ_2 рассматривались как коды аутентификации с секретностью. Нетрудно заметить, что в условиях теоремы 1 имеют место следующие оценки вероятностей p_0, p_1 успеха имитации и подмены для шифра Σ_1:

$$p_0 = q^{-1}, \quad p_1 < r\delta q^{-1}.$$

Например, при $q = 2^{128}$, $r = 2^{32}$, $\delta = 2^{32}$ имеем оценки

$$p_0 = 2^{-128}, \quad p_1 < 2^{-64}.$$

2. Другие определения «почти совершенного» шифра

Хорошо известно (см., например, [2]), что критерием совершенности шифра Σ является любое из равенств

$$p_{M|S}(m|s) = p_M(m),$$
$$p_{M|S}(m|s) = p_{M|S}(m|s'),$$
$$p_{S,M}(s,m) = p_S(s)\,p_M(m),$$

которые должны выполняться для любых $s, s' \in \mathcal{S}$, $m \in \mathcal{M}$. В связи с этим мы могли бы определить ε-совершенный шифр одним из соответствующих неравенств

$$\max_{s,m} \left| p_{M|S}(m|s) - p_M(m) \right| \leqslant \varepsilon; \tag{10}$$

$$\max_{s,s',m} \left| p_{M|S}(m|s) - p_{M|S}(m|s') \right| \leqslant \varepsilon; \tag{11}$$

$$\max_{s,m} |p_{S,M}(s,m) - p_S(s)\,p_M(m)| \leqslant \varepsilon. \tag{12}$$

Оценим правые части неравенств (10)–(12) для шифра Σ_1. Из (2) и (8) следует, что для определений (10) и (11) шифр Σ_1 является ε-совершенным для $\varepsilon = 1/\,|\mathcal{K}| = q^{-2}$, причём для любого распределения $\mathcal{P}_S$. Такой же вывод справедлив и для определения (12). В самом деле,

$$\max_{s,m} |p_{S,M}(s,m) - p_S(s)\,p_M(m)| = \max_{s,m} \left| p_S(s)\,p_{M|S}(m|s) - p_S(s)\,p_M(m) \right| =$$
$$= \max_{s,m} p_S(s) \left| p_{M|S}(m|s) - p_M(m) \right| = \max_{s,m} p_S(s) \cdot \nabla(s,m) < \max_{s,m} \nabla(s,m) < 1/\,|\mathcal{K}|\,.$$

Полученные оценки свидетельствуют о том, что определения (6) и (10)–(12) не эквивалентны.

Введённые определения не являются единственно возможными определениями «почти совершенного» шифра. В [3] предлагаются другие определения с позиции статистической неразличимости, а также соотношения между ними. Кратко изложим содержание этой работы и оценим стойкость шифра Σ_1 с этих позиций.

Для распределений вероятностей $\mathcal{P}_U$, $\mathcal{P}_V$ на конечном множестве $\mathcal{A}$ *статистическим расстоянием* (или *расстоянием по вариации*) называется величина

$$d\left(\mathcal{P}_U, \mathcal{P}_V\right) = 0{,}5 \sum_{a \in \mathcal{A}} |p_U(a) - p_V(a)|\,.$$

Известно, что эта величина представляется также в виде

$$d\left(\mathcal{P}_U, \mathcal{P}_V\right) = \max_{f:\mathcal{A}\to\{0,1\}} |\mathsf{P}\left[f(U)=1\right] - \mathsf{P}\left[f(V)=1\right]|\,,$$

где U и V в правой части равенства рассматриваются как случайные величины на множестве $\mathcal{A}$, имеющие распределения $\mathcal{P}_U, \mathcal{P}_V$, а максимум берётся по всем возможным отображениям $f : \mathcal{A} \to \{0,1\}$.

Пусть для шифра Σ распределение $\mathcal{P}_{\mathcal{S}}$ может выбираться из некоторого семейства распределений $\mathcal{P}\left[\mathcal{S}\right]$. Шифр Σ называется

— $\varepsilon(S|M)$-*стойким*, если для любого $m \in \mathcal{M}$ справедливо неравенство

$$d\left(\mathcal{P}_{S|M}\left(\cdot|m\right), \mathcal{P}_S\left(\cdot\right)\right) \leqslant \varepsilon;$$

— $\varepsilon(M|S)$-*стойким*, если для любого $s \in \mathcal{S}$ справедливо неравенство

$$d\left(\mathcal{P}_{M|S}\left(\cdot|s\right), \mathcal{P}_M\left(\cdot\right)\right) \leqslant \varepsilon;$$

— $\varepsilon(M|S^2)$-*стойким*, если для любых $s, s' \in \mathcal{S}$ справедливо неравенство

$$d\left(\mathcal{P}_{M|S}\left(\cdot|s\right), \mathcal{P}_{M|S}\left(\cdot|s'\right)\right) \leqslant \varepsilon;$$

— $\varepsilon(S, M)$-*стойким*, если

$$d\left(\mathcal{P}_{S,M}\left(\cdot,\cdot\right), \mathcal{P}_S\left(\cdot\right)\cdot\mathcal{P}_M\left(\cdot\right)\right) \leqslant \varepsilon.$$

Эти определения являются аналогами определений (5), (10)–(12) соответственно. Очевидно, что при $\varepsilon = 0$ введённые шифры являются совершенными.

В [3] приведены ещё два определения: $\varepsilon(M|S^2)$-стойкий шифр назван также $IND(\varepsilon)$-*стойким* (или *статистически ε-неразличимым*). Шифр Σ назван *статистически ε-семантически стойким* (кратко — $SS(\varepsilon)$-*стойким*), если для любого $\mathcal{P}_S \in \mathcal{P}[\mathcal{S}]$ и любого отображения $f : \mathcal{M} \to \{0, 1\}$ существует случайная переменная G_f на $\{0, 1\}$, зависящая от f, но не зависящая от M, такая, что для каждого отображения $h : \mathcal{M} \to \{0, 1\}$ выполняется условие

$$|\mathsf{P}\left[f(M) = h(M)\right] - \mathsf{P}\left[G_f = h(M)\right]| \leqslant \varepsilon. \tag{13}$$

Качественно это определение означает, что шифртекст «почти бесполезен» для получения любого одного бита информации об открытом тексте M, поскольку из (13) следует, что при угадывании бита $h(M)$ «почти нет разницы» в том, что использовать — M и f или f и случайный бит G_f.

Результаты работы [3] сформулируем в виде следующего утверждения.

Теорема 2.

Шифр Σ является $\varepsilon(M|S)$-стойким тогда и только тогда, когда он является $IND(\varepsilon)$-стойким.

Шифр Σ является $\varepsilon(S, M)$-стойким, если он является $IND(\varepsilon)$-стойким. Обратно, шифр Σ является $IND(2\varepsilon)$-стойким, если он является $\varepsilon(S, M)$-стойким.

Шифр Σ является $SS(\varepsilon)$-стойким, если он является $IND(\varepsilon)$-стойким. Обратно, если шифр Σ является $SS(\varepsilon)$-стойким, то он является $IND(4\varepsilon)$-стойким.

Из теоремы 2 следует, что $\varepsilon(M|S)$, $\varepsilon(S, M)$, $IND(\varepsilon)$ и $SS(\varepsilon)$ — равносильные понятия стойкости. В то же время $\varepsilon(S|M)$ — более сильное понятие. В этом можно убедиться на примере шифра Σ, приведённого в [3], который для сколь угодно малого ε является $IND(\varepsilon)$-стойким, но при этом $\varepsilon'(S|M)$-стойким лишь для $\varepsilon' > 0{,}5$.

В свою очередь, введённые в [3] понятия стойкости сильнее их аналогов из [1]. Например, $\varepsilon(S|M)$-стойкий шифр является 2ε-совершенным в смысле определения (6), поскольку

$$\max_{a\in\mathcal{A}} |p_U(a) - p_V(a)| \leqslant \sum_{a\in\mathcal{A}} |p_U(a) - p_V(a)| \leqslant 2\varepsilon.$$

Для более точного сравнения подходов оценим стойкость шифра Σ_1 с позиции введённых понятий стойкости в случае, когда $\mathcal{P}_S$, $\mathcal{P}_K$ — равномерные распределения.

Используем определение $\varepsilon(M|S)$-стойкости. Вычислим расстояние

$$\sigma = d\left(\mathcal{P}_{M|S}\left(\cdot|s\right), \mathcal{P}_M\left(\cdot\right)\right) = 0{,}5 \sum_{m\in\mathcal{M}} \nabla(s, m). \tag{14}$$

Как показано в п. 1,

$$\nabla(s,m) = \frac{1}{|\mathcal{K}|}\left||\mathcal{K}(s,m)| - \frac{|\mathcal{K}(m)|}{|\mathcal{S}|}\right|. \tag{15}$$

Для каждого $s \in \mathcal{S}$ имеется q^2 элементов $m \in \mathcal{M}$, для которых $|\mathcal{K}(s,m)| = 1$. Для остальных элементов m выполняется равенство $|\mathcal{K}(s,m)| = 0$. Отсюда и из (14), (15) получаем

$$\sigma = 0{,}5\frac{1}{q^2}\left[q^2\left(1 - \frac{1}{q^{r-1}}\right) + \left(q^{r+1} - q^2\right)\frac{1}{q^{r-1}}\right] = \frac{q^{r-1}-1}{q^{r-1}}.$$

Таким образом, шифр Σ_1 является лишь $\dfrac{q^{r-1}-1}{q^{r-1}}$ $(M|S)$-стойким. Полученное значение ε неулучшаемо, поскольку мы точно вычислили $d\left(\mathcal{P}_{M|S}\left(\cdot|s\right), \mathcal{P}_M\left(\cdot\right)\right)$.

Используем определение $\varepsilon(S|M)$-стойкости. Вычислим расстояние

$$\sigma = d\left(\mathcal{P}_{S|M}\left(\cdot|m\right), \mathcal{P}_S\left(\cdot\right)\right) = 0{,}5\sum_{s\in\mathcal{S}}\Delta(s,m). \tag{16}$$

Как показано в п. 1,

$$\sigma = 0{,}5\sum_{s\in\mathcal{S}}\frac{p_S(s)}{p_M(m)}\left|p_{M|S}(m|s) - p_M(m)\right| = 0{,}5\frac{1}{q}\sum_{s\in\mathcal{S}}\left||\mathcal{K}(s,m)| - \frac{1}{q^{r+1}}\right|. \tag{17}$$

Для каждого $m \in \mathcal{M}$ имеется q элементов $s \in \mathcal{S}$, для которых $|\mathcal{K}(s,m)| = 1$. Для остальных элементов s выполняется равенство $|\mathcal{K}(s,m)| = 0$. Отсюда и из (16), (17) получаем

$$\sigma = \frac{0{,}5}{q}\left[q\left(1 - \frac{1}{q^{r-1}}\right) + \left(q^r - q\right)\frac{1}{q^{r-1}}\right] = \frac{q^{r-1}-1}{q^{r-1}}.$$

Таким образом, определения $\varepsilon(M|S)$- и $\varepsilon(S|M)$-стойкости дают для шифра Σ_1 одинаковые значения $\varepsilon = 1 - 1/q^{r-1}$. Нетрудно проверить, что для $\varepsilon(S,M)$-стойкости значение ε точно такое же.

Используем определение $\varepsilon(M|S^2)$. Вычисляем расстояние

$$\sigma = d\left(\mathcal{P}_{M|S}\left(\cdot|s\right), \mathcal{P}_{M|S}\left(\cdot|s'\right)\right),$$

в результате имеем

$$\sigma = 0{,}5q^{-2}\sum_{m\in\mathcal{M}}\left||\mathcal{K}(s,m)| - |\mathcal{K}(s',m)|\right|.$$

Пусть $s = (s_1, s_2, s_3, \ldots, s_r)$, $s' = (s_1, s_2, s_3', \ldots, s_r')$, тогда если $m = E_k(s)$, а $m' = E_{k'}(s')$ для некоторых ключей k, k', то равенство $m = m'$ невозможно. Поэтому множества из q^2 элементов m и m', таких, что $|\mathcal{K}(s,m)| = 1$ и $|\mathcal{K}(s',m')| = 1$, не пересекаются. Отсюда

$$\sigma = 0{,}5q^{-2}\left[q^2 + q^2\right] = 1.$$

Таким образом, пользуясь определением $\varepsilon(M|S^2)$-стойкости, получаем $\varepsilon = 1$. Следовательно, с позиции подхода к оценке стойкости, предложенного в [3], ни о какой «близости» шифра к совершенному шифру речи идти не может. Это и не удивительно, поскольку сумма вида $\sum\limits_{s\in\mathcal{S}}\Delta(s,m)$ накапливает большое количество «малых слагаемых», давая в результате «большую величину». В то же время определения (6) или

(10)–(12), на наш взгляд, более адекватны, поскольку отвечают классическому подходу «в худшем случае», который оценивает изучаемый параметр своим максимально возможным значением. Другой общепринятый подход — подход «в среднем» — оценивает изучаемый параметр своим средним значением, которое, очевидно, даёт не большее значение ε, чем определение (6).

Возвращаясь к исходной идее К. Шеннона о том, что шифртекст совершенного шифра не должен давать дополнительной вероятностной информации об открытом тексте (к известной априорной информации о нём), мы должны констатировать, что подход к определению «почти совершенного шифра» из [1] является более тонким, чем подход работы [3]. В самом деле, «почти совершенный» шифр должен быть таким, чтобы шифртекст «почти не давал» дополнительной информации об открытом тексте. Но если мера $\varepsilon(S|M)$-стойкости принимает для шифра Σ_1 значение, близкое к максимально возможному, то это свидетельствует о том, что шифртекст должен практически однозначно определять открытый текст. Получаем явное противоречие с тем, что для шифра Σ_1 (при равномерном распределении $\mathcal{P}_S$)

$$\max_{(s,m)} \left| p_{S|M}(s|m) - p_S(s) \right| \leqslant q^{-1}.$$

При $q = 2^{128}$, например, правая часть неравенства ($\approx 3 \cdot 10^{-39}$) практически равна 0, и нет никаких оснований полагать, что шифртекст сколь-нибудь значимо увеличивает априорную информацию об открытом тексте.

Заключение

На примере шифра, предложенного в [1], проведено сравнение различных определений «близости» шифра к совершенному шифру. На этом основании сделан вывод о том, что введённое в [1] определение ε-совершенного шифра и его аналоги соответствуют более адекватным мерам «близости» к совершенному шифру, нежели определения из [3]. Показано, что шифры из [1] остаются ε-совершенными для класса распределений на множестве открытых текстов, удовлетворяющих незначительному ограничению.

ЛИТЕРАТУРА

1. *Зубов А. Ю.* Почти совершенные шифры и коды аутентификации // Прикладная дискретная математика. 2011. № 4(14). С. 28–33.
2. *Зубов А. Ю.* Криптографические методы защиты информации. Совершенные шифры. М.: Гелиос АРВ, 2005.
3. *Iwamoto M. and Ohta K.* Security Notions for Information Theoretically Secure Encryptions. arXiv: 1106.1731 v2 [cs.CR], 4 Jan 2012. 6 p.

REFERENCES

1. *Zubov A. Yu.* Pochti sovershennye shifry i kody autentifikatsii [Almost perfect ciphers and authentication codes]. Prikladnaya Diskretnaya Matematika, 2011, no. 4(14), pp. 28–33.
2. *Zubov A. Yu.* Kriptograficheskie metody zashchity informatsii. Sovershennye shifry [Cryptographic Methods of Information Security. Perfect Ciphers]. Moscow, Gelios ARV Publ., 2005.
3. *Iwamoto M. and Ohta K.* Security Notions for Information Theoretically Secure Encryptions. arXiv: 1106.1731 v2 [cs.CR], 4 Jan 2012. 6 p.

УДК 512.13

ЗАДАНИЕ ПОДСТАНОВОК АЛГОРИТМОВ БЛОЧНОГО ШИФРОВАНИЯ МАГМА И 2-ГОСТ С ПОМОЩЬЮ АЛГЕБРАИЧЕСКИХ ПОРОГОВЫХ ФУНКЦИЙ

Д. А. Сошин

ФГУП «НИИ «Квант», г. Москва, Россия

Предложено задание подстановок алгоритмов блочного шифрования Магма и 2-ГОСТ через линейные комбинации алгебраических пороговых функций (АПФ). Для этого приведены полученные автором результаты представимости геометрических типов булевых функций от четырёх переменных через АПФ.

Ключевые слова: *алгебраические пороговые функции, геометрические типы, подстановки, блочные шифры.*

DOI 10.17223/20710410/33/4

THE IMPLEMENTATION OF MAGMA AND 2-GOST BLOCK CIPHER SUBSTITUTIONS BY ALGEBRAIC THRESHOLD FUNCTIONS

D. A. Soshin

Technology Federal State Unitary Enterprise "Research Institute Kvant", Moscow, Russia

E-mail: danil_re@list.ru

The paper deals with the implementation of substitutions in Magma and 2-GOST block cipher algorithms by algebraic threshold functions (ATF). For this purpose, the representations of all the geometric types of Boolean functions in 4 variables by ATF are given.

Keywords: *algebraic threshold functions, geometric types, substitutions, block ciphers.*

Введение

Пороговые функции представляют интерес с точки зрения синтеза узлов переработки информации [1] благодаря простой логике их задания и возможности быстрого выполнения операций в перспективной элементной базе, например в оптической [2]. Работы [3–5] посвящены изучению способов синтеза подстановок на основе классических пороговых функций. В данной работе предложено задание подстановок алгоритмов блочного шифрования Магма [6] и 2-ГОСТ [7] с помощью так называемых алгебраических пороговых функций. Класс АПФ предложен в работах [8, 9] и отличается от пороговых функций добавлением свободного члена к линейной форме и приведением её по определённому модулю до выполнения операций сравнения. Основным результатом первой работы является описание некоторого класса сбалансированных АПФ, а второй — доказательство того, что только один геометрический тип булевых функций от трёх переменных не принадлежит классу АПФ. Координатные функции указанных подстановок зависят от четырёх переменных, поэтому в работе приведены представления геометрических типов булевых функций четырёх переменных через АПФ.

Полученный каталог позволил изучить представимость линейных комбинаций координатных функций исследуемых подстановок. Через линейные комбинации, являющиеся АПФ, выразилась только одна подстановка. Остальные подстановки удалось задать через линейные комбинации пяти АПФ, полученных добавлением к координатным функциям специально выбранной АПФ. Отметим, что линейные функции содержатся в классе АПФ, поэтому полученные задания подстановок реализуются на единой элементной базе.

1. Представление геометрических типов булевых функций от четырёх переменных через АПФ

Определение 1. Функцию k-значной логики $f_n^k : \Omega_k^n \longrightarrow \Omega_k$ назовём *алгебраической пороговой*, если существуют целочисленные наборы $c = (c_0, c_1, \ldots, c_n)$, $b = (b_0, b_1, \ldots, b_k)$ и модуль $m \in \mathbb{N}$, такие, что для любого $\alpha \in \Omega_k$ выполняется

$$f_n^k(x_1, x_2, \ldots, x_n) = \alpha \quad \Leftrightarrow \quad b_\alpha \leqslant r_m(c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n) < b_{\alpha+1},$$

где $r_m(x)$ — функция взятия остатка числа x по модулю m, $r_m(x) \in \{0, 1, \ldots, m-1\}$; $\Omega_k = \{0, 1, 2, \ldots, k-1\}$. Тройку $(c; b; m)$ будем называть структурой функции f_n^k.

Далее рассмотрим двоичный случай. В случае двузначной логики АПФ будем задавать следующим образом:

$$f = 1 \quad \Leftrightarrow \quad r_m(c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n) \geqslant b$$

и писать $f : ((c_0, c_1, c_2, \ldots, c_n); b; m)$.

В [9] изложен подход к изучению представимости булевых функций трёх переменных через АПФ. Логика подхода заключается в разбиении всех булевых функций трёх переменных на классы эквивалентности относительно трёх операций — инвертирования переменных, перестановки переменных и инвертирования функции — и рассмотрении только одного представителя из класса. Класс эквивалентности относительно указанных операций будем называть *геометрическим типом*, а произвольное множество функций, замкнутое относительно этих операций, — *геометрически замкнутым*. Корректность такого подхода следует из того, что класс АПФ геометрически замкнут [9].

Далее представлены результаты исследования представимости через АПФ булевых функций от четырёх переменных. Для этого воспользуемся каталогом геометрических типов, составленным В. Г. Никоновым [10]. Удобство использования данного каталога заключается в предложенной автором нумерации, состоящей из четвёрки чисел $a.b.c.d$. Номер a отвечает за вес функций в геометрическом типе, причём если вес функции $\|f\| > 8$, то следует искать соответствующего представителя с весом менее 8 ($\|\overline{f}\| < 8$). Номер b отвечает за количество компонент связности в графе $\mathfrak{G}_f = (X, U)$ функции f. Вершинами X графа связности $\mathfrak{G}_f$ являются точки множества Ω_2^4, в которых функция принимает значение 1 (носитель функции):

$$X = \{(a_1, a_2, a_3, a_4) \in \Omega_2^4 : f(a_1, a_2, a_3, a_4) = 1\},$$

а множеством рёбер U является множество рёбер куба, соединяющих соседние вершины графа:

$$U = \left\{((a_1, a_2, a_3, a_4), (b_1, b_2, b_3, b_4)) \in X^2 | \quad \chi\left((a_1, a_2, a_3, a_4), (b_1, b_2, b_3, b_4)\right) = 1\right\},$$

где χ — расстояние Хемминга между точками (a_1, a_2, a_3, a_4) и (b_1, b_2, b_3, b_4):

$$\chi\left((a_1, a_2, a_3, a_4), (b_1, b_2, b_3, b_4)\right) = \sum_{i=1}^{4} \mathrm{Ind}(a_i \neq b_i),$$

$$\mathrm{Ind}(a_i \neq b_i) = \begin{cases} 1, & \text{если } a_i \neq b_i, \\ 0, & \text{если } a_i = b_i. \end{cases}$$

Номер c — порядковый номер графа связности с числом вершин a и количеством компонент связности b; d — порядковый номер геометрического типа с графом связности $a.b.c$. Стоит отметить, что предложенный каталог составлен в 1994 г. вручную и был впоследствии полностью подтверждён при сверке с результатами работы компьютера.

Далее сохранена введённая нумерация, и читатель может обратиться к указанному каталогу, но представитель геометрического типа будет предложен другой, а именно минимальный по лексикографическому номеру, либо $\overline{f}$ в случае, если первый представитель геометрического типа с минимальным номером f имеет вес более 8.

В [9] доказано, что класс АПФ замкнут относительно фиксации переменных и среди булевых функций от трёх переменных только один геометрический тип не принадлежит классу АПФ (для удобства представителя данного типа будем обозначать f^*). Функция f^* задаётся вектор-строкой

$$f^* = (1, 1, 1, 0, 0, 1, 0, 0). \tag{1}$$

В табл. 1 приведены геометрические типы булевых функций от четырёх переменных с указанием фиксации переменной, при которой подфункция эквивалентна функции (1). Из 222 геометрических типов 70 обладают указанным свойством. Отметим, что если в каталоге представитель f некоторого геометрического типа не содержит подфункцию f^*, то искать её среди подфункций $\overline{f}$ нет смысла, поскольку инвертирование подфункции не выводит за геометрический тип.

Утверждение 1. Геометрические типы булевых функций от четырёх переменных, представленные в табл. 1, не принадлежат классу АПФ.

Для оставшихся 152 геометрических типов была написана программа, которая нашла структуры 101 представителя. Логика работы программы заключается в переборе всевозможных АПФ, у которых коэффициенты линейной формы $(c_0, c_1, c_2, c_3, c_4)$ ограничены некоторой константой, а модуль m и порог b ограничивались максимальным значением $c_0 + c_1 + c_2 + c_3 + c_4 + 1$ текущей линейной формы. Для каждой из полученных АПФ проверяется её наличие среди представителей геометрических типов. Программа опробовала все задания АПФ, у которых коэффициенты линейной формы не превосходили значения 40. Результаты работы программы приведены в табл. 2.

Среди полученных 101 структур представителей геометрических типов максимальный коэффициент линейной формы равен 8, а максимальный модуль — 9. Доказать, что есть соответствующие ограничения, начиная с которых новые АПФ перестанут появляться, не удалось. Подходы, предложенные для подсчёта асимптотики пороговых функций в работах [11–13], на данный момент результатов не дали.

Таблица 1

Геометрические типы булевых функций от четырёх переменных, содержащие в качестве подфункции функцию, не принадлежащую классу АПФ

Номер $a.b.c.d$	Вектор-столбец представителя геометрического типа $a.b.c.d$	Фиксация $x_i = \alpha$	Номер $a.b.c.d$	Вектор-столбец представителя геометрического типа $a.b.c.d$	Фиксация $x_i = \alpha$
4.1.3.1	$(1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0)$	$x_4 = 0$	5.1.3.1	$(1,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0)$	$x_4 = 0$
5.1.4.2	$(1,1,0,1,0,0,0,1,1,0,0,0,0,0,0,0)$	$x_4 = 0$	5.2.3.1	$(1,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0)$	$x_3 = 0$
5.2.3.2	$(0,0,1,1,0,1,0,1,1,0,0,0,0,0,0,0)$	$x_4 = 0$	6.1.3.1	$(1,1,1,1,0,1,0,0,1,0,0,0,0,0,0,0)$	$x_2 = 0$
6.1.5.1	$(1,1,0,1,1,0,0,1,1,0,0,0,0,0,0,0)$	$x_3 = 0$	6.1.6.1	$(1,0,1,1,1,1,0,0,1,0,0,0,0,0,0,0)$	$x_3 = 0$
6.1.8.1	$(1,1,0,1,0,1,0,1,1,0,0,0,0,0,0,0)$	$x_2 = 0$	6.1.9.1	$(1,0,1,0,0,0,1,1,1,1,0,0,0,0,0,0)$	$x_4 = 0$
6.1.9.2	$(1,0,1,1,0,1,0,1,1,0,0,0,0,0,0,0)$	$x_3 = 0$	6.2.3.1	$(1,1,0,1,0,1,1,0,1,0,0,0,0,0,0,0)$	$x_2 = 0$
6.2.3.2	$(0,1,0,1,0,1,1,0,1,1,0,0,0,0,0,0)$	$x_2 = 0$	6.2.3.3	$(0,1,0,0,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_2 = 0$
6.2.4.4	$(0,1,0,0,1,1,1,0,0,1,1,0,0,0,0,0)$	$x_2 = 0$	6.2.4.5	$(1,0,0,1,1,0,1,0,1,1,0,0,0,0,0,0)$	$x_2 = 0$
6.2.7.1	$(0,1,0,1,1,0,1,0,1,1,0,0,0,0,0,0)$	$x_3 = 0$	6.2.7.2	$(0,0,0,1,1,0,1,1,1,1,0,0,0,0,0,0)$	$x_4 = 0$
6.3.3.1	$(1,1,0,0,1,0,0,1,0,1,1,0,0,0,0,0)$	$x_2 = 0$	6.3.3.2	$(1,0,0,0,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_4 = 0$
7.1.3.1	$(1,1,1,1,0,1,0,1,1,0,0,0,0,0,0,0)$	$x_2 = 0$	7.1.5.1	$(1,0,1,1,1,1,0,1,1,0,0,0,0,0,0,0)$	$x_2 = 0$
7.1.7.1	$(1,1,0,1,1,0,1,0,1,1,0,0,0,0,0,0)$	$x_1 = 0$	7.1.8.1	$(1,1,0,1,0,1,1,1,1,0,0,0,0,0,0,0)$	$x_2 = 0$
7.1.8.2	$(1,0,1,0,1,0,1,1,1,1,0,0,0,0,0,0)$	$x_2 = 0$	7.1.9.1	$(1,1,1,1,0,1,1,0,1,0,0,0,0,0,0,0)$	$x_2 = 0$
7.1.10.1	$(1,1,1,0,0,0,1,1,1,1,0,0,0,0,0,0)$	$x_1 = 0$	7.1.11.1	$(1,0,0,0,1,1,0,1,1,1,1,0,0,0,0,0)$	$x_4 = 0$
7.1.13.1	$(0,1,1,1,0,1,1,0,1,1,0,0,0,0,0,0)$	$x_2 = 0$	7.1.15.1	$(0,1,0,0,0,1,1,1,1,1,1,0,0,0,0,0)$	$x_4 = 0$
7.1.15.2	$(1,0,0,1,1,0,1,1,1,1,0,0,0,0,0,0)$	$x_2 = 0$	7.1.15.4	$(0,1,1,0,1,1,1,0,0,1,1,0,0,0,0,0)$	$x_2 = 0$
7.2.3.1	$(1,1,0,0,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_1 = 1$	7.2.5.1	$(1,0,1,1,0,1,1,0,1,1,0,0,0,0,0,0)$	$x_1 = 0$
7.2.5.2	$(0,1,0,1,1,1,1,0,0,1,1,0,0,0,0,0)$	$x_2 = 0$	7.2.6.1	$(1,1,1,0,1,0,0,1,0,1,1,0,0,0,0,0)$	$x_2 = 0$
7.2.6.2	$(0,1,1,1,0,0,0,1,0,1,1,0,1,0,0,0)$	$x_2 = 1$	7.2.8.1	$(0,1,0,0,1,1,1,1,0,1,1,0,0,0,0,0)$	$x_2 = 0$
7.2.9.1	$(1,0,1,0,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_1 = 0$	7.2.12.1	$(0,0,0,1,1,0,0,1,1,1,1,0,1,0,0,0)$	$x_2 = 0$
7.2.12.2	$(0,1,1,0,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_2 = 0$	7.2.16.1	$(0,0,1,0,1,1,0,1,1,1,1,0,0,0,0,0)$	$x_3 = 0$
7.3.3.1	$(1,1,0,1,0,0,0,1,0,1,1,0,1,0,0,0)$	$x_4 = 0$	7.3.4.1	$(0,0,1,1,0,1,0,1,0,1,1,0,1,0,0,0)$	$x_4 = 0$
7.3.5.1	$(0,1,0,1,1,0,0,1,0,1,1,0,1,0,0,0)$	$x_1 = 1$	8.1.4.1	$(1,1,1,1,0,1,1,1,1,0,0,0,0,0,0,0)$	$x_1 = 0$
8.1.7.1	$(1,1,1,1,0,1,1,0,1,1,0,0,0,0,0,0)$	$x_1 = 0$	8.1.8.1	$(1,1,1,1,0,0,1,1,1,1,0,0,0,0,0,0)$	$x_1 = 0$
8.1.9.1	$(1,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0)$	$x_2 = 0$	8.1.11.1	$(1,1,0,1,1,0,1,1,1,1,0,0,0,0,0,0)$	$x_1 = 0$
8.1.14.1	$(0,1,1,1,1,1,1,0,0,1,1,0,0,0,0,0)$	$x_1 = 0$	8.1.15.1	$(0,1,1,1,0,0,0,1,1,1,1,0,1,0,0,0)$	$x_1 = 0$
8.1.19.1	$(0,1,1,0,0,1,1,1,1,1,1,0,0,0,0,0)$	$x_1 = 0$	8.1.19.2	$(0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,0)$	$x_1 = 0$
8.1.21.1	$(1,1,1,0,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_1 = 0$	8.1.23.1	$(1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,0)$	$x_1 = 1$
8.1.23.2	$(0,1,1,1,1,0,1,1,1,1,0,0,0,0,0,0)$	$x_1 = 1$	8.1.25.1	$(0,1,0,0,1,1,1,1,1,1,1,0,0,0,0,0)$	$x_3 = 0$
8.1.26.1	$(1,1,1,1,1,0,0,1,0,1,1,0,0,0,0,0)$	$x_1 = 0$	8.1.27.1	$(0,1,0,1,1,0,0,1,1,1,1,0,1,0,0,0)$	$x_1 = 1$
8.1.28.1	$(0,1,1,0,1,1,0,1,1,1,1,0,0,0,0,0)$	$x_1 = 1$	8.1.28.2	$(0,0,1,1,0,1,0,1,1,1,1,0,1,0,0,0)$	$x_4 = 0$
8.1.30.1	$(0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0)$	$x_4 = 0$	8.1.31.1	$(0,0,1,1,1,1,0,1,1,1,1,0,0,0,0,0)$	$x_2 = 1$
8.2.5.1	$(1,0,1,1,1,1,0,1,0,1,1,0,0,0,0,0)$	$x_1 = 0$	8.2.10.1	$(1,1,0,1,1,0,0,1,0,1,1,0,1,0,0,0)$	$x_1 = 1$
8.2.12.1	$(0,0,1,1,1,1,0,1,0,1,1,0,1,0,0,0)$	$x_2 = 1$	8.2.16.1	$(0,1,1,1,1,0,0,1,0,1,1,0,1,0,0,0)$	$x_2 = 1$
8.2.22.1	$(0,0,1,0,0,1,1,1,1,1,0,1,1,0,0,0)$	$x_4 = 0$	8.3.3.1	$(1,0,1,1,0,1,0,1,0,1,1,0,1,0,0,0)$	$x_2 = 1$

Таблица 2

Геометрические типы булевых функций от четырёх переменных, принадлежащие классу АПФ

№ п/п	Номер класса $a.b.c.d$	Вектор-столбец представителя геометрического типа $a.b.c.d$	Структура представителя геометрического типа $a.b.c.d$
1	0.0.1.1	$(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$	$((0,0,0,0,0);1;1)$
2	1.1.1.1	$(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$	$((4,1,1,1,1);4;5)$
3	2.1.1.1	$(1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$	$((3,0,1,1,1);3;4)$
4	2.2.1.1	$(0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0)$	$((0,5,5,2,2);5;6)$
5	2.2.1.2	$(0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0)$	$((0,1,4,5,3);5;6)$
6	2.2.1.3	$(0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0)$	$((0,1,1,1,3);3;4)$
7	3.1.1.1	$(1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0)$	$((5,1,1,2,2);5;7)$
8	3.2.1.1	$(1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0)$	$((7,3,7,1,4);7;9)$
9	3.2.1.2	$(1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0)$	$((5,5,5,5,1);5;7)$
10	3.3.1.1	$(0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0)$	$((0,4,4,4,3);4;5)$
11	3.3.1.2	$(0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0)$	$((0,2,2,4,4);4;5)$
12	3.3.1.3	$(0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0)$	$((1,6,5,5,7);7;9)$
13	4.1.1.1	$(1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)$	$((2,0,0,1,1);2;3)$
14	4.1.2.1	$(1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0)$	$((4,1,1,1,2);4;6)$
15	4.2.1.1	$(0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0)$	$((0,6,6,5,4);5;7)$
16	4.2.1.2	$(0,0,0,1,0,1,0,1,1,0,0,0,0,0,0,0)$	$((1,2,1,1,4);4;6)$
17	4.2.1.4	$(1,0,0,1,1,0,0,0,1,0,0,0,0,0,0,0)$	$((5,4,4,1,1);5;7)$
18	4.2.2.1	$(0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0)$	$((0,0,3,3,2);3;4)$
19	4.2.2.3	$(0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0)$	$((0,0,1,1,2);2;3)$
20	4.3.1.1	$(0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0)$	$((1,6,7,5,5);6;8)$
21	4.3.1.2	$(0,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0)$	$((1,1,5,4,6);6;8)$
22	4.3.1.3	$(0,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0)$	$((0,3,3,6,5);5;7)$
23	4.4.1.1	$(1,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0)$	$((3,2,2,2,1);3;4)$
24	4.4.1.2	$(0,1,1,0,1,0,0,0,1,0,0,0,0,0,0,0)$	$((0,3,3,3,3);3;4)$
25	4.4.1.3	$(0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0)$	$((1,5,5,4,6);6;8)$
26	4.4.1.4	$(0,0,0,1,0,1,1,0,1,0,0,0,0,0,0,0)$	$((0,2,2,2,4);4;5)$
27	4.4.1.5	$(0,0,0,0,1,0,0,1,0,1,1,0,0,0,0,0)$	$((0,2,2,3,1);3;4)$
28	4.4.1.6	$(0,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0)$	$((0,1,1,2,2);3;4)$
29	5.1.1.1	$(1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0)$	$((5,1,1,2,3);5;8)$
30	5.1.2.1	$(1,1,1,0,1,0,0,0,1,0,0,0,0,0,0,0)$	$((3,1,1,1,1);3;5)$
31	5.1.4.1	$(1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0)$	$((6,8,2,2,4);6;9)$
32	5.2.1.1	$(0,1,0,1,0,1,0,1,1,0,0,0,0,0,0,0)$	$((2,3,1,1,5);5;8)$
33	5.2.2.3	$(0,0,0,1,0,1,1,1,1,0,0,0,0,0,0,0)$	$((1,1,1,1,3);3;5)$
34	5.2.4.2	$(1,0,0,1,1,0,0,1,1,0,0,0,0,0,0,0)$	$((6,5,5,1,2);6;9)$
35	5.2.4.3	$(1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0)$	$((5,7,5,5,2);5;8)$
36	5.3.1.1	$(0,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0)$	$((0,5,5,4,4);4;6)$
37	5.3.1.3	$(0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,0)$	$((1,4,5,5,3);4;6)$
38	5.3.1.4	$(0,1,0,0,0,1,1,0,0,1,1,0,0,0,0,0)$	$((1,5,4,2,2);6;9)$
39	5.3.2.1	$(0,0,1,1,1,1,0,0,1,0,0,0,0,0,0,0)$	$((1,1,4,4,5);5;7)$
40	5.4.1.1	$(1,0,0,1,0,1,1,0,1,0,0,0,0,0,0,0)$	$((5,4,4,4,1);5;7)$
41	5.4.1.2	$(0,0,0,1,0,1,1,0,1,1,0,0,0,0,0,0)$	$((1,1,2,2,3);4;6)$
42	5.4.1.3	$(1,0,0,0,1,0,0,1,0,1,1,0,0,0,0,0)$	$((5,3,3,1,5);5;7)$
43	5.5.1.1	$(0,1,1,0,1,0,0,1,1,0,0,0,0,0,0,0)$	$((1,4,4,4,5);5;7)$
44	5.5.1.2	$(1,0,0,0,0,0,0,1,0,1,1,0,1,0,0,0)$	$((2,1,1,1,2);2;3)$
45	5.5.1.3	$(1,0,0,0,0,1,1,0,0,1,1,0,0,0,0,0)$	$((5,3,3,5,5);5;7)$
46	6.1.1.1	$(1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0)$	$((3,0,1,1,2);3;5)$
47	6.1.2.1	$(1,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0)$	$((4,1,1,2,2);4;7)$
48	6.1.7.1	$(0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0)$	$((0,5,5,5,3);4;6)$
49	6.2.1.2	$(0,1,0,1,0,1,1,1,1,0,0,0,0,0,0,0)$	$((1,5,6,6,3);4;7)$
50	6.2.2.2	$(1,0,0,0,0,0,0,1,0,0,0,1,0,1,1,1)$	$((2,3,3,3,3);2;4)$
51	6.2.4.2	$(0,1,1,1,0,1,1,0,1,0,0,0,0,0,0,0)$	$((1,6,6,7,4);5;8)$
52	6.2.4.3	$(0,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0)$	$((0,7,6,6,5);5;8)$

Окончание табл. 2

№ п/п	Номер класса $a.b.c.d$	Вектор-столбец представителя геометрического типа $a.b.c.d$	Структура представителя геометрического типа $a.b.c.d$
53	6.2.5.1	$(1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0)$	$((3,0,3,3,1);3;5)$
54	6.2.6.2	$(0,0,1,0,1,0,1,1,1,1,0,0,0,0,0,0)$	$((1,1,5,5,3);4;7)$
55	6.2.8.1	$(0,1,1,0,0,1,1,0,0,1,1,0,0,0,0,0)$	$((1,3,3,1,1);4;6)$
56	6.2.8.3	$(0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0)$	$((0,1,1,2,3);3;5)$
57	6.3.4.1	$(0,0,1,1,0,1,1,0,1,1,0,0,0,0,0,0)$	$((2,1,3,2,4);5;8)$
58	6.3.5.1	$(0,0,1,1,1,1,0,0,1,1,0,0,0,0,0,0)$	$((0,0,2,2,2);2;3)$
59	6.4.1.2	$(0,0,0,1,1,1,1,0,0,1,1,0,0,0,0,0)$	$((1,2,2,4,3);5;8)$
60	6.4.1.3	$(1,1,0,0,0,0,0,1,0,1,1,0,1,0,0,0)$	$((3,1,2,2,4);3;5)$
61	6.4.2.1	$(0,1,1,0,1,0,0,1,0,1,1,0,0,0,0,0)$	$((1,3,3,4,1);4;6)$
62	6.4.2.2	$(0,1,1,0,0,0,0,1,0,1,1,0,1,0,0,0)$	$((0,4,4,3,1);4;6)$
63	6.5.1.1	$(1,0,0,1,0,0,0,1,0,1,1,0,1,0,0,0)$	$((3,3,3,4,2);3;5)$
64	6.6.1.1	$(1,0,0,1,0,1,1,0,0,1,1,0,0,0,0,0)$	$((4,3,3,4,4);4;6)$
65	6.6.1.2	$(0,0,0,1,0,1,1,0,0,1,1,0,1,0,0,0)$	$((0,1,1,1,1);2;3)$
66	7.1.1.1	$(1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0)$	$((4,1,1,1,3);4;7)$
67	7.1.2.1	$(1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0)$	$((5,1,2,2,3);5;9)$
68	7.1.14.1	$(1,0,1,1,1,1,0,0,1,1,0,0,0,0,0,0)$	$((4,6,2,2,2);4;7)$
69	7.2.1.1	$(0,1,1,1,0,1,1,1,1,0,0,0,0,0,0,0)$	$((1,7,7,8,4);5;9)$
70	7.2.2.2	$(1,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1)$	$((3,5,5,4,4);3;6)$
71	7.2.7.1	$(0,1,1,1,1,1,1,0,1,0,0,0,0,0,0,0)$	$((0,4,4,4,3);3;5)$
72	7.2.10.1	$(0,0,1,1,1,0,1,1,1,1,0,0,0,0,0,0)$	$((1,1,6,7,4);5;9)$
73	7.2.11.1	$(1,0,0,0,0,0,0,1,1,0,0,1,0,1,1,1)$	$((4,2,2,2,1);3;6)$
74	7.2.13.1	$(0,0,1,1,1,1,1,0,1,1,0,0,0,0,0,0)$	$((1,6,5,5,4);4;7)$
75	7.2.14.1	$(0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0)$	$((1,1,1,3,4);4;7)$
76	7.2.15.1	$(1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0)$	$((5,7,7,5,3);5;9)$
77	7.3.4.3	$(1,1,1,0,0,0,0,1,0,1,1,0,1,0,0,0)$	$((4,2,2,3,6);4;7)$
78	7.3.6.1	$(0,0,0,1,1,1,1,0,1,1,1,0,0,0,0,0)$	$((1,1,1,2,2);3;5)$
79	7.4.1.2	$(0,1,0,1,0,1,1,0,0,1,1,0,1,0,0,0)$	$((1,3,2,2,2);4;7)$
80	7.4.2.1	$(1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,0)$	$((5,3,3,1,5);4;7)$
81	7.4.3.1	$(0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,0)$	$((0,3,3,3,1);3;5)$
82	7.5.1.1	$(1,0,0,1,0,1,0,1,0,1,1,0,1,0,0,0)$	$((4,4,5,5,3);4;7)$
83	7.7.1.1	$(1,0,0,1,0,1,1,0,0,1,1,0,1,0,0,0)$	$((3,3,3,3,3);3;5)$
84	8.1.1.1	$(1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0)$	$((1,0,0,0,1);1;2)$
85	8.1.2.1	$(1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,0)$	$((3,1,1,1,2);3;6)$
86	8.1.5.1	$(1,1,1,1,1,1,0,0,1,1,0,0,0,0,0,0)$	$((2,0,1,1,1);2;4)$
87	8.1.16.1	$(0,1,1,1,1,1,1,0,1,1,0,0,0,0,0,0)$	$((0,7,6,6,5);4;8)$
88	8.1.29.1	$(0,1,0,1,1,1,1,0,1,1,1,0,0,0,0,0)$	$((1,6,7,5,5);4;8)$
89	8.2.1.1	$(0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0)$	$((0,5,5,5,3);3;6)$
90	8.2.9.2	$(0,1,1,1,0,1,1,0,0,1,1,0,1,0,0,0)$	$((1,3,3,2,2);4;8)$
91	8.2.11.1	$(1,1,1,0,1,0,0,1,0,1,1,0,1,0,0,0)$	$((3,2,2,2,5);3;6)$
92	8.2.13.1	$(0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0)$	$((0,0,3,3,2);2;4)$
93	8.2.17.1	$(0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0)$	$((1,1,1,2,3);3;6)$
94	8.2.20.1	$(0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0)$	$((0,0,0,1,1);1;2)$
95	8.2.21.1	$(0,0,0,1,0,1,1,1,1,1,1,0,1,0,0,0)$	$((0,1,1,1,2);2;4)$
96	8.3.4.1	$(1,1,1,0,0,1,1,1,0,0,0,1,1,0,0,0)$	$((2,1,1,3,2);2;4)$
97	8.3.5.1	$(0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,0)$	$((0,4,4,1,3);3;6)$
98	8.4.2.1	$(1,0,0,0,0,1,1,1,0,1,1,1,1,0,0,0)$	$((2,3,3,2,2);2;4)$
99	8.4.3.1	$(1,1,0,0,0,0,1,1,0,0,1,1,1,1,0,0)$	$((1,0,1,1,1);1;2)$
100	8.5.1.1	$(1,0,0,1,0,1,1,1,0,1,1,0,1,0,0,0)$	$((3,4,4,4,3);3;6)$
101	8.8.1.1	$(0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0)$	$((0,1,1,1,1);1;2)$

Например, в [11, 14] предлагается использовать векторы параметров Чоу. Если f — булева функция от n переменных, то, сложив как векторы все наборы x, на которых $f(x)=0$, получим целочисленный n-мерный вектор $(s_1, s_2, \ldots, s_n)$. Дополнив его нулевой координатой $s_0 = |f^{-1}(0)|$, равной числу наборов x, на которых $f(x) = 0$, получим $(n+1)$-мерный вектор параметров Чоу. В работе [11] приведено одно из доказательств того, что если две пороговые функции различны, то они имеют разные векторы параметров Чоу. Булевы функции от двух переменных $f = x_1 + x_2$ и $g = x_1 + x_2 + 1$ имеют одинаковые векторы параметров Чоу, в то время как в работе [9] доказано, что любая линейная функция k-значной логики принадлежит классу АПФ. Поэтому использование подхода на основе векторов параметров Чоу невозможно.

В связи с тем, что ограничение на коэффициенты линейной формы не получено и гарантии, что на компьютере построены все представители геометрических типов булевых функций от четырёх переменных, нет, для оставшихся 51 геометрических типов (табл. 3) выдвигается гипотеза об их непринадлежности классу АПФ.

Таблица 3

Геометрические типы булевых функций от четырёх переменных, по отношению к которым выдвигается гипотеза об их непринадлежности к классу АПФ

Номер $a.b.c.d$	Вектор-столбец представителя геометрического типа $a.b.c.d$	Номер $a.b.c.d$	Вектор-столбец представителя геометрического типа $a.b.c.d$
4.2.1.3	$(1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0)$	4.2.2.2	$(1,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0)$
5.2.2.1	$(1,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0)$	5.2.2.2	$(1,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0)$
5.2.4.1	$(1,0,0,1,0,1,0,1,1,0,0,0,0,0,0,0)$	5.3.1.2	$(0,1,0,1,1,0,0,1,1,0,0,0,0,0,0,0)$
5.3.1.5	$(0,0,0,0,1,1,0,1,0,1,1,0,0,0,0,0)$	5.3.2.2	$(0,0,0,1,1,0,1,0,1,1,0,0,0,0,0,0)$
6.1.4.1	$(1,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0)$	6.1.4.2	$(1,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0)$
6.2.1.1	$(0,1,1,1,0,1,0,1,1,0,0,0,0,0,0,0)$	6.2.2.1	$(1,1,1,0,1,0,0,1,1,0,0,0,0,0,0,0)$
6.2.4.1	$(0,0,1,1,1,1,0,1,1,0,0,0,0,0,0,0)$	6.2.6.1	$(1,0,0,1,0,1,1,1,1,0,0,0,0,0,0,0)$
6.2.8.2	$(0,0,0,0,1,1,0,1,1,1,1,0,0,0,0,0)$	6.2.8.4	$(0,1,1,0,0,0,1,1,1,1,0,0,0,0,0,0)$
6.2.8.5	$(0,1,1,0,1,0,1,0,1,1,0,0,0,0,0,0)$	6.3.1.1	$(0,0,0,0,1,1,1,1,0,1,1,0,0,0,0,0)$
6.3.2.1	$(1,1,0,0,0,1,1,0,0,1,1,0,0,0,0,0)$	6.3.2.2	$(0,1,1,1,1,0,0,1,1,0,0,0,0,0,0,0)$
6.3.4.2	$(0,0,1,0,1,1,0,1,0,1,1,0,0,0,0,0)$	6.4.1.1	$(1,0,0,1,0,1,1,0,1,1,0,0,0,0,0,0)$
6.4.2.3	$(0,0,0,1,1,0,0,1,0,1,1,0,1,0,0,0)$	7.1.4.1	$(1,1,1,0,1,0,1,0,1,1,0,0,0,0,0,0)$
7.1.6.1	$(1,1,1,1,1,0,0,1,1,0,0,0,0,0,0,0)$	7.1.12.1	$(1,1,1,0,0,1,1,0,0,1,1,0,0,0,0,0)$
7.1.15.3	$(0,1,1,1,1,0,1,0,1,1,0,0,0,0,0,0)$	7.2.2.1	$(1,1,0,1,0,1,1,0,1,1,0,0,0,0,0,0)$
7.2.4.1	$(0,1,1,1,1,1,0,1,1,0,0,0,0,0,0,0)$	7.2.4.2	$(1,1,0,0,1,1,1,0,0,1,1,0,0,0,0,0)$
7.2.15.2	$(0,1,1,0,1,0,1,1,1,1,0,0,0,0,0,0)$	7.3.1.1	$(1,0,0,0,1,1,1,1,0,1,1,0,0,0,0,0)$
7.3.2.1	$(1,1,0,1,0,1,1,0,0,1,1,0,0,0,0,0)$	7.3.4.2	$(1,0,0,1,1,1,0,1,0,1,1,0,0,0,0,0)$
7.3.6.2	$(1,0,0,0,0,1,1,1,1,0,0,1,1,0,0,0)$	7.4.1.1	$(1,0,0,1,1,1,1,0,0,1,1,0,0,0,0,0)$
8.1.3.1	$(1,1,1,1,1,1,0,1,1,0,0,0,0,0,0,0)$	8.1.6.1	$(1,1,1,1,1,0,1,0,1,1,0,0,0,0,0,0)$
8.1.10.1	$(1,1,1,0,1,0,1,1,1,1,0,0,0,0,0,0)$	8.1.17.1	$(1,1,1,1,0,1,1,0,0,1,1,0,0,0,0,0)$
8.1.20.1	$(1,0,1,0,1,1,0,1,1,1,1,0,0,0,0,0)$	8.1.22.1	$(1,0,1,1,1,1,1,0,1,1,0,0,0,0,0,0)$
8.1.24.1	$(1,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0)$	8.1.30.2	$(0,1,1,0,1,1,1,0,1,1,1,0,0,0,0,0)$
8.2.6.1	$(1,1,0,1,1,1,1,0,0,1,1,0,0,0,0,0)$	8.2.9.1	$(1,0,0,1,1,1,1,0,1,1,1,0,0,0,0,0)$
8.2.11.2	$(0,1,1,0,1,0,0,1,1,1,1,0,1,0,0,0)$	8.2.12.2	$(1,1,0,0,0,1,1,1,1,0,0,1,1,0,0,0)$
8.2.21.2	$(1,0,0,1,0,1,1,1,1,1,1,0,0,0,0,0)$	8.3.2.1	$(1,0,0,1,1,1,1,1,0,1,1,0,0,0,0,0)$
8.4.1.1	$(1,1,0,1,0,1,1,0,0,1,1,0,1,0,0,0)$		

Вероятно, для каждого геометрического типа можно доказать его непринадлежность классу АПФ, но случай булевых функций от трёх переменных [9] показал, что данный процесс весьма затруднителен.

2. Представление подстановок алгоритма блочного шифрования Магма линейными комбинациями АПФ

В стандарте ГОСТ Р 34.12-2015 [6] в качестве нелинейного биективного преобразования выступает набор подстановок π_i, $i = 0, 1, 2, \ldots, 7$. Они заданы в виде массивов

$$
\begin{aligned}
\pi_0 &= (12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1);\\
\pi_1 &= (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15);\\
\pi_2 &= (11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0);\\
\pi_3 &= (12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11);\\
\pi_4 &= (7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12);\\
\pi_5 &= (5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0);\\
\pi_6 &= (8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7);\\
\pi_7 &= (1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2).
\end{aligned}
$$

Представим подстановки в виде координатных функций, обозначив через f_3^i, f_2^i, f_1^i, f_0^i координатные функции подстановки π_i, $i = 0, \ldots, 7$, от старших разрядов к младшим соответственно:

$$
\pi_0 = \begin{pmatrix} f_3^0 \\ f_2^0 \\ f_1^0 \\ f_0^0 \end{pmatrix} = \begin{pmatrix}
1&0&0&0&1&0&1&1&1&1&1&0&0&0&1&0\\
1&1&1&0&0&1&0&0&1&0&1&1&0&0&1&0\\
0&0&1&1&1&0&1&0&1&0&0&1&0&1&1&0\\
0&0&0&0&0&1&1&1&0&0&1&1&0&1&1&1
\end{pmatrix}; \qquad
\pi_1 = \begin{pmatrix} f_3^1 \\ f_2^1 \\ f_1^1 \\ f_0^1 \end{pmatrix} = \begin{pmatrix}
0&1&0&0&1&1&0&1&0&1&0&0&1&1&0&1\\
1&0&0&0&0&0&1&1&0&1&1&1&0&1&0&1\\
1&0&1&1&0&1&0&0&0&1&0&1&1&0&0&1\\
0&0&0&1&1&0&1&0&1&0&0&1&1&1&0&1
\end{pmatrix};
$$

$$
\pi_2 = \begin{pmatrix} f_3^2 \\ f_2^2 \\ f_1^2 \\ f_0^2 \end{pmatrix} = \begin{pmatrix}
1&0&0&1&0&1&1&1&1&0&0&0&1&1&0&0\\
0&0&1&0&0&1&0&1&1&0&1&1&1&0&1&0\\
1&1&0&0&1&1&1&0&1&0&1&0&0&0&1&0\\
1&1&1&0&0&1&0&1&0&1&1&0&0&1&0&0
\end{pmatrix}; \qquad
\pi_3 = \begin{pmatrix} f_3^3 \\ f_2^3 \\ f_1^3 \\ f_0^3 \end{pmatrix} = \begin{pmatrix}
1&1&0&0&1&0&1&0&0&0&1&0&0&1&1&1\\
1&0&0&0&1&1&1&1&0&0&1&0&1&1&0&0\\
0&0&1&0&0&0&1&1&1&0&1&0&1&1&0&1\\
0&0&0&1&1&0&1&0&1&0&0&1&1&0&1&1
\end{pmatrix};
$$

$$
\pi_4 = \begin{pmatrix} f_3^4 \\ f_2^4 \\ f_1^4 \\ f_0^4 \end{pmatrix} = \begin{pmatrix}
0&1&0&1&1&0&0&1&0&1&0&1&1&0&0&1\\
1&1&1&0&0&0&1&1&0&0&0&1&0&1&0&1\\
1&1&0&1&0&0&1&0&0&0&1&1&1&0&1&0\\
1&1&1&0&0&1&0&1&0&1&1&0&1&0&0&0
\end{pmatrix}; \qquad
\pi_5 = \begin{pmatrix} f_3^5 \\ f_2^5 \\ f_1^5 \\ f_0^5 \end{pmatrix} = \begin{pmatrix}
0&1&1&0&1&0&1&1&1&0&1&0&0&0&1&0\\
1&1&1&1&0&0&1&0&0&1&0&0&1&0&1&0\\
0&0&1&1&0&1&0&1&1&1&0&0&0&1&1&0\\
1&1&1&0&1&0&0&0&1&1&0&1&0&1&0&0
\end{pmatrix};
$$

$$
\pi_6 = \begin{pmatrix} f_3^6 \\ f_2^6 \\ f_1^6 \\ f_0^6 \end{pmatrix} = \begin{pmatrix}
1&1&0&0&0&1&0&1&1&1&1&0&1&1&0&0\\
0&1&0&1&1&0&0&1&1&1&0&0&1&0&0&1\\
0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&1\\
0&0&0&1&0&1&1&0&1&0&1&0&1&0&1&1
\end{pmatrix}; \qquad
\pi_7 = \begin{pmatrix} f_3^7 \\ f_2^7 \\ f_1^7 \\ f_0^7 \end{pmatrix} = \begin{pmatrix}
0&0&1&1&0&0&1&0&0&1&1&0&1&1&1&0\\
0&1&1&1&0&1&0&0&1&1&0&1&0&1&0&0\\
0&1&1&0&0&0&0&1&0&1&1&1&0&0&1&1\\
1&1&0&1&0&1&0&1&0&1&0&0&1&0&1&0
\end{pmatrix}.
$$

Задача данного пункта — представить подстановки $\pi_0, \pi_1, \ldots, \pi_7$ через линейные комбинации функций из класса АПФ, использовав при этом для каждой подстановки минимальное количество таких функций, а значит, и минимальное количество пороговых элементов при технической реализации. Минимальное количество функций, через линейные комбинации которых можно реализовать подстановки, равно 4. В этом случае соответствующие АПФ лежат в подпространстве, порождённом координатными функциями подстановок. На первом этапе была исследована представимость каждой подстановки через АПФ линейных комбинаций координатных функций. Результаты приведены в табл. 4.

Таблица 4

Представление линейных комбинаций координатных функций подстановок ГОСТ Р 34.12-2015 через АПФ

π_i	Линейные комбинации координатных функций	Структура АПФ, задающей линейную комбинацию координатных функций	π_i	Линейные комбинации координатных функций	Структура АПФ, задающей линейную комбинацию координатных функций
π_1	$f_3^{(1)}$	$((0,3,1,3,0);2;4)$	π_4	$f_3^{(4)}$	$((0,2,1,3,0);2;4)$
	$f_3^{(1)} \oplus f_2^{(1)} \oplus f_0^{(1)}$	$((7,5,1,3,6);4;8)$		$f_3^{(4)} \oplus f_2^{(4)} \oplus f_1^{(4)}$	$((0,5,5,6,2);4;8)$
	$f_3^{(1)} \oplus f_2^{(1)}$	$((6,1,3,6,3);4;8)$		$f_3^{(4)} \oplus f_2^{(4)} \oplus f_0^{(4)}$	$((0,6,1,6,3);4;8)$
	$f_2^{(1)} \oplus f_0^{(1)}$	$((6,3,3,1,6);4;8)$	π_5	$f_2^{(5)} \oplus f_0^{(5)}$	$((0,3,2,5,7);4;8)$
π_2	$f_2^{(2)}$	$((0,3,7,2,5);4;8)$		$f_2^{(5)} \oplus f_1^{(5)}$	$((6,7,5,2,6);4;8)$
	$f_1^{(2)} \oplus f_0^{(2)}$	$((1,2,5,6,3);4;8)$		$f_3^{(5)} \oplus f_2^{(5)}$	$((5,5,5,7,2);4;8)$
	$f_3^{(2)} \oplus f_2^{(2)}$	$((7,2,5,2,1);4;8)$		$f_3^{(5)} \oplus f_2^{(5)} \oplus f_0^{(5)}$	$((0,7,5,3,2);4;8)$
π_3	$f_3^{(3)} \oplus f_0^{(3)}$	$((6,7,2,3,6);4;8)$	π_6	$f_3^{(6)} \oplus f_1^{(6)}$	$((7,2,7,5,2);4;8)$
π_7	$f_0^{(7)}$	$((4,3,7,6,5);4;8)$		$f_3^{(6)} \oplus f_2^{(6)} \oplus f_1^{(6)}$	$((6,1,6,5,6);4;8)$

Из табл. 4 видно, что функции $f_3^{(1)}$, $f_2^{(2)}$, $f_3^{(4)}$, $f_0^{(7)}$ являются алгебраическими пороговыми и имеют следующие задания:

$$f_3^{(1)} = 1 \;\Leftrightarrow\; r_4\,(3x_1 + x_2 + 3x_3) \geqslant 2, \quad f_2^{(2)} = 1 \;\Leftrightarrow\; r_8\,(3x_1 + 7x_2 + 2x_3 + 5x_4) \geqslant 4,$$
$$f_3^{(4)} = 1 \;\Leftrightarrow\; r_4\,(2x_1 + x_2 + 3x_3) \geqslant 2, \quad f_0^{(7)} = 1 \;\Leftrightarrow\; r_8\,(4 + 3x_1 + 7x_2 + 6x_3 + 5x_4) \geqslant 4.$$

Важно отметить, что функции $f_3^{(1)}$ и $f_3^{(4)}$ фиктивно зависят от переменной x_4 (x_4 — старший бит входного числа). Последнее влечёт ухудшение перемешивающих свойств нелинейного слоя.

У подстановки π_5 найдено 4 независимых линейных комбинаций, представимых функциями из класса АПФ. Найдём выражение координатных функций через соответствующие АПФ. Для этого разрешим относительно $f_0^{(5)}$, $f_1^{(5)}$, $f_2^{(5)}$, $f_3^{(5)}$ следующую систему:

$$\begin{pmatrix} f_2^{(5)} \oplus f_0^{(5)} \\ f_2^{(5)} \oplus f_1^{(5)} \\ f_3^{(5)} \oplus f_2^{(5)} \\ f_3^{(5)} \oplus f_2^{(5)} \oplus f_0^{(5)} \end{pmatrix} = \begin{pmatrix} \varphi_0^{(5)} \\ \varphi_1^{(5)} \\ \varphi_2^{(5)} \\ \varphi_3^{(5)} \end{pmatrix},$$

где функции $\varphi_0^{(5)}, \varphi_1^{(5)}, \varphi_2^{(5)}, \varphi_3^{(5)}$ задают соответствующие линейные комбинации из табл. 4:

$$\varphi_0^{(5)} = 1 \;\Leftrightarrow\; r_8\,(3x_1 + 2x_2 + 5x_3 + 7x_4) \geqslant 4,$$
$$\varphi_1^{(5)} = 1 \;\Leftrightarrow\; r_8\,(6 + 7x_1 + 5x_2 + 2x_3 + 6x_4) \geqslant 4,$$
$$\varphi_2^{(5)} = 1 \;\Leftrightarrow\; r_8\,(5 + 5x_1 + 5x_2 + 7x_3 + 2x_4) \geqslant 4,$$
$$\varphi_3^{(5)} = 1 \;\Leftrightarrow\; r_8\,(7x_1 + 5x_2 + 3x_3 + 2x_4) \geqslant 4.$$

Решение системы и подстановка в целом задаются следующим образом:

$$\pi_5 = \begin{pmatrix} f_3^{(5)} \\ f_2^{(5)} \\ f_1^{(5)} \\ f_0^{(5)} \end{pmatrix} = \begin{pmatrix} \varphi_0^{(5)} \oplus \varphi_3^{(5)} \\ \varphi_0^{(5)} \oplus \varphi_2^{(5)} \oplus \varphi_3^{(5)} \\ \varphi_0^{(5)} \oplus \varphi_1^{(5)} \oplus \varphi_2^{(5)} \oplus \varphi_3^{(5)} \\ \varphi_2^{(5)} \oplus \varphi_3^{(5)} \end{pmatrix}.$$

Данное представление позволяет реализовать подстановку, используя пороговые элементы, поскольку модульное сложение также принадлежит классу АПФ. Отметим, что структуры АПФ в табл. 4 получены с помощью структур, найденных в табл. 2. Основная сложность использования данной таблицы заключается в определении преобразований, переводящих представителя в заданную функцию геометрического типа. Изменения структуры при действии указанных преобразований эквивалентности описаны в [9]. Определение самого геометрического типа не вызывает затруднений при использовании оригинального каталога геометрических типов, опубликованного в [10].

У остальных подстановок подпространство, порождённое координатными функциями, не содержит базис из класса АПФ. В предположении, что все геометрические типы из табл. 3 не являются АПФ, получаем, что только одна из восьми подстановок реализуется через линейные комбинации четырёх АПФ.

Для остальных подстановок выбиралась некоторая функция из класса АПФ произвольного веса, не лежащая в подпространстве, порождённом координатными функциями. Получив базис из пяти функций, мы искали АПФ из расширенного подпространства. Из линейных комбинаций, принадлежащих классу АПФ, составлялась расширенная система, аналогичная полученной при задании подстановки π_5, и решалась относительно координатных функций подстановки. Благоприятным исходом была разрешимость данной системы и соответственно представление подстановки через линейные комбинации пяти АПФ.

Пятая функция выбиралась так, чтобы две или три координатные функции при сложении с ней образовывали функцию из класса АПФ. Один из таких способов — взять произведение двух координатных функций. Произведение функций и их дополнение до исходных имеют вес четыре, а значит, с высокой вероятностью мы получим три функции из класса АПФ. Высокая вероятность обусловливается тем, что из всех 19 геометрических типов, представители которых имеют вес 4, только 3 типа не принадлежат классу АПФ. Кроме того, функция веса 4 не лежит среди линейных комбинаций координатных функций, а значит, линейно не выражается через них.

Второй подход — выбор системы вершин куба, на которых пятая функция принимает единичные значения, так, чтобы при её булевом суммировании (в случае, если эта функция из класса АПФ) с координатными функциями подфункции результирующих функций не были эквивалентны f^* — функции от трёх переменных, которая не представляется через АПФ. Если эта функция имеет вес менее 8, то она также линейно не выражается через линейные комбинации координатных функций. Второй подход обеспечивает получение функций из геометрических типов, перечисленных в табл. 2 и 3. С учётом того, что количество геометрических типов в табл. 2 преобладает, велика вероятность набрать необходимое количество линейных комбинаций из класса АПФ.

Применение предложенных подходов позволило получить задания всех оставшихся подстановок через линейные комбинации пяти АПФ, а если верно предположение, что в табл. 3 нет геометрических типов из класса АПФ, то данные представления минимальны по количеству используемых функций. Ниже приведены реализации подстановок через АПФ без подробного описания способа получения пятой функции:

$$\pi_0 = \begin{pmatrix} \varphi_0^{(0)} \oplus \varphi_1^{(0)} \oplus \varphi_2^{(0)} \oplus \varphi_3^{(0)} \\ \varphi_0^{(0)} \oplus \varphi_1^{(0)} \\ \varphi_2^{(0)} \oplus \varphi_4^{(0)} \\ \varphi_1^{(0)} \oplus \varphi_2^{(0)} \end{pmatrix}, \quad \begin{array}{l} \varphi_0^{(0)} : ((5,4,7,3,2);5;8), \\ \varphi_1^{(0)} : ((1,5,5,7,2);4;8), \\ \varphi_2^{(0)} : ((1,6,5,3,1);6;8), \\ \varphi_3^{(0)} : ((0,1,1,3,2);3;5), \\ \varphi_4^{(0)} : ((0,3,1,3,3);2;5); \end{array}$$

$$\pi_1 = \begin{pmatrix} \varphi_0^{(1)} \\ \varphi_0^{(1)} \oplus \varphi_1^{(1)} \\ \varphi_3^{(1)} \oplus \varphi_4^{(1)} \\ \varphi_1^{(1)} \oplus \varphi_2^{(1)} \end{pmatrix}, \qquad \begin{array}{l} \varphi_0^{(1)} : ((0,3,1,3,0);2;4), \\ \varphi_1^{(1)} : ((6,1,3,6,3);4;8), \\ \varphi_2^{(1)} : ((7,5,1,3,6);4;8), \\ \varphi_3^{(1)} : ((0,2,3,1,2);3;4), \\ \varphi_4^{(1)} : ((6,4,5,2,5);6;8); \end{array}$$

$$\pi_2 = \begin{pmatrix} \varphi_0^{(2)} \oplus \varphi_1^{(2)} \\ \varphi_0^{(2)} \\ \varphi_2^{(2)} \oplus \varphi_3^{(2)} \\ \varphi_2^{(2)} \oplus \varphi_3^{(2)} \oplus \varphi_4^{(2)} \end{pmatrix}, \qquad \begin{array}{l} \varphi_0^{(2)} : ((0,3,7,2,5);4;8), \\ \varphi_1^{(2)} : ((7,2,5,2,1);4;8), \\ \varphi_2^{(2)} : ((7,5,5,2,7);6;8), \\ \varphi_3^{(2)} : ((0,5,3,6,3);5;7), \\ \varphi_4^{(2)} : ((1,2,5,6,3);4;8); \end{array}$$

$$\pi_3 = \begin{pmatrix} \varphi_0^{(3)} \oplus \varphi_1^{(3)} \oplus \varphi_3^{(3)} \\ \varphi_0^{(3)} \oplus \varphi_1^{(3)} \oplus \varphi_2^{(3)} \oplus \varphi_4^{(3)} \\ \varphi_0^{(3)} \oplus \varphi_2^{(3)} \\ \varphi_0^{(3)} \oplus \varphi_1^{(3)} \end{pmatrix}, \qquad \begin{array}{l} \varphi_0^{(3)} : ((1,3,4,1,5);6;8), \\ \varphi_1^{(3)} : ((1,3,2,4,6);5;7), \\ \varphi_2^{(3)} : ((1,3,5,3,6);5;7), \\ \varphi_3^{(3)} : ((5,1,6,5,2);4;8), \\ \varphi_4^{(3)} : ((5,6,2,3,5);4;8); \end{array}$$

$$\pi_4 = \begin{pmatrix} \varphi_0^{(4)} \\ \varphi_0^{(4)} \oplus \varphi_2^{(4)} \oplus \varphi_3^{(4)} \oplus \varphi_4^{(4)} \\ \varphi_1^{(4)} \oplus \varphi_2^{(4)} \oplus \varphi_3^{(4)} \oplus \varphi_4^{(4)} \\ \varphi_3^{(4)} \oplus \varphi_4^{(4)} \end{pmatrix}, \qquad \begin{array}{l} \varphi_0^{(4)} : ((0,2,1,3,0);2;4), \\ \varphi_1^{(4)} : ((0,5,5,6,2);4;8), \\ \varphi_2^{(4)} : ((0,6,1,6,3);4;8), \\ \varphi_3^{(4)} : ((1,3,4,1,5);6;8), \\ \varphi_4^{(4)} : ((4,2,2,4,6);3;7); \end{array}$$

$$\pi_6 = \begin{pmatrix} \varphi_0^{(6)} \oplus \varphi_3^{(6)} \oplus \varphi_4^{(6)} \\ \varphi_2^{(6)} \oplus \varphi_3^{(6)} \\ \varphi_0^{(6)} \oplus \varphi_2^{(6)} \oplus \varphi_3^{(6)} \oplus \varphi_4^{(6)} \\ \varphi_0^{(6)} \oplus \varphi_1^{(6)} \end{pmatrix}, \qquad \begin{array}{l} \varphi_0^{(6)} : ((1,3,4,1,5);6;8), \\ \varphi_1^{(6)} : ((0,4,2,1,3);5;7), \\ \varphi_2^{(6)} : ((4,6,1,3,6);4;8), \\ \varphi_3^{(6)} : ((5,7,2,3,2);4;8), \\ \varphi_4^{(6)} : ((1,1,3,6,2);3;7); \end{array}$$

$$\pi_7 = \begin{pmatrix} \varphi_0^{(7)} \oplus \varphi_2^{(7)} \\ \varphi_2^{(7)} \oplus \varphi_3^{(7)} \\ \varphi_1^{(7)} \oplus \varphi_3^{(7)} \oplus \varphi_4^{(7)} \\ \varphi_1^{(7)} \end{pmatrix}, \qquad \begin{array}{l} \varphi_0^{(8)} : ((1,3,4,1,5);6;8), \\ \varphi_1^{(8)} : ((4,3,7,6,5);4;8), \\ \varphi_2^{(8)} : ((1,7,6,2,5);4;8), \\ \varphi_3^{(8)} : ((0,4,2,0,2);3;5), \\ \varphi_4^{(8)} : ((3,1,0,3,3);3;5). \end{array}$$

3. Представление подстановок алгоритма блочного шифрования 2-ГОСТ линейными комбинациями АПФ

В работе [7] предложен алгоритм блочного шифрования 2-ГОСТ, являющийся модификацией шифрсистемы ГОСТ 28147-89 и отличающийся от последней лишь алгоритмом развертывания ключа, а также тем, что набор S-боксов фиксирован $\pi' = \pi_0 = \pi_1 = \pi_2 = \pi_3$, $\pi'' = \pi_4 = \pi_5 = \pi_6 = \pi_7$, где π_0 и π_4 заданы нижними строками подстановок

$$\pi' = (6, 10, 15, 4, 3, 8, 5, 0, 13, 14, 7, 1, 2, 11, 12, 9),$$
$$\pi'' = (14, 0, 8, 1, 7, 10, 5, 6, 13, 2, 4, 9, 3, 15, 12, 11).$$

Представим подстановки в виде координатных функций

$$\pi' = \begin{pmatrix} f_3' \\ f_2' \\ f_1' \\ f_0' \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,0\,1\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,1\,0 \\ 1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,1\,0\,0 \\ 0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,1\,0\,1\,0\,1 \end{pmatrix}, \qquad \pi'' = \begin{pmatrix} f_3'' \\ f_2'' \\ f_1'' \\ f_0'' \end{pmatrix} = \begin{pmatrix} 1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,1\,1\,1 \\ 1\,0\,0\,0\,1\,0\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0 \\ 1\,0\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,0\,1\,1\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,1 \end{pmatrix}.$$

Данная модификация предназначена для низкоресурсной реализации алгоритма блочного шифрования, поэтому задача реализации подстановок с помощью АПФ весьма актуальна.

Для данных подстановок применим приём, описанный в п. 2. В результате анализа каждую из них удалось задать через линейные комбинации пяти АПФ. При этом стоит отметить, что удалось получить линейные комбинации не более двух АПФ для задания координатной функции:

$$\pi' = \begin{pmatrix} f_3' \\ f_2' \\ f_1' \\ f_0' \end{pmatrix} = \begin{pmatrix} g_1 \oplus g_2 \\ g_3 \oplus g_4 \\ g_3 \\ g_5 \end{pmatrix}, \qquad \pi'' = \begin{pmatrix} f_3'' \\ f_2'' \\ f_1'' \\ f_0'' \end{pmatrix} = \begin{pmatrix} v_1 \oplus v_2 \\ v_1 \oplus v_3 \\ v_1 \oplus v_4 \\ v_4 \oplus v_5 \end{pmatrix},$$

где функции g_1, g_2, g_3, g_4, g_5, v_1, v_2, v_3, v_4, v_5 — алгебраические пороговые:

$$\begin{aligned}
g_1 = 1 \quad &\Leftrightarrow \quad r_9\,(x_1 + 5x_2 + 2x_3 + 2x_4) \geqslant 6, \\
g_2 = 1 \quad &\Leftrightarrow \quad r_7\,(5x_1 + 5x_2 + x_3 + 6x_4) \geqslant 2, \\
g_3 = 1 \quad &\Leftrightarrow \quad r_8\,(7 + 6x_1 + 5x_2 + 6x_3 + x_4) \geqslant 4, \\
g_4 = 1 \quad &\Leftrightarrow \quad r_8\,(2 + 5x_1 + 7x_2 + 3x_3 + 2x_4) \geqslant 4, \\
g_5 = 1 \quad &\Leftrightarrow \quad r_8\,(3 + 5x_1 + x_2 + 2x_3 + 3x_4) \geqslant 4, \\
\\
v_1 = 1 \quad &\Leftrightarrow \quad r_8\,(6 + 5x_1 + 5x_2 + 2x_3 + x_4) \geqslant 6, \\
v_2 = 1 \quad &\Leftrightarrow \quad r_8\,(3 + 2x_1 + 4x_2 + x_3 + 5x_4) \geqslant 6, \\
v_3 = 1 \quad &\Leftrightarrow \quad r_7\,(3 + x_1 + 6x_2 + 3x_3 + 4x_4) \geqslant 5, \\
v_4 = 1 \quad &\Leftrightarrow \quad r_8\,(2 + x_1 + 6x_2 + 3x_3 + 2x_4) \geqslant 4, \\
v_5 = 1 \quad &\Leftrightarrow \quad r_8\,(3x_1 + 2x_2 + 2x_3 + x_4) \geqslant 4.
\end{aligned}$$

Здесь x_1 — младший бит входного числа, x_4 — старший.

Выводы

В работе приведена типизация булевых функций от четырёх переменных относительно операций перестановки переменных, инвертирования переменных и инвертирования функции. Доказано, что 70 геометрических типов не принадлежат классу АПФ. Для оставшихся применён алгоритм поиска структур АПФ, и для 101 представителя найдены их структуры. По отношению к 51 геометрическому типу выдвинута гипотеза об их непринадлежности классу АПФ.

Рассмотрен вопрос об представимости подстановок из отечественных алгоритмов блочного шифрования Магма и 2-ГОСТ через линейные комбинации минимального количества АПФ. При этом только одна подстановка представилась линейными комбинациями четырёх АПФ в алгоритме Магма, а остальные — через линейные комбинации пяти АПФ. Указаны подходы к эффективному выбору пятой функции в систему координатных функций подстановок.

Результаты работы могут быть использованы для быстрой реализации алгоритмов блочного шифрования Магма и 2-ГОСТ на перспективной элементной базе.

ЛИТЕРАТУРА

1. *Дертоузос М.* Пороговая логика. М.: Мир, 1967.
2. *Морага К.* Многозначная пороговая логика // Оптические вычисления. М.: Мир, 1993. С. 162–182.
3. *Никонов В. Г., Сидоров Е. С.* О способе построения взаимно однозначных отображений при помощи квазиадамаровых матриц // Вестник Московского государственного университета леса — Лесной вестник. 2009. № 2(65). С. 155–157.
4. *Никонов В. Г., Сошин Д. А.* Геометрический метод построения сбалансированных k-значных пороговых функций и синтез подстановок на их основе // Образовательные ресурсы и технологии. 2014. № 2(5). С. 76–80.
5. *Сошин Д. А.* Построение подстановок на основе пороговых функций многозначной логики // Прикладная дискретная математика. 2016. № 2(32). С. 20–32.
6. ГОСТ Р 34.12-2015 Информационная технология. Криптографическая защита информации. Блочные шифры. М.: Стандартинформ, 2015.
7. *Dmukh A. A., Dygin D. M., and Marshalko G. B.* A lightweight-friendly modification of GOST block cipher // Математические вопросы криптографии. 2014. Т. 5. № 2. С. 47–55.
8. *Сошин Д. А.* Конструктивный метод синтеза сбалансированных k-значных алгебраических пороговых функций // Comp. Nanotechnol. 2015. № 4. С. 31–36.
9. *Сошин Д. А.* Представление геометрических типов булевых функций от трех переменных алгебраическими пороговыми функциями // Прикладная дискретная математика. 2016. № 1(31). С. 32–45.
10. *Никонов В. Г.* Классификация минимальных базисных представлений всех булевых функций от четырех переменных // Обозрение прикладной и промышленной математики. Сер. Дискретная математика. 1994. Т. 1. № 3. С. 458–545.
11. *Зуев Ю. А.* Комбинаторно-вероятностные и геометрические методы в пороговой логике // Дискретная математика. 1991. Т. 3. № 2. С. 47–57.
12. *Ирматов А. А.* Оценки числа пороговых функций // Дискретная математика. 1996. Т. 8. № 4. С. 92–107.
13. *Ирматов А. А., Ковиянич Ж. Д.* Об асимптотике логарифма числа пороговых функций K-значной логики // Дискретная математика. 1998. Т. 10. № 3. С. 35–56.
14. *Winder R. O.* Show parametrs in threshold logic // J. Association for Computing Machinery. 1971. V. 18. No. 2. P. 265–289.

REFERENCES

1. *Dertouzos M.* Porogovaya logika [Threshold Logic]. Moscow, Mir Publ., 1967. (in Russian)
2. *Moraga K.* Mnogoznachnaya porogovaya logika [Multiple-valued threshold logic]. Opticheskie Vychisleniya. Moscow, Mir Publ., 1993, pp. 162–182. (in Russian)
3. *Nikonov V. G. and Sidorov E. S.* O sposobe postroeniya vzaimno odnoznachnykh otobrazheniy pri pomoshchi kvaziadamarovykh matrits [About the possibility of one-to-one mappings representation by the quasi-hadamard matrixes]. Vestnik Moskovskogo Gosudarstvennogo Universiteta Lesa — Lesnoy Vestnik, 2009, no. 2(65), pp. 155–157. (in Russian)

4. *Nikonov V. G. and Soshin D. A.* Geometricheskiy metod postroeniya sbalansirovannykh k-znachnykh porogovykh funktsiy i sintez podstanovok na ikh osnove [The geometric method for constructing a balanced k-valued threshold functions and construction of substitutions based on them]. Obrazovatel'nye Resursy i Tekhnologii, 2014, no. 2(5), pp. 76–80. (in Russian)
5. *Soshin D. A.* Postroenie podstanovok na osnove porogovykh funktsiy mnogoznachnoy logiki [Constructing substitutions on the basis of threshold functions of multivalued logic]. Prikladnaya Diskretnaya Matematika, 2016, no. 2(32), pp. 20–32. (in Russian)
6. GOST R 34.12-2015 Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Blochnye shifry [GOST R 34.12-2015 Information technology. Cryptographic protection of information. Block ciphers]. Moscow, Standartinform Publ., 2015. (in Russian)
7. *Dmukh A. A., Dygin D. M., and Marshalko G. B.* A lightweight-friendly modification of GOST block cipher. Mat. Vopr. Kriptogr., 2014, vol. 5, iss. 2, pp. 47–55.
8. *Soshin D. A.* Konstruktivnyy metod sinteza sbalansirovannykh k-znachnykh algebraicheskikh porogovykh funktsiy [The constructive method for synthesis of balanced k-valued algebraic threshold functions]. Comp. Nanotechnol., 2015, no. 4, pp. 31–36. (in Russian)
9. *Soshin D. A.* Predstavlenie geometricheskikh tipov bulevykh funktsiy ot trekh peremennykh algebraicheskimi porogovymi funktsiyami [Representation of geometric types of Boolean functions in three variables by algebraic threshold functions]. Prikladnaya Diskretnaya Matematika, 2016, no. 1(31), pp. 32–45. (in Russian)
10. *Nikonov V. G.* Klassifikatsiya minimal'nykh bazisnykh predstavleniy vsekh bulevykh funktsiy ot chetyrekh peremennykh [The classification of minimal basic representations of Boolean functions of four variables]. Obozrenie Prikladnoy i Promyshlennoy Matematiki. Ser. Diskretnaya Matematika, 1994, vol. 1, no. 3, pp. 458–545. (in Russian)
11. *Zuev Yu. A.* Kombinatorno-veroyatnostnye i geometricheskie metody v porogovoy logike [Combinatorial-probability and geometric methods in threshold logic]. Diskr. Mat., 1991, vol. 3, no. 2, pp. 47–57. (in Russian)
12. *Irmatov A. A.* Otsenki chisla porogovykh funktsiy [Estimates for the number of threshold functions]. Diskr. Mat., 1996, vol. 8, iss. 4, pp. 92–107. (in Russian)
13. *Irmatov A. A., Koviyanich Zh. D.* Ob asimptotike logarifma chisla porogovykh funktsiy K-znachnoy logiki [On the asymptotics of the logarithm of the number of threshold functions of K-valued logic]. Diskr. Mat., 1998, vol. 10, iss. 3, pp. 35–56. (in Russian)
14. *Winder R. O.* Show parametrs in threshold logic. J. Association for Computing Machinery, 1971, vol. 18, no. 2, pp. 265–289.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 681.142.1+621.316

ЦЕПОЧЕЧНЫЕ СТРУКТУРЫ В ЗАДАЧАХ О РАСПИСАНИЯХ[1]

А. М. Магомедов

Дагестанский государственный университет, г. Махачкала, Россия

Разработаны алгоритмы построения однопроцессорного расписания с условиями частичного предшествования и мультипроцессорного расписания без простоев и условий частичного предшествования.

Ключевые слова: *расписание, граф, алгоритм, раскраска, сложность.*

DOI 10.17223/20710410/33/5

CHAIN STRUCTURES IN SCHEDULES TASKS

A. M. Magomedov

Dagestan State University, Makhachkala, Russia

E-mail: magomedtagir1@yandex.ru

Some algorithms for constructing the following shedules are developed: 1) one-processor schedule with a partial precedence, 2) a multiprocessor schedule without idle time and conditions of partial precedence.

Keywords: *schedule, graph, algorithm, colors, complexity.*

Введение

В исследованиях задач составления расписаний учебных занятий [1, с. 403], их приведения к «безоконному» виду [2], а также в задачах о расписаниях обслуживания множества заданий в мультипроцессорной системе без простоя процессоров и прерываний обработки каждого задания [3] нередко используются методы теории графов [4–6].

Пусть $G = (V, E)$ — связный граф. Цепью в G называется [7] такая последовательность вершин a_i и попарно различных ребер b_i (вершины могут повторяться)

$$P_0 = (a_0, b_0, a_1, b_1, \ldots, b_{k-1}, a_k), \qquad (1)$$

что каждые два соседних ребра b_{i-1} и b_i имеют общую вершину a_i; если все вершины в (1) попарно различны, то цепь называется простой. Если концевые вершины a_0 и a_n (простой) цепи (1) совпадают, то (простая) цепь (1) называется (простым) циклом. В настоящей работе обсуждается использование цепочечных структур (цепей и конкатенаций нескольких цепей): в п. 1 — для построения однопроцессорного расписания

[1]Работа выполнена при финансовой поддержке отдела математики и информатики ДНЦ РАН и проекта № 2588 в рамках базовой части госзадания Минобрнауки России.

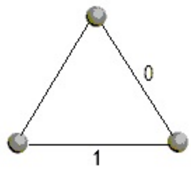
с условиями предшествования, в п. 2 — для проверки существования интервальной рёберной раскраски двудольных графов (графической интерпретации многопроцессорного расписания без простоев и условий предшествования).

Интервальной рёберной раскраской графа $G = (V, E)$ называется такое отображение множества ребер E в множество целых чисел («цветов»), при котором для каждого v из множества вершин V цвета рёбер, инцидентных v, различны и образуют целочисленный интервал (далее будем писать кратко «интервальная раскраска»).

Пример 1.

1) Простой цикл нечётной длины не имеет интервальной раскраски. Смежные рёбра на рис. 1, *а* раскрашены в цвета a и $a+1$ ($a = 0$). Тогда цвет третьего ребра при интервальной раскраске должен быть отличным от a и $a+1$ целым числом, соседним как с a, так и с $a+1$ (а такого числа не существует).

2) Простой цикл чётной длины интервально раскрашиваем (достаточно взглянуть на раскраску рёбер графа на рис. 1,*б*).



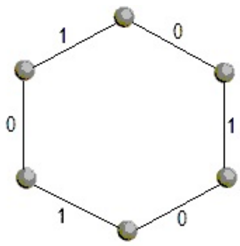


Рис. 1. Циклы нечётной длины не допускают интервальной раскраски, чётной длины — допускают

Задача об интервальной раскраске заданного графа $G = (V, E)$ NP-полна [2]. Свойство NP-полноты сохраняется и в случае двудольного графа [8]. Сложность задачи об интервальной раскрашиваемости может различаться для разных классов двудольных графов. Ниже показано, что любой регулярный двудольный граф допускает интервальную раскраску (результат Р. Камаляна [9]), при этом задача об интервальной раскрашиваемости обыкновенного регулярного графа NP-полна [2].

Общеизвестно, что текстуально близкие задачи теории графов нередко принадлежат разным классам сложности. Из примеров, приведённых в [10, с. 104], уместно указать на следующие две задачи: «Кратчайший путь между двумя вершинами» и «Самый длинный путь между двумя вершинами» в заданном графе $G = (V, E)$; первая, как известно, разрешима за время $O(|V|^2)$, вторая же NP-полна — к этой задаче сводится задача «Гамильтонов путь между двумя вершинами» [10, с. 269].

Замечание 1. Для задачи «Кратчайший путь между двумя вершинами» неизвестен алгоритм с оценкой $O(|E|)$ [11, с. 236].

Сложность задачи о гамильтоновом маршруте (цикле или пути) принципиально различна для графов разных классов. Рассмотренные далее задачи тесно связаны с вычислением гамильтоновых маршрутов в графах простой структуры: в связном регулярном графе чётной степени и ациклическом ориентированном графе.

1. Расписание выполнения заданий в однопроцессорной системе с условиями частичного предшествования

1.1. Достаточные условия интервальной раскрашиваемости

Из теоремы Эйлера [7, с. 192] следует, что связный регулярный граф G чётной степени обладает эйлеровым циклом (под степенью регулярного графа будем понимать степень вершины графа); теорема Ю. Петерсена [12] утверждает, что связный регулярный граф чётной степени $2n$ допускает разложение на n 2-факторов (каждая связная компонента 2-фактора является гамильтоновым циклом). Если каждый из этих 2-факторов является набором циклов чётной длины в графе G, то, помечая последовательные рёбра каждого цикла i-го набора метками

$$2i,\ 2i+1,\ 2i,\ 2i+1,\ \ldots\ (i = 0, 1, \ldots, n-1),$$

можно получить интервальную раскраску. Из перечисленных свойств графа сохраним лишь требование чётности длины любого цикла. Но последнее, согласно теореме Кенига, означает двудольность графа, а как уже отмечалось ранее, не всякий двудольный граф интервально раскрашиваем. Более того, проверка интервальной раскрашиваемости является NP-полной задачей, следовательно, не существует эффективно проверяемых необходимых и достаточных условий (в предположении истинности гипотезы P$\neq$NP). Поэтому для обеспечения свойства интервальной раскрашиваемости следует добавить некоторое дополнительное условие (разумеется, менее слабое, чем регулярность с чётной степенью). Потребуем выполнение свойства регулярности без условия чётности степени графа.

В случае нечётной степени Δ достаточно найти в заданном регулярном двудольном графе $G = (X, Y, E)$ полное паросочетание X с Y (существование такого паросочетания следует из следствия 8.16.1 [1, с. 165]), раскрасить его рёбра цветом $\Delta - 1$ и удалить его, затем воспользоваться теоремой Ю. Петерсена для интервальной раскраски оставшегося графа $(\Delta - 1)$ цветами: $0, 1, \ldots, \Delta - 2$. Таким образом, мы пришли к известному результату [9] об интервальной раскрашиваемости любого регулярного двудольного графа.

1.2. Перечисление допустимых расписаний с условиями частичного предшествования

Пусть однопроцессорной системе выделены для обработки объекты $0, 1, \ldots, n-1$ и заданы условия частичного предшествования: для некоторых пар объектов указано, что один из объектов пары должен быть обработан раньше другого. Требуется построить допустимое расписание обработки, другими словами, упорядочить все объекты в соответствии с условиями частичного предшествования. Рассмотрим орграф $G = (V, E)$, вершины $v_0, v_1, \ldots, v_{n-1}$ которого соответствуют заданным объектам, дуги — условиям частичного предшествования: из вершины v_i в вершину v_j дуга проведена тогда и только тогда, когда объект с номером j предшествует объекту с номером i. Если в орграфе G имеется цикл, то искомое расписание, очевидно, не существует (ориентированный граф имеет цикл в том и только в том случае, когда алгоритм поиска в глубину находит обратную дугу [13]). Если G — ациклический орграф, то для построения допустимого расписания достаточно применить алгоритм топологической сортировки [1, с. 96]. Предлагается близкий по идее алгоритм 1, на основе которого удаётся не только построить допустимое расписание, но и выполнить перечисление таких расписаний. В алгоритме используются следующие обозначения: $m(v)$ — целочисленные метки вершин $v \in V$; $L[i]$ — список всех вершин с i исходящими дугами,

$i = 0, 1, \ldots, n-1$ (сначала все списки пусты). Нетрудно видеть, что сложность алгоритма 1 не превышает $\mathrm{O}(n^2)$.

Алгоритм 1. Иерархическая сортировка вершин графа

Вход: связный ациклический орграф $G = (V, E)$ на n вершинах

Выход: такая упорядоченная последовательность T вершин множества V, что для каждой дуги $(v_i, v_j) \in E$ вершина v_j предшествует вершине v_i в последовательности T

Для всех $v \in V$ выполнить:
  $m(v) := 0$; $i := d^-(v)$; добавить v в $L[i]$.
Пока $E \neq \varnothing$:
  выбрать произвольную вершину v из $L[0]$ и дугу (u, v).
  Если $m(v) + 1 > m(u)$, **то**
    $m(u) := m(v) + 1$.
  $i := d^-(u)$.
  Удалить дугу (u, v) из E.
  Переместить вершину u из $L[i]$ в $L[i-1]$.
  Если $d^+(v) = 0$, **то**
    удалить вершину v из графа и из $L[0]$.
Восстановить исходный граф $G = (V, E)$.
$m_0 := \max\limits_{v \in V} m(v)$, $T := \varnothing$.
Для $i = 0, \ldots, m_0$ выполнить:
  Для всех $v \in V$ выполнить:
    Если $m(v) = i$, **то**
      добавить v к концу последовательности T.

Утверждение 1. Упорядочение объектов, индуцированное последовательностью T, является допустимым расписанием.

Доказательство. Отсутствие вершины с нулевой полустепенью исхода в орграфе G (или в подграфе, полученном в каждой итерации цикла) означало бы присутствие в орграфе цикла, что противоречит условию применения иерархической сортировки.

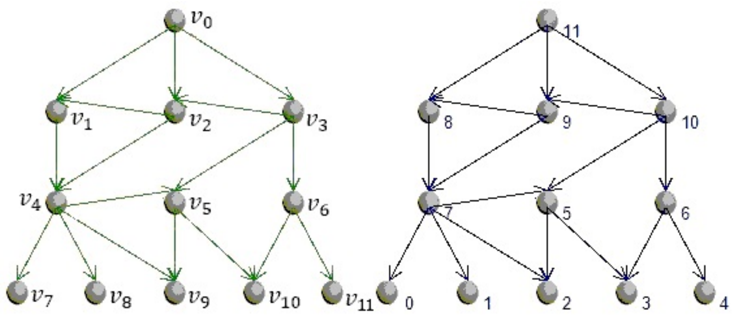
Для каждой дуги (v_i, v_j) метка, присваиваемая вершине v_i алгоритмом 1, превышает метку, присваиваемую вершине v_j: $m(v_i) \geqslant m(v_j) + 1$. Следовательно, если известно, что j-й объект предшествует i-му объекту, то в списке, индуцированном последовательностью T, объект с номером j располагается раньше объекта с номером i, т. е. получено допустимое расписание. ■

В следующем следствии через V_i обозначено множество всех вершин v с меткой $m(v) = i$, $i = 0, \ldots, m_0$.

Следствие 1. Количество допустимых расписаний равно $|V_0|! \ldots |V_{m_0}|!$

Замечание 2. Задача иерархической сортировки текстуально близка к задаче поиска ориентированного гамильтонова пути в каждой компоненте ориентированного графа. Уже отмечалось, что в общем случае задача о гамильтоновом пути NP-полна, но для ациклических ориентированных графов задача разрешима за полиномиальное время [14].

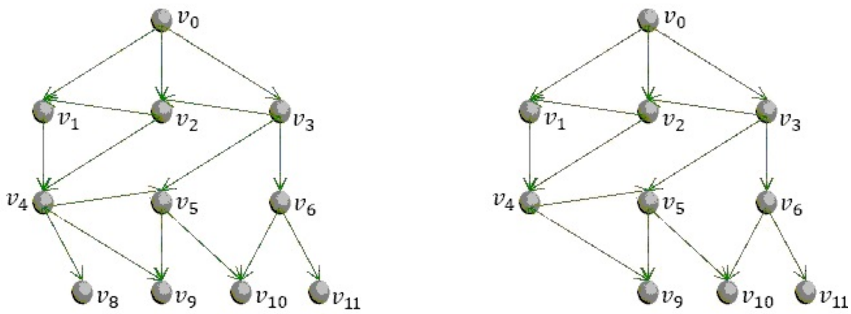
Пример 2. Пусть $n = 12$, для каждого объекта в скобках указаны предшествующие ему объекты: $0(1, 2, 3)$, $1(4)$, $2(1, 4)$, $3(2, 5, 6)$, $4(5, 7, 8, 9)$, $5(9, 10)$, $6(10, 11)$. В частности, отсюда видно, что у объектов $7, \ldots, 10$ нет предшественников. На рис. 2, *а* приведён орграф G.



а *б*

Рис. 2. Орграф G (*а*); результат алгоритма 1 — вершины G упорядочены по неубыванию меток (*б*)

На рис. 3, *а* отображены результаты выполнения первой итерации цикла 2, где выбраны вершина $v_7 \in L[0]$ и дуга (v_4, v_7); метке $m(v_4)$ присвоено значение 1, переменной i — значение 2; дуга (v_4, v_7) удалена; вершина v_4 перемещена из $L[2]$ в $L[1]$; вершина v_7 удалена из графа и из $L[0]$. Во второй итерации цикла 2 (рис. 3, *б*) для вершины v_8 выполнены аналогичные действия. Окончательный результат приведён на рис. 2, *б*.



а *б*

Рис. 3. Первая (*а*) и вторая (*б*) итерации цикла 2 алгоритма 1

2. Жадный алгоритм интервальной раскраски

2.1. Кусочно-непрерывный путь

Цепь (1) в графе G называется *непродолжимой*, если P_0 содержит все рёбра графа G, инцидентные вершине a_k. Обозначим через G' копию графа G и определим кусочно-непрерывный путь в G следующим образом.

Пока имеется ребро графа G', не входящее в P_0, выполним действия: для графа, полученного удалением из G' всех рёбер, включённых в P_0, сохраним обозначение G' и выберем a_k — самую правую из вершин P_0, имеющих в графе G' инцидентные рёбра; построим в графе G' непродолжимую цепь P_1, начинающуюся с вершины a_k, затем выполним конкатенацию P_0 и цепи P_1 и обозначим результат конкатенации через P_0. Последовательность рёбер графа G, перечисленных в том порядке, в каком они следуют в последнем P_0, будем называть кусочно-непрерывным путём в графе G и обозначать через P. Очевидно, кусочно-непрерывных путей в графе может быть несколько.

Поясним на примере графа G, приведённого на рис. 4, *а*. Выберем в G непродолжимую цепь $P_0 = (a_0, b_0, a_4, b_1, a_1, b_2, a_5, b_3, a_0, b_4, a_7, b_5, a_2, b_6, a_4)$. Затем удалим из копии G' графа G рёбра $b_0, \ldots, b_6$, включенные в P_0. Самой правой вершиной P_0, обладающей в оставшемся графе G' инцидентными рёбрами, будет вершина a_2 (инцидентные рёбра b_7 и b_8). Построим в G' непродолжимую цепь $P_1 = (a_2, b_7, a_5)$; для результата конкатенации P_0 и P_1 сохраним обозначение P_0: $P_0 = (a_0, b_0, a_4, b_1, a_1, b_2, a_5, b_3, a_0, b_4, a_7, b_5, a_2, b_6, a_4; a_2, b_7, a_5)$ — и удалим из графа G' ребро b_7. Самой правой вершиной P_0, обладающей инцидентными ребрами, снова является вершина a_2, $P_1 = (a_2, b_8, a_{11}, b_9, a_3, b_{10}, a_6, b_{11}, a_1, b_{12}, a_8, b_{13}, a_3, b_{14}, a_7)$ — непродолжимая цепь графа G', начинающаяся в вершине a_2. Результат конкатенации P_0 и P_1 обозначим через P_0: $P_0 = (a_0, b_0, a_4, b_1, a_1, b_2, a_5, b_3, a_0, b_4, a_7, b_5, a_2, b_6, a_4; a_2, b_7, a_5; a_2, b_8, a_{11}, b_9, a_3, b_{10}, a_6, b_{11}, a_1, b_{12}, a_8, b_{13}, a_3, b_{14}, a_7)$. Непродолжимая цепь, начинающаяся в вершине a_3: $P_1 = (a_3, b_{15}, a_9, b_{16}, a_1)$. Выполним конкатенацию с P_0, после чего в качестве заключительного шага выберем непродолжимую цепь $P_1 = (a_3, b_{17}, a_{10})$ и выполним её конкатенацию с P_0. В результате получим кусочно-непрерывный путь в графе G: $P = (b_0, b_1, \ldots, b_{17})$.

Замечание 3. Естественным представляется вместо «просто» непродолжимой цепи рассматривать самую длинную цепь в графе. Однако, как отмечено выше, её построение затруднено NP-полнотой соответствующей задачи распознавания.

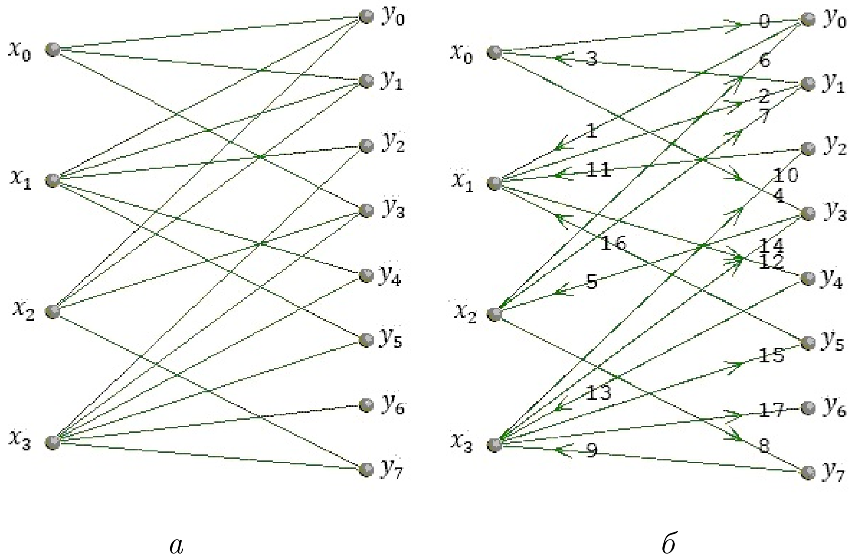


Рис. 4. Двудольный граф G (*а*); кусочно-непрерывный путь P (*б*). Номера рёбер соответствуют порядку их следования в P

2.2. Жадный алгоритм интервальной раскраски

Пусть в двудольном графе $G = (V, E)$, $V = (X, Y)$, $|E| = m$, $|V| = n$, построен кусочно-непрерывный путь P из последовательных рёбер $e_i = (v_i, w_i)$, $i = 0 \ldots, m - 1$. В прикладных задачах нередко требуется построить интервальную раскраску, удовлетворяющую дополнительному условию — все цвета принадлежат некоторому «допустимому» диапазону. Допустимый для раскрашивания рёбер диапазон цветов обозначим через $[-t..t]$, где t — заданное натуральное число. Цвет ребра e_i будем обозначать $C[i]$, $i = 0, \ldots, m - 1$. Цвета всех раскрашенных рёбер, инцидентных вершине v_i, будем называть цветами, *представленными в вершине* v_i; для их хранения используем список $L[v_i]$, $i = 0, \ldots, n - 1$.

Предлагается раскрасить последовательные рёбра кусочно-непрерывного пути P цветами из допустимого диапазона так, чтобы в каждый момент времени для каждой вершины v_i цвета, представленные в v_i, составляли (целочисленный) интервал. Алгоритм раскрашивания, обладающий таким свойством, будем называть *жадным алгоритмом.*

Для заданного непустого целочисленного интервала цветов $[a..b] = \{a, a+1, \ldots, b\}$ цвета из множества $\{a-1, b+1\} \cap [-t..t]$ будем называть *приграничными цветами*; для пустого интервала приграничным цветом считается каждый цвет из допустимого диапазона. Для ребра $e_i = (v_i, w_i)$ через $M_1 = [min_1..max_1]$ и $M_2 = [min_2..max_2]$ будем обозначать интервалы цветов, представленных в вершинах v_i и w_i, M_1' и M_2' — множества цветов, приграничных для M_1 и M_2 соответственно. Из соображений краткости мы не будем снабжать эти обозначения индексом i, поскольку всюду в дальнейшем индекс i однозначно определяется из контекста.

Утверждение 2. Пусть ребро $e_i = (v_i, w_i)$ не раскрашено. Для раскраски ребра e_i в соответствии с жадным алгоритмом может быть выбран любой цвет из пересечения $M_1' \cap M_2'$, и никакой другой.

Доказательство. Справедливость утверждения следует из того, что жадный алгоритм сохраняет свойство интервальности как для цветов, представленных в вершине v_i, так и для цветов, представленных в вершине w_i. ■

Пусть заданный допустимый интервал $[-t..t]$ достаточно большой, списки цветов $L[v_i]$ и $L[w_i]$, представленных в концевых вершинах ребра $e_i = (v_i, w_i)$, образуют непустые интервалы M_1 и M_2 соответственно. Цвет $r \in M_1' \cap M_2'$, допустимый для раскрашивания ребра e_i, может: 1) отсутствовать (например, при $M_1 = [1..2]$, $M_2 = [5..10]$); 2) определяться однозначно (например, $r = 3$ при $M_1 = [1..2]$, $M_2 = [4..5]$); 3) принимать одно из двух значений (например, при $M_1 = M_2 = [3..4]$ можно выбрать $r = 2$ или $r = 5$).

Для формулировки алгоритма 2 нам понадобится способ перебора «состояний» поиска допустимого цвета для рассматриваемого ребра e_i: $0, 1, \ldots$; текущее состояние ребра $e_i = (v_i, w_i)$ будем обозначать q_i; логическое условие, выражающее взаимное расположение интервалов M_1 и M_2 (и случаи, когда один или оба интервала M_1 и M_2 пусты), — $p_i[q_i]$; цвет, выбираемый для присвоения ребру e_i в зависимости от $p_i[q_i]$, — $c_i[q_i]$.

При принятом подходе количество таких «состояний» равно 9. Соотнесём каждому $q_i = 0, 1, \ldots, 8$ соответствующие $p_i[q_i]$ и $c_i[q_i]$.

Список состояний

$p_i[0] := (L[v_i] = \varnothing)\,\&\,(L[w_i] = \varnothing)$; $\quad c_i[0] := 0$;

$p_i[1] := (L[v_i] = \varnothing)\,\&\,(L[w_i] \neq \varnothing)\,\&\,(min_2 > -t)$; $\quad c_i[1] := min_2 - 1$;

$p_i[2] := (L[v_i] = \varnothing)\,\&\,(L[w_i] \neq \varnothing)\,\&\,(max_2 < t)$; $\quad c_i[2] := max_2 + 1$;

$p_i[3] := (L[v_i] \neq \varnothing)\,\&\,(L[w_i] = \varnothing)\,\&\,(min_1 > -t)$; $\quad c_i[3] := min_1 - 1$;

$p_i[4] := (L[v_i] \neq \varnothing)\,\&\,(L[w_i] = \varnothing)\,\&\,(max_1 < t)$; $\quad c_i[4] := max_1 + 1$;

$p_i[5] := (L[v_i] \neq \varnothing)\,\&\,(L[w_i] \neq \varnothing)\,\&\,(min_1 > -t)\,\&\,(min_1 = min_2)$; $\quad c_i[5] := min_1 - 1$;

$p_i[6] := (L[v_i] \neq \varnothing)\,\&\,(L[w_i] \neq \varnothing)\,\&\,(max_1 = max_2)\,\&\,(max_1 < t)$; $\quad c_i[6] := max_1 + 1$;

$p_i[7] := (L[v_i] \neq \varnothing)\,\&\,(L[w_i] \neq \varnothing)\,\&\,(max_1 + 2 = min_2)$; $\quad c_i[7] := max_1 + 1$;

$p_i[8] := (L[v_i] \neq \varnothing)\,\&\,(L[w_i] \neq \varnothing)\,\&\,(max_2 + 2 = min_1)$; $\quad c_i[8] := max_2 + 1$;

Например, для ребра $e_i = (v_i, w_i)$ при $q_i = 8$ условие $p_i[q_i]$ означает, что списки цветов, представленных в вершинах v_i и w_i, не пусты, интервал M_2 располагается на числовой оси левее интервала M_1, при этом наибольшее значение (max_2) из интервала M_2 на 2 меньше наименьшего значения (min_1) из интервала M_1. Поэтому множество $M_1' \cap M_2'$ цветов, приграничных как для M_1, так и для M_2, содержит единственный элемент, равный $max_2 + 1$. Понятно, что он и выбирается в качестве цвета, присваиваемого ребру e_i: $c_i[8] := max_2 + 1$.

В основе алгоритма 2 лежит идея перебора с возвратами. Каждая итерация цикла по раскрашиванию рёбер $e_i (i = 0, \ldots n-1)$ состоит из двух шагов — прямого и возвратного. Прямой шаг включает действия по раскрашиванию ребра e_i кусочно-непрерывного пути P с сохранением цветов предшествующих рёбер. Возвратный шаг выполняется, когда прямой шаг по раскрашиванию очередного ребра e_i не увенчался успехом, и включает действия по изменению цветов некоторых рёбер, предшествующих e_i, с целью подготовить условия для успешного раскрашивания ребра e_i. Нумерация рёбер соответствует порядку их следования в кусочно-непрерывном пути P.

Алгоритм 2 открывает перспективы для решения известной проблемы: верно ли, что любой двудольный граф с числом вершин $n = 15, 16, 17, 18$ допускает интервальную раскраску? Известно [15], что при $n < 15$ ответ положителен, а при $n > 18$ — отрицателен.

На рис. 5 представлена интервальная раскраска, сгенерированная алгоритмом 2 для двудольного графа рис. 4, *а*. Время счёта — приблизительно 2 мс на компьютере с характеристиками 2,5 ГГц, 8 Гб. Отметим, что первый возвратный шаг выполнен для ребра $e_7 = (x_2, y_1)$, а общее число возвратных шагов — 24.

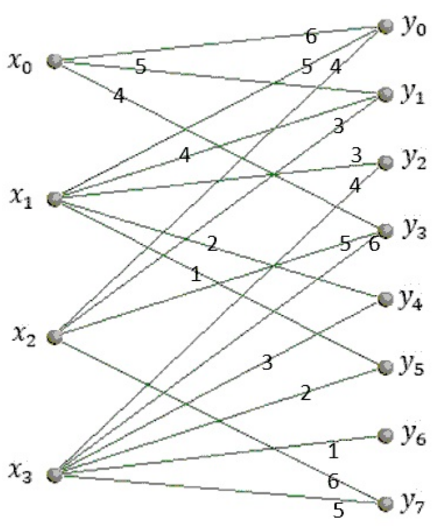


Рис. 5. Интервальная раскраска графа G

Алгоритм 2. Жадный алгоритм интервальной раскраски

Вход: $G = (V, E)$ — связный двудольный граф, $|V| = n$, $|E| = m$; кусочно-непрерывный путь P в графе G

Выход: массив $C[0..m-1]$ для цветов рёбер и логическая переменная $Success$: $Success = \texttt{true}$, если интервальная раскраска получена, и $Success = \texttt{false}$ в противном случае

1. *Инициализация*

1: **Для** $k = 0, \dots, m-1$ выполнить:
 $q_k := -1$
2: **Для** $k = 0, \dots, n-1$ выполнить:
3: $L[v_k] := \varnothing$
4: $i := 0$

2. *Прямой шаг*

5: **Если** $q_i > 8$, **то**
 переход к пункту *Возвратный шаг.*
6: **Если** $L[v_i] \neq \varnothing$, **то**
7: переменным min_1 и max_1 присвоить соответственно минимальное и максимальное значения списка $L[v_i]$.
8: **Если** $L[w_i] \neq \varnothing$, **то**
9: переменным min_2 и max_2 присвоить соответственно минимальное и максимальное значения списка $L[w_i]$.
10: **Для** $k = q_i + 1, \dots, 8$ выполнить:
11: **Если** $p_i[k] = \texttt{true}$, **то**
 переход к строке 13
12: переход к пункту *Возвратный шаг.*
13: $C[i] := c_i[k]$;
 добавить $C[i]$ в списки $L[v_i]$ и $L[w_i]$
14: **Если** $i < m - 1$, **то**
 $i := i + 1$; переход к пункту *Прямой шаг*
15: **иначе**
16: $Success := \texttt{true}$; завершить алгоритм.

3. *Возвратный шаг*

17: $q_i := -1$; $i := i - 1$;
18: **Если** $i = -1$, **то**
 $Success := \texttt{false}$; завершить алгоритм.
19: Удалить $C[i]$ из $L[v_i]$ и $L[w_i]$; $q_i := q_i + 1$; переход к пункту *Прямой шаг.*

Заключение

Алгоритмы 1 и 2 воплощены в компьютерные программы, получены свидетельства о государственной регистрации в Реестре программ для ЭВМ (2016 г.). Из п. 1.2 видно, что алгоритм иерархической сортировки вершин имеет фактически линейную вычислительную сложность относительно длины входа. Вычислительные трудности жадного алгоритма интервальной раскраски коренятся в NP-полноте задачи распознавания об интервальной рёберной раскрашиваемости двудольного графа. В худшем случае время работы данного алгоритма оценивается экспоненциально, однако компьютерные эксперименты подтверждают его эффективность для практической работы с графами, в которых число вершин не превышает 20.

ЛИТЕРАТУРА

1. *Свами М., Тхуласираман К.* Графы, сети и алгоритмы. М.: Мир, 1984. 455 с.
2. *Асратян А. С., Камалян Р. Р.* Интервальные раскраски ребер мультиграфа // Прикладная математика. Вып. 5. Ереван: Изд-во Ереванского ун-та, 1987. С. 25–34.
3. *Танаев В. С., Сотсков Ю. Н., Струсевич В. А.* Теория расписаний. Многостадийные системы. М.: Наука, 1989. 328 с.
4. *Магомедов А. М.* К вопросу об интервальной Δ-раскраске двудольных графов // Автоматика и телемеханика. 2015. № 1. С. 101–109.
5. *Магомедов А. М., Магомедов Т. А.* Реберно-вершинные инцидентные паросочетания в задачах расписаний // Прикладная дискретная математика. 2015. № 1(27). С. 92–95.
6. *Магомедов А. М., Магомедов Т. А.* Последовательное разбиение ребер двудольного графа на паросочетания // Дискретная математика. 2016. Т. 28. Вып. 1. С. 78–86.
7. *Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.* Последовательное разбиение ребер двудольного графа на паросочетания. М.: Книжный дом «Либроком», 2009. 392 с.
8. *Севастьянов С. В.* Об интервальной раскрашиваемости ребер двудольного графа // Методы дискретного анализа. 1990. Т. 50. С. 61–72.
9. *Камалян Р .Р.* Интервальные раскраски полных двудольных графов и деревьев / Препринт ВЦ АН АрмССР. Ереван, 1989. 11 с.
10. *Гэри М, Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
11. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
12. *Petersen J.* Die Theorie der regularen Graphs // Acta Math. 1891. No. 15. P. 193–220.
13. *Tucker A.* Ch. 2. Covering Circuits and Graph Colorings // Appl. Combinat. 5th Ed. Hoboken: John Wiley & Sons, 2006. P. 49.
14. *Lawler E. L.* Combinatorial Optimization: Networks and Matroids. N. Y.: Holt, Rinehart and Winston, 1976.
15. *Giaro K.* Compact task scheduling on dedicated processors with no waiting period. PhD thesis. Technical University of Gdansk, IETI Faculty, Gdansk, 1999. (in Polish)

REFERENCES

1. *Swamy M. N. S. and Thulasiraman K.* Graphs, Networks, and Algorithms. Wiley-InterScience, 1981.
2. *Asratyan A. S. and Kamalyan R. R.* Interval'nye raskraski reber mul'tigrafa [Interval coloring of multigraph edges]. Prikladnaya Matematika, iss. 5, Yerevan SU Publ., 1987, pp. 25–34. (in Russian)
3. *Tanaev V. S., Sotskov Yu. N., and Strusevich V. A.* Teoriya raspisaniy. Mnogostadiynye sistemy [Shedules Theory. The Multi-Stage Systems]. Moscow, Nauka Publ., 1989. (in Russian)
4. *Magomedov A. M.* K voprosu ob interval'noy Δ-raskraske dvudol'nykh grafov [On interval Δ-coloring of bipartite graphs]. Avtomatika i Telemekhanika, 2015, no. 1, pp. 101–109. (in Russian)

5. *Magomedov A. M. and Magomedov T. A.* Reberno-vershinnye intsidentnye parosochetaniya v zadachakh raspisaniy [Edge-vertex incident matchings in scheduling]. Prikladnaya Diskretnaya Matematika, 2015, no. 1(27), pp. 92–95. (in Russian)
6. *Magomedov A. M. and Magomedov T. A.* Posledovatel'noe razbienie reber dvudol'nogo grafa na parosochetaniya [Sequential partitioning of bipartite graph edges on matching]. Diskr. Mat., 2016, vol. 28, iss. 1, pp. 78–86. (in Russian)
7. *Emelichev V. A., Mel'nikov O. I., Sarvanov V. I., and Tyshkevich R. I.* Posledovatel'noe razbienie reber dvudol'nogo grafa na parosochetaniya [Sequential Partitioning of Bipartite Graph Edges on matching]. Moscow, Librokom Publ., 2009. (in Russian)
8. *Sevast'yanov S. V.* Ob interval'noy raskrashivaemosti reber dvudol'nogo grafa [On the interval coloring of edges of a bipartite graph]. Metody Diskretnogo Analiza, 1990, vol. 50, pp. 61–72. (in Russian)
9. *Kamalyan R .R.* Interval'nye raskraski polnykh dvudol'nykh grafov i derev'ev [Interval Coloring of Complete Bipartite Graphs and Trees]. Preprint VTs AN ArmSSR, Yerevan, 1989. (in Russian)
10. *Garey R. and Johnson D.* Computers and Intractability. N. Y., USA, W. H. Freeman & Co, 1979.
11. *Aho A., Hopcroft J., and Ullman J.* The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1974.
12. *Petersen J.* Die Theorie der regularen Graphs. Acta Math., 1891, no. 15, pp. 193–220.
13. *Tucker A.* Ch. 2. Covering Circuits and Graph Colorings. Appl. Combinat., 5th Ed., Hoboken, John Wiley & Sons, 2006, p. 49.
14. *Lawler E. L.* Combinatorial Optimization: Networks and Matroids. N. Y., Holt, Rinehart and Winston, 1976.
15. *Giaro K.* Compact task scheduling on dedicated processors with no waiting period. PhD thesis. Technical University of Gdansk, IETI Faculty, Gdansk, 1999. (in Polish)

УДК 519.1

НОВАЯ УНИВЕРСАЛЬНАЯ ОЦЕНКА ЭКСПОНЕНТОВ ГРАФОВ[1]

В. М. Фомичев

Финансовый университет при Правительстве Российской Федерации, г. Москва, Россия
Национальный исследовательский ядерный университет «МИФИ», г. Москва, Россия
Институт проблем информатики ФИЦ ИУ РАН, г. Москва, Россия

Получена новая универсальная оценка экспонентов n-вершинных примитивных орграфов через характеристики содержащихся в орграфе контуров $C_1,\ldots,C_k$ длины $l_1,\ldots,l_k$ соответственно, где $(l_1,\ldots,l_k)=1$. Показано, что $\exp\Gamma \leqslant n(r+1)+ g(l_1,\ldots,l_k)+L$, где r — число компонент связности орграфа $C_1\cup\ldots\cup C_k$; $g(l_1,\ldots,l_k)$ — число Фробениуса; L — линейная комбинация длин определённых контуров орграфа Γ. Дано уточнение оценки для некоторых частных случаев. Приведены примеры оценок экспонентов орграфов. Полученная универсальная оценка для многих n-вершинных примитивных орграфов улучшает известные оценки. Порядок её величины варьируется от $\mathrm{O}(n)$ до $\mathrm{O}(n^2)$. Оценка принимает наибольшие значения порядка $\mathrm{O}(n^2)$, только если кратчайший контур примитивной системы имеет длину порядка $\mathrm{O}(n)$. На графах Виландта полученная оценка совпадает с оценкой Виландта.

Ключевые слова: *число Фробениуса, примитивный граф, экспонент графа.*

DOI 10.17223/20710410/33/6

THE NEW UNIVERSAL ESTIMATION FOR EXPONENTS OF GRAPHS

V. M. Fomichev

Financial University under the Government of the Russian Federation, Moscow, Russia
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, Russia
The Institute of Informatics Problems of the Russian Academy of Sciences, Moscow, Russia

E-mail: fomichev@nm.ru

It is shown that, for a n-vertex primitive digraph Γ with a system of directed circuits $C_1,\ldots,C_k$ of lengths $l_1,\ldots,l_k$ respectively, $\exp\Gamma \leqslant n(r+1)+g(l_1,\ldots,l_k)+L$, where r is the number of connected components in the digraph $C_1\cup\ldots\cup C_k$, $g(l_1,\ldots,l_k)$ is the Frobenius's number, and L is a linear combination of the lengths of some directed circuits in Γ. This estimation is mostly better than other estimations known for many cases. Some more precise expressions of it are given for some particular types of graphs. The value of the estimation varies from $\mathrm{O}(n)$ to $\mathrm{O}(n^2)$ as n increases indefinitely and equals $\mathrm{O}(n^2)$, only if the length of the shortest directed circuit is $\mathrm{O}(n)$. For Wielandt's graphs, this estimation coincides with the Wielandt's one.

Keywords: *Frobenius's number, primitive graph, exponent of graph.*

[1]Работа поддержана грантом РФФИ № 16-01-00226.

Введение

Введём основные обозначения:

— $\mathbb{N}$ — множество натуральных чисел, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$;
— (l, λ) — наибольший общий делитель натуральных чисел l, λ;
— $\langle l_1, \dots, l_k \rangle$ — аддитивная полугруппа, порождённая множеством натуральных чисел $l_1, \dots, l_k$;
— $g(l_1, \dots, l_k)$ — число Фробениуса для натуральных аргументов $l_1, \dots, l_k$;
— $V(\hat{C})$ — множество вершин системы контуров $\hat{C}$ в графе;
— $w(a, b)$ — путь из вершины a в вершину b в графе;
— $w \cdot w'$ — конкатенация путей w и w', где совпадают последняя вершина пути w и первая вершина пути w';
— $\operatorname{len} w$ — длина пути w в графе.

В соответствии с известным критерием сильносвязный граф является примитивным тогда и только тогда, когда содержит систему контуров, имеющих взаимно простые длины. Абсолютная оценка экспонента [1] примитивного n-вершинного орграфа Γ использует только число вершин:

$$\exp \Gamma \leqslant n^2 - 2n + 2. \tag{1}$$

Так как в примитивном графе имеются контуры, универсальной можно считать также следующую оценку [2, с. 227]: если в Γ имеется контур длины l, то

$$\exp \Gamma \leqslant n + l(n - 2). \tag{2}$$

Последующие результаты уточняли универсальные оценки экспонента с учётом длин имеющихся в орграфе контуров. Если, в частности, в орграфе Γ имеется d петель [2, с. 408], то $\exp \Gamma \leqslant 2n - d - 1$. Если в орграфе Γ известны длины l и λ двух простых контуров C и Z [3, с. 408], где $(l, \lambda) = 1$, $1 < \lambda < l \leqslant n$, $n > 2$ и h — число общих вершин контуров C и Z, то $\exp \Gamma \leqslant l\lambda - 2l - 3\lambda + 3n$ при $h = 0$ и $\exp \Gamma \leqslant l\lambda - l - 3\lambda + h + 2n$ при $h > 0$. Список уточнений имеет продолжение [4].

Данная работа посвящена получению новой универсальной оценки экспонентов n-вершинных примитивных орграфов. Оценка использует длины ряда контуров, содержащихся в примитивном орграфе. Полученная универсальная оценка по сравнению с оценками (1) и (2) использует больше информации о графе и, как правило, улучшает их.

1. Универсальная оценка экспонента с использованием длин контуров графа

Обозначим: $\hat{C} = \{C_1, \dots, C_k\}$ — система контуров в орграфе Γ длин $l_1, \dots, l_k$ соответственно, $k \in \mathbb{N}$; $\Gamma(\hat{C}) = C_1 \cup \ldots \cup C_k$ — часть орграфа Γ.

Систему $\hat{C}$ назовём примитивной, если $(l_1, \dots, l_k) = 1$. Систему контуров $\hat{C}$ назовём связной, если граф $\Gamma(\hat{C})$ связный. В орграфе Γ контуры C и C' назовём смежными (записываем $C \sim C'$), если они имеют общую вершину.

Лемма 1. Если $\hat{C}$ — связная система контуров орграфа Γ, то в орграфе $\Gamma(\hat{C})$ имеется контур, обходящий однократно каждый контур системы $\hat{C}$.

Доказательство[2]. Индукция по числу контуров связной системы. При $k = 1$ утверждение тривиально. Если при $k = 2$ контуры C_1 и C_2 длин l_1 и l_2 смежные и i — их общая вершина, то искомый контур C есть $C_1(i) \cdot C_2(i)$.

[2]Лемма доказана совместно с Я. Э. Авезовой.

Пусть $k \geqslant 2$ и лемма выполнена для любой связной системы менее k контуров. По условию контур C_1 смежный с другим контуром системы (пусть $C_1 \sim C_2$). Тогда либо C_1, либо C_2 является смежным с некоторым из остальных контуров системы (пусть это C_3). Следовательно, система контуров $\{C_1, C_2, C_3\}$ является связной. Рассуждая аналогично, построим связную систему контуров $\{C_1, \ldots, C_{k-1}\}$. По предположению индукции в $\Gamma(\hat{C})$ существует контур C', однократно обходящий каждый из контуров $C_1, \ldots, C_{k-1}$. По условию контур C_k смежный с некоторым из контуров $C_1, \ldots, C_{k-1}$, значит, $C_k \sim C'$ и по предположению индукции в $\Gamma(\hat{C})$ существует контур C, обходящий однократно контуры C' и C_k. ■

Контур Z, обходящий однократно каждый контур связной системы $\hat{C}$, назовём квазиэйлеровым $\hat{C}$-контуром, при этом $\operatorname{len} Z = l_1 + \ldots + l_k$ в силу леммы 1.

Следствие 1. В орграфе $\Gamma(\hat{C})$ множеству $\{\hat{C}_1, \ldots, \hat{C}_r\}$ компонент связности соответствует множество $\{Z_1, \ldots, Z_r\}$ (в общем случае не единственное) независимых контуров, $1 \leqslant r \leqslant k$, где Z_j — квазиэйлеров $\hat{C}_j$-контур, $j = 1, \ldots, r$.

Доказательство. Следует из леммы 1 и свойств разбиений множеств. ■

Пусть далее $l_1 \leqslant \ldots \leqslant l_k$, $\lambda_1 \geqslant \ldots \geqslant \lambda_r$, где $\lambda_j = \operatorname{len} Z_j$, $j = 1, \ldots, r$. При $(l_1, \ldots, l_k) = 1$ обозначим $g(l_1, \ldots, l_k)$ число Фробениуса (наибольшее число, не принадлежащее полугруппе $\langle l_1, \ldots, l_k \rangle$ при $k > 1$; $g(1) = -1$).

Теорема 1. Пусть n-вершинный орграф Γ содержит примитивную систему контуров $\hat{C} = \{C_1, \ldots, C_k\}$ длин $l_1, \ldots, l_k$ соответственно и компонента связности $\hat{C}_j$ орграфа $\Gamma(\hat{C})$ содержит квазиэйлеров $\hat{C}_j$-контур Z_j длины λ_j, $j = 1, \ldots, r$. Тогда

$$\exp \Gamma \leqslant n(r+1) + g(l_1, \ldots, l_k) - \sum_{j=1}^{r} (l_j + (j-1)\lambda_j).$$

Доказательство. Для вершин a, b орграфа Γ обозначим: $w(a, a_j)$ — кратчайший путь из a до ближайшей вершины a_j контура Z_j; $w(a_j, b)$ — кратчайший путь из a_j до b, $j = 1, \ldots, r$; $mC(i)$ — контур, полученный при m-кратном проходе контура C из вершины i, где $m \geqslant 0$; $0C(i)$ — пустой контур.

Без ограничения общности положим, что контур Z_j содержит вершины контуров $C_{t_{j-1}+1}, \ldots, C_{t_j}$, и только этих контуров, где $0 = t_0 < \ldots < t_r = k$ и $\operatorname{len} Z_j = \lambda_j = l_{t_{j-1}+1} + \ldots + l_{t_j}$ в силу леммы 1, $j = 1, \ldots, r$. Перенумеруем все вершины контура Z_j числами от 1 до λ_j, где нумерация начинается с вершины a_j, и обозначим:

1) путь $w(\tau, \theta) = (\tau, \tau+1, \ldots, \theta)$ при $\tau \leqslant \theta$, путь $w(\tau, \theta)$ пустой при $\tau = \theta$;
2) ξ_i — наименьший номер вершины, принадлежащей контуру C_i, $i = t_{j-1}+1, \ldots, t_j$;
3) $v_j = \max\{\xi_{t_{j-1}+1}, \ldots, \xi_{t_j}\}$.

Так как $l_{t_{j-1}+1} \leqslant \ldots \leqslant l_{t_j}$, то $v_j \leqslant \lambda_j - l_{t_{j-1}+1} + 1$. Положив для определённости $\xi_{t_{j-1}+1} \geqslant \ldots \geqslant \xi_{t_j}$ (в общем случае не исключено любое упорядочение этих номеров), определим путь $w(Z_j)$ при неотрицательных целых числах $m_{t_j}, \ldots, m_{t_{j-1}+1}$:

$$w(Z_j) = w(1, \xi_{t_j}) \cdot m_{t_j} C_{t_j}(\xi_{t_j}) \cdot \ldots \cdot w(\xi_{t_{j-1}+2}, \xi_{t_{j-1}+1}) \cdot m_{t_{j-1}+1} C_{t_{j-1}+1}(\xi_{t_{j-1}+1}).$$

Отсюда при любом порядке указанных номеров для длины пути $w(Z_j)$ выполнено

$$\begin{aligned} \operatorname{len} w(Z_j) = \operatorname{len} w(1, \xi_{t_j}) + \ldots + \operatorname{len} w(\xi_{t_{j-1}+2}, \xi_{t_{j-1}+1}) + \\ + m_{t_j} l_{t_j} + \ldots + m_{t_{j-1}+1} l_{t_{j-1}+1} = v_j - 1 + S_j, \end{aligned} \tag{3}$$

где $v_j - 1 \leqslant \lambda_j - l_{t_{j-1}+1}$; $S_j = m_{t_j} l_{t_j} + \ldots + m_{t_{j-1}+1} l_{t_{j-1}+1}$.

Построим путь $w(a,b)$ в Γ для любых вершин a,b:

$$w(a,b) = w(a_0,a_1)\cdot w(Z_1)\cdot w(a_1,a_2)\cdot w(Z_2)\cdot\ldots\cdot w(a_{r-1},a_r)\cdot w(Z_r)\cdot w(a_r,b).$$

Здесь $a_0 = a$; $w(a_j,a_{j+1})$ — кратчайший путь из вершины a_j до ближайшей вершины a_{j+1} графа $C_{t_j+1}\cup\ldots\cup C_k$, $j=0,\ldots,r-1$; $w(a_r,b)$ — кратчайший путь из a_r до вершины b. Заметим, что $\operatorname{len} w(a_r,b)\leqslant n-1$, и с учётом разбиения графа $\Gamma(\hat{C})$ на компоненты связности

$$\operatorname{len} w(a_j,a_{j+1})\leqslant n-(\lambda_{j+1}+\ldots+\lambda_r),\quad j=0,\ldots,r-1.$$

Тогда при $\lambda_1\geqslant\ldots\geqslant\lambda_r$ получаем

$$\operatorname{len} w(a,b)\leqslant n(r+1)-\sum_{j=1}^{r} j\lambda_j-1+\sum_{j=1}^{r}\operatorname{len} w(Z_j).$$

Из равенств (3) для $j=1,\ldots,r$ следует, что при $l_1\leqslant\ldots\leqslant l_k$

$$\sum_{j=1}^{r}\operatorname{len} w(Z_j)=\sum_{j=1}^{r}(\lambda_j-l_j)+\sum_{j=1}^{r}S_j,$$

где $\sum\limits_{j=1}^{r}S_j=\sum\limits_{i=1}^{k}m_il_i$ с учётом разбиения графа $\Gamma(\hat{C})$ на компоненты связности. В силу примитивности системы контуров $\hat{C}$ сумма $\sum\limits_{i=1}^{k}m_il_i$ в зависимости от набора неотрицательных целых коэффициентов $m_1,\ldots,m_k$ принимает любое значение, большее $g(l_1,\ldots,l_k)$. Следовательно, $\operatorname{len} w(a,b)\leqslant n(r+1)+g(l_1,\ldots,l_k)-\sum\limits_{j=1}^{r}(l_j+(j-1)\lambda_j)$. ■

Следствие 2. Если примитивная система контуров $\hat{C}$ является связной, то

$$\exp\Gamma\leqslant 2n-l_1+g(l_1,\ldots,l_k).$$

Доказательство. В оценке теоремы 1 следует положить $r=1$. ■

Следствие 3. Если в условиях теоремы 1 $|V(\hat{C})|=n$, то

$$\exp\Gamma\leqslant nr+g(l_1,\ldots,l_k)-\sum_{j=1}^{r}(l_j+(j-2)\lambda_j).$$

Доказательство. Если $|V(\hat{C})|=n$, то $n=\lambda_1+\ldots+\lambda_r$ и вершина a принадлежит компоненте связности орграфа $\Gamma(\hat{C})$. Значит, $\operatorname{len} w(a,a_1)=n-(\lambda_1+\ldots+\lambda_r)=0$. Следовательно, оценку $\operatorname{len} w(a,b)$, приведённую в доказательстве теоремы 1, а также оценку экспонента орграфа Γ в условиях следствия 3 можно уменьшить на величину $n-(\lambda_1+\ldots+\lambda_r)$. ■

Примитивную систему контуров $\hat{C}$ назовём минимальной, если любая её собственная подсистема не примитивная. Система $\hat{C}$ минимальная примитивная, если и только если $l_j\notin\langle l_1,\ldots,l_{j-1}\rangle$ при $j=2,\ldots,k$ [5, с. 21].

Пример 1. Оценка экспонента перемешивающего орграфа регистра с одной обратной связью.

Рассмотрим перемешивающий орграф Γ преобразования автономного двоичного регистра левого сдвига длины 8 с функцией обратной связи $f(x_1,\ldots,x_8)=x_1\oplus x_3x_4x_6$. Зададим орграф Γ списком дуг $E_\Gamma=\{(i,i-1): i=2,\ldots,8\}\cup\{(1,8),(3,8),(4,8),(6,8)\}$. Следовательно, список всех простых контуров есть $\{C_1=(8,7,6,5,4,3,2,1),\ C_2= (8,7,6,5,4,3),\ C_3=(8,7,6,5,4),\ C_4=(8,7,6)\}$, длины контуров равны соответственно 8, 6, 5 и 3.

В орграфе Γ имеются примитивные системы контуров $\{C_1,C_2,C_3,C_4\}$, $\{C_1,C_2,C_3\}$, $\{C_1,C_2,C_4\}$, $\{C_1,C_3,C_4\}$, $\{C_1,C_3\}$, $\{C_1,C_4\}$, $\{C_2,C_3\}$ и $\{C_3,C_4\}$, все они являются связными. Минимальными являются системы контуров $\{C_1,C_3\}$, $\{C_1,C_4\}$, $\{C_2,C_3\}$ и $\{C_3,C_4\}$. Используя формулу числа Фробениуса для двух аргументов

$$g(l_1,l_2)=l_1l_2-l_1-l_2, \tag{4}$$

получаем в соответствии со следствием 2, что оценка равна: 38 для системы $\{C_1,C_3\}$, 26 для системы $\{C_1,C_4\}$, 30 для системы $\{C_2,C_3\}$ и 20 для системы $\{C_3,C_4\}$. Наименьшая оценка экспонента достигается для системы контуров $\{C_3,C_4\}$: $\exp\Gamma\leqslant 20$.

Заметим, что абсолютная оценка Виландта (1) принимает значение 50, оценка (2) принимает значение 26 при $l=3$.

Пример 2. Оценка экспонента перемешивающего орграфа регистра с двумя обратными связями.

Рассмотрим перемешивающий орграф Γ преобразования автономного двоичного регистра левого сдвига длины 8 с функциями обратной связи $f_8(x_1,\ldots,x_8)=x_1\oplus x_5$ и $f_3(x_1,\ldots,x_8)=x_1x_4$, которые определяют значения соответственно 8-й и 3-й координатных функций преобразования.

Зададим орграф Γ списком дуг $E_\Gamma=\{(i,i-1): i=2,\ldots,8\}\cup\{(1,8),(5,8),(1,3)\}$. Тогда список всех простых контуров есть $\{C_1=(8,7,6,5,4,3,2,1),\ C_2=(8,7,6,5),\ C_3=(3,2,1)\}$, длины контуров равны соответственно 8, 4 и 3.

В орграфе Γ имеются примитивные системы контуров $\{C_1,C_2,C_3\}$, $\{C_1,C_3\}$ и $\{C_2,C_3\}$. Системы $\{C_1,C_2,C_3\}$ и $\{C_1,C_3\}$ являются связными, система $\{C_2,C_3\}$ несвязная. Минимальными являются системы контуров $\{C_1,C_3\}$ и $\{C_2,C_3\}$. Используя формулу (4), получаем в соответствии со следствием 2, что оценка равна 26 для связной системы $\{C_1,C_3\}$ и 19 для несвязной системы $\{C_2,C_3\}$, во втором случае в оценке теоремы 1 следует положить $r=2$, $l_1=\lambda_2=3$, $l_2=4$. Наименьшая оценка экспонента достигается для системы контуров $\{C_2,C_3\}$: $\exp\Gamma\leqslant 19$.

Оценки (1) и (2) принимают те же значения, что и в примере 1.

Пример 3. Сравнение полученной оценки с оценкой Виландта в случае n-вершинного графа Виландта.

В n-вершинном графе Виландта имеется связная система двух контуров длины $n-1$ и n. Тогда в соответствии со следствием 2

$$\exp\Gamma\leqslant 2n-(n-1)+g(n,n-1)=n^2-2n+2,$$

что совпадает с абсолютной оценкой Виландта.

Оценим возможные значения величины k для минимальной примитивной системы. Обозначим p_i простое число с номером i (числа берутся в порядке возрастания), $i=1,2,\ldots$, то есть $p_1=2$, $p_2=3$ и т. д.

Теорема 2. *В n-вершинном орграфе Γ при $n>2$ минимальная примитивная система имеет не более k простых контуров, где k — наибольший номер, при котором $p_2\ldots p_k\leqslant n$.*

Доказательство. Наибольшее значение k достигается при наименьших взаимно простых длинах контуров $l_1, \ldots, l_k$, где $l_k \leqslant n$. Наименьшие числа, составляющие множество k взаимно простых чисел, имеют следующий вид [6, с. 9–10]: $l_i = p_1 \ldots p_k / p_i$, $i = 1, \ldots, k$. Тогда $l_k = p_2 \ldots p_k \leqslant n$. ■

Следствие 4. В условиях теоремы 1

$$\exp \Gamma - g(l_1, \ldots, l_k) + \sum_{j=1}^{r} (l_j + (j-1)\lambda_j) \leqslant \begin{cases} 3n, & 3 \leqslant n \leqslant 14, \\ 4n, & 15 \leqslant n \leqslant 104, \\ 5n, & 105 \leqslant n \leqslant 1154, \\ \lfloor \ln n \rfloor n, & 1155 \leqslant n. \end{cases}$$

Доказательство. В таблице для некоторых интервалов значений n приведены оценки k и r, полученные с помощью теоремы 2.

$3 \leqslant n \leqslant 14$	$15 \leqslant n \leqslant 104$	$105 \leqslant n \leqslant 1154$	$1155 \leqslant n \leqslant 15014$	$15015 \leqslant n \leqslant 255254$	$255255 \leqslant n \leqslant 4849844$
$1 \leqslant r \leqslant k = 2$	$1 \leqslant r \leqslant k \leqslant 3$	$1 \leqslant r \leqslant k \leqslant 4$	$1 \leqslant r \leqslant k \leqslant 5$	$1 \leqslant r \leqslant k \leqslant 6$	$1 \leqslant r \leqslant k \leqslant 7$

Оценки следствия 4 вытекают из теоремы 1 и из таблицы, так как $k \leqslant \lfloor \ln n \rfloor - 1$ при $n \geqslant 1155$. ■

Следствие 5. При $n \to \infty$ выполнена асимптотическая оценка $k = \mathrm{o}(\ln n)$.

Доказательство. Воспользуемся известной оценкой [7] $p_k > k \ln k$. Тогда из теоремы 2 следует

$$n \geqslant \prod_{j=2}^{k} j \ln j.$$

Отсюда $k! < n$ при $k > 3$. Используя формулу Стирлинга и логарифмируя неравенство $(k/e)^k < n$, получаем, что $k(\ln k - 1) < \ln n$ при $k \to \infty$. Следовательно, $k = \mathrm{o}(\ln n)$ при $n \to \infty$. ■

Замечание 1. Оценки и алгоритмы вычисления чисел Фробениуса для любого числа аргументов, требуемые при оценивании экспонентов орграфов, можно найти в ряде публикаций [8, 9].

Выводы

1) Для получения оценки экспонента графа в соответствии с теоремой 1 и её следствиями можно использовать любую примитивную систему контуров. Наименьшие значения оценок достигаются при одной из минимальных примитивных систем.
2) Порядок величины полученной универсальной оценки экспонента n-вершинного примитивного орграфа варьируется от $\mathrm{O}(n)$ до $\mathrm{O}(n^2)$ при $n \to \infty$. Оценки теоремы 1 и её следствий принимают значения порядка $\mathrm{O}(n^2)$, только если кратчайший контур примитивной системы имеет длину порядка $\mathrm{O}(n)$.
3) На множестве графов Виландта полученная оценка совпадает с абсолютной оценкой Виландта.

ЛИТЕРАТУРА

1. *Wielandt H.* Unzerlegbare nicht negative Matrizen // Math. Zeitschr. 1950. N. 52. S. 642–648.
2. *Сачков В. Н., Тараканов В. Е.* Комбинаторика неотрицательных матриц. М.: ТВП, 2000. 448 с.

3. *Фомичев В. М.* Оценки экспонентов примитивных графов // Прикладная дискретная математика. 2011. № 2(12). С. 101–112.
4. *Когос К. Г., Фомичев В. М.* Положительные свойства неотрицательных матриц // Прикладная дискретная математика. 2012. № 4(18). С. 116–121.
5. *Фомичев В. М.* Эквивалентные по Фробениусу примитивные множества чисел // Прикладная дискретная математика. 2014. № 1(23). С. 20–26.
6. *Кяжин С. Н., Фомичев В. М.* О примитивных наборах натуральных чисел // Прикладная дискретная математика. 2012. № 2(16). С. 5–14.
7. *Rosser B.* The n-th prime is greater than $n \log n$ // Proc. London Math. Soc. 1939. V. 45. P. 21–44.
8. *Alfonsin J. R.* The Diophantine Frobenius Problem. Oxford University Press, 2005.
9. *Фомичев В. М.* Оценка экспонента некоторых графов с помощью чисел Фробениуса для трёх аргументов // Прикладная дискретная математика. 2014. № 2(24). С. 88–96.

REFERENCES

1. *Wielandt H.* Unzerlegbare nicht negative Matrizen. Math. Zeitschr., 1950, n. 52, s. 642–648.
2. *Sachkov V. N. and Tarakanov V. E.* Kombinatorika neotritsatel'nykh matrits [Combinatorics of Non-Negative Matrices]. Moscow, TVP Publ., 2000. (in Russian)
3. *Fomichev V. M.* Otsenki eksponentov primitivnykh grafov [The estimates of exponents for primitive graphs]. Prikladnaya Diskretnaya Matematika, 2011, no. 2(12), pp. 101–112. (in Russian)
4. *Kogos K. G. and Fomichev V. M.* Polozhitel'nye svoystva neotritsatel'nykh matrits [Positive properties of non-negative matrices]. Prikladnaya Diskretnaya Matematika, 2012, no. 4(18), pp. 116–121. (in Russian)
5. *Fomichev V. M.* Ekvivalentnye po Frobeniusu primitivnye mnozhestva chisel [Primitive sets of numbers being equivalent by Frobenius]. Prikladnaya Diskretnaya Matematika, 2014, no. 1(23), pp. 20–26. (in Russian)
6. *Kyazhin S. N. and Fomichev V. M.* O primitivnykh naborakh natural'nykh chisel [About primitive systems of natural numbers]. Prikladnaya Diskretnaya Matematika, 2012, no. 2(16), pp. 5–14. (in Russian)
7. *Rosser B.* The n-th prime is greater than $n \log n$. Proc. London Math. Soc., 1939, vol. 45, pp. 21–44.
8. *Alfonsin J. R.* The Diophantine Frobenius Problem. Oxford University Press, 2005.
9. *Fomichev V. M.* Otsenka eksponenta nekotorykh grafov s pomoshch'yu chisel Frobeniusa dlya trekh argumentov [Estimates for exponent of some graphs by Frobenius's numbers of three arguments]. Prikladnaya Diskretnaya Matematika, 2014, no. 2(24), pp. 88–96. (in Russian)

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 512.54

О coNP-ПОЛНОТЕ ЗАДАЧИ «ИНЪЕКТИВНЫЙ РЮКЗАК»

В. Г. Дурнев, О. В. Зеткина, А. И. Зеткина, Д. М. Мурин

Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия

Устанавливается coNP-полнота задачи «Инъективный рюкзак».

Ключевые слова: *NP-полные и coNP-полные задачи, инъективный рюкзак, m-инъективный рюкзак.*

DOI 10.17223/20710410/33/7

ABOUT THE coNP-COMPLETE “INJECTIVE KNAPSACK” PROBLEM

V. G. Durnev, O. V. Zetkina, A. I. Zetkina, D. M. Murin

P. G. Demidov Yaroslavl State University, Yaroslavl, Russua

E-mail: durnev@uniyar.ac.ru, oks_68@mail.ru, aizetkina@gmail.ru, nirum87@mail.ru

It is proved that the “Injective knapsack” problem is coNP-complete.

Keywords: *NP-complete and coNP-complete problems, injective knapsack, m-injective knapsack.*

1. Задачи о рюкзаках

Хорошо известно [1–3], что «Проблема рюкзака», или «Проблема суммы элементов подмножеств», является NP-полной. Она может быть сформулирована как вопрос о разрешимости в числах 0, 1 уравнения вида

$$a_1x_1 + \ldots + a_nx_n = b, \tag{1}$$

где $a_1, \ldots, a_n$ и b — произвольные натуральные числа; $x_1, \ldots, x_n$ — неизвестные.

В 1978 г. Р. Меркль и М. Хеллман [4] предложили первую (в открытых источниках) асимметричную систему шифрования, которая базировалась на «Проблеме рюкзака». Спустя четыре года А. Шамир показал, что метод генерации открытых ключей по закрытым в криптосистеме Меркля — Хеллмана не является надёжным и практически все индивидуальные представители «Проблемы рюкзака», возникающие при использовании этой системы, поддаются решению.

Несмотря на результаты А. Шамира, «Проблема рюкзака» обладает полезными для криптографии свойствами: во-первых, о ней известно, что она является достаточно сложной, и, во-вторых, процесс шифрования в криптосистеме Меркля — Хеллмана существенно быстрее, чем процесс шифрования в криптосистеме RSA, и представляется вероятным, что можно построить стойкие криптосистемы, основанные на «Проблеме рюкзака», с высокой скоростью шифрования и расшифрования, что является крайне важным прикладным свойством, например для задачи синхронизации центров обработки данных по неконтролируемому (незащищённому) каналу связи. Благодаря этим свойствам работы по созданию криптосистем, базирующихся на «Проблеме рюкзака», не прекращаются [5–8].

В работе [7] введена в рассмотрение «Проблема обобщённого рюкзака», которая для фиксированного натурального числа $m \geqslant 2$ состоит в ответе на вопрос, разрешимо ли уравнение (1) в числах $0, 1, \ldots, m-1$.

В связи с шифрсистемой с открытым ключом, предложенной Р. Мерклем и М. Хелманом на основе «супервозрастающего» рюкзака, стали представлять особый интерес так называемые «Инъективные рюкзаки».

Будем говорить, что набор различных натуральных чисел $a_1, \ldots, a_n$ определяет «Инъективный рюкзак» (соответственно «m-инъективный рюкзак»), если не существует двух различных наборов чисел 0, 1 (соответственно наборов чисел $0, \ldots, m-1$, где натуральное число $m \geqslant 2$) $\alpha_1, \ldots, \alpha_n$ и $\beta_1, \ldots, \beta_n$, таких, что

$$a_1\alpha_1 + \ldots + a_n\alpha_n = a_1\beta_1 + \ldots + a_n\beta_n.$$

В противном случае набор натуральных чисел $a_1, \ldots, a_n$ определяет «Неинъективный рюкзак» (соответственно «m-неинъективный рюкзак»).

Задача «Инъективный рюкзак»
Дано: набор различных натуральных чисел $a_1, \ldots, a_n$.
Определить: является ли определяемый ими рюкзак инъективным.

Задача «Неинъективный рюкзак»
Дано: набор различных натуральных чисел $a_1, \ldots, a_n$.
Определить: является ли определяемый ими рюкзак неинъективным.

Задача «m-инъективный рюкзак»
Дано: набор различных натуральных чисел $a_1, \ldots, a_n$, натуральное число $m \geqslant 2$.
Определить: является ли определяемый числами $a_1, \ldots, a_n$ рюкзак m-инъективным.

Задача «m-неинъективный рюкзак»
Дано: набор различных натуральных чисел $a_1, \ldots, a_n$, натуральное число $m \geqslant 2$.
Определить: является ли определяемый числами $a_1, \ldots, a_n$ рюкзак m-неинъективным.

Очевидно, что задача «2-инъективный рюкзак» эквивалентна задаче «Инъективный рюкзак», поэтому задача «m-инъективный рюкзак» включает в себя задачу «Инъективный рюкзак».

Ясно, что задача «Неинъективный рюкзак» является дополнением задачи «Инъективный рюкзак» и принадлежит классу NP. Аналогично задача «m-неинъективный рюкзак» является дополнением задачи «m-инъективный рюкзак» и, в силу включения задачи «Неинъективный рюкзак», принадлежит классу NP.

2. Теорема о coNP-полноте задачи «Инъективный рюкзак»

Теорема 1. Задача «Инъективный рюкзак» является coNP-полной.

Доказательство. Необходимо доказать, что задача «Неинъективный рюкзак» (дополнение задачи «Инъективный рюкзак») является NP-полной. Так как она принадлежит классу NP, то остаётся доказать её NP-трудность. Для этого достаточно установить полиномиальную сводимость к ней некоторой подходящей NP-полной задачи. В качестве таковой возьмём задачу о разрешимости в числах 0, 1 уравнения вида

$$2(a_1x_1 + \ldots + a_nx_n) = \sum_{i=1}^{n} a_i, \tag{2}$$

где $a_1, \ldots, a_n$ — произвольные натуральные числа. Эта задача хорошо известна под названием «Разбиение». Столь же хорошо известна и её NP-полнота. При построении полиномиальной сводимости воспользуемся некоторыми идеями работы [9], с которой нас любезно познакомил М. В. Волков.

По коэффициентам уравнения (2) по аналогии с работой [9] построим последовательность из $3n$ натуральных чисел

$$\bar{a}_1, \ldots, \bar{a}_n,\ b_1, \ldots, b_n,\ c_1, \ldots, c_n, \tag{3}$$

полагая

$$\begin{gathered}\bar{a}_i = a_i \cdot 10^{2n} + 10^{n+i-1} + 10^{i-1} \quad \text{для} \ \ i = 1, \ldots, n,\\ b_i = 10^{n+i-1} + 10^i \quad \text{для} \ \ i = 1, \ldots, n-1,\\ b_n = 10^{2n-1} + 1,\\ c_i = 2 \cdot 10^{i-1} \quad \text{для} \ \ i = 1, \ldots, n.\end{gathered}$$

Покажем, что уравнение (2) имеет решение в числах 0, 1 тогда и только тогда, когда натуральные числа (3) образуют «Неинъективный рюкзак». Рассмотрим функцию

$$f(x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_n) = \bar{a}_1 x_1 + \ldots + \bar{a}_n x_n + b_1 y_1 + \ldots + b_n y_n + c_1 z_1 + \ldots + c_n z_n.$$

Для произвольных десятичных цифр $d_1, \ldots, d_k$ через $\overline{d_k \ldots d_1}_{10}$, как обычно, будем обозначать число

$$d_k \cdot 10^{k-1} + \ldots + d_2 \cdot 10 + d_1.$$

Рассмотрим f как отображение множества $\{0,1\}^{3n}$ в множество натуральных чисел. Тогда натуральные числа (3) задают «Неинъективный рюкзак» в том и только в том случае, когда это отображение не является инъективным.

Нетрудно понять, что для произвольных чисел $\alpha_1, \ldots, \alpha_n$, $\beta_1, \ldots, \beta_n$, $\gamma_1, \ldots, \gamma_n$ из множества $\{0,1\}$ справедливо равенство

$$\begin{aligned} f(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n) &= (a_1\alpha_1 + \ldots + a_n\alpha_n)10^{2n} + \overline{(\beta_n + \alpha_n) \ldots (\beta_1 + \alpha_1)}_{10} \cdot 10^n + \\ &+ \overline{(\beta_{n-1} + \alpha_n + 2\gamma_n)(\beta_{n-2} + \alpha_{n-1} + 2\gamma_{n-1}) \ldots (\beta_1 + \alpha_2 + 2\gamma_2)(\beta_n + \alpha_1 + 2\gamma_1)}_{10}.\end{aligned}$$

Поэтому для произвольных чисел $\alpha_1, \ldots, \alpha_n$, $\beta_1, \ldots, \beta_n$, $\gamma_1, \ldots, \gamma_n$, $\alpha'_1, \ldots, \alpha'_n$, $\beta'_1, \ldots, \beta'_n$, $\gamma'_1, \ldots, \gamma'_n$ из множества $\{0,1\}$ равенство

$$f(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n) = f(\alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n)$$

равносильно системе равенств

$$\begin{cases} a_1\alpha_1 + \ldots + a_n\alpha_n = a_1\alpha'_1 + \ldots + a_n\alpha'_n,\\ \beta_n + \alpha_n = \beta'_n + \alpha'_n,\\ \beta_{n-1} + \alpha_{n-1} = \beta'_{n-1} + \alpha'_{n-1},\\ \ldots\\ \beta_2 + \alpha_2 = \beta'_2 + \alpha'_2,\\ \beta_1 + \alpha_1 = \beta'_1 + \alpha'_1,\\ \beta_{n-1} + \alpha_n + 2\gamma_n = \beta'_{n-1} + \alpha'_n + 2\gamma'_n,\\ \beta_{n-2} + \alpha_{n-1} + 2\gamma_{n-1} = \beta'_{n-2} + \alpha'_{n-1} + 2\gamma'_{n-1},\\ \ldots\\ \beta_1 + \alpha_2 + 2\gamma_2 = \beta'_1 + \alpha'_2 + 2\gamma'_2,\\ \beta_n + \alpha_1 + 2\gamma_1 = \beta'_n + \alpha'_1 + 2\gamma'_1.\end{cases}$$

Предположим, что уравнение (2) имеет решение $\alpha_1, \ldots, \alpha_n$ в числах 0, 1. Покажем, что найдутся такие числа $\beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n, \alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n \in \{0, 1\}$, которые нарушают инъективность рюкзака (3), т. е. наборы

$$(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n) \quad \text{и} \quad (\alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n)$$

различны, но выполняется равенство

$$f(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n) = f(\alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n).$$

Уравнение (2) даёт равенство $a_1\alpha_1 + \ldots + a_n\alpha_n = a_1(1 - \alpha_1) + \ldots + a_n(1 - \alpha_n)$, поэтому в качестве чисел $\alpha'_1, \ldots, \alpha'_n$ можно взять $1 - \alpha_1, \ldots, 1 - \alpha_n$. Тогда $\alpha_i \neq \alpha'_i$ при произвольном i, $1 \leqslant i \leqslant n$. Для нахождения чисел $\beta_1, \ldots, \beta_n$, $\gamma_1, \ldots, \gamma_n$, $\beta'_1, \ldots, \beta'_n$, $\gamma'_1, \ldots, \gamma'_n$ получаем две системы

$$\begin{cases} \beta_n + 2\alpha_n = 1 + \beta'_n, \\ \beta_{n-1} + 2\alpha_{n-1} = 1 + \beta'_{n-1}, \\ \quad \ldots \\ \beta_2 + 2\alpha_2 = 1 + \beta'_2, \\ \beta_1 + 2\alpha_1 = 1 + \beta'_1 \end{cases} \quad \text{и} \quad \begin{cases} \beta_{n-1} + 2\alpha_n = 1 + \beta'_{n-1} + 2(\gamma_n - \gamma'_n), \\ \beta_{n-2} + 2\alpha_{n-1} = 1 + \beta'_{n-2} + 2(\gamma_{n-1} - \gamma'_{n-1}), \\ \quad \ldots \\ \beta_1 + 2\alpha_2 = 1 + \beta'_1 + 2(\gamma_2 - \gamma'_2), \\ \beta_n + 2\alpha_1 = 1 + \beta'_n + 2(\gamma_1 - \gamma'_1). \end{cases}$$

Из первой системы при произвольном i, $1 \leqslant i \leqslant n$, получаем $\beta_i - \beta'_i = 1 - 2\alpha_i$. Правая часть этого равенства равна ± 1, что позволяет при любом значении α_i однозначно определить числа β_i и β'_i. При этом $\beta_i - \beta'_i = \pm 1$, поэтому $\beta_i + \beta'_i = 1$.

Для нахождения чисел $\gamma_1, \ldots, \gamma_n$, $\gamma'_1, \ldots, \gamma'_n$ воспользуемся системой

$$\begin{cases} \beta_{n-1} + 2\alpha_n = 1 + \beta'_{n-1} + 2(\gamma_n - \gamma'_n), \\ \beta_{n-2} + 2\alpha_{n-1} = 1 + \beta'_{n-2} + 2(\gamma_{n-1} - \gamma'_{n-1}), \\ \quad \ldots \\ \beta_1 + 2\alpha_2 = 1 + \beta'_1 + 2(\gamma_2 - \gamma'_2), \\ \beta_n + 2\alpha_1 = 1 + \beta'_n + 2(\gamma_1 - \gamma'_1). \end{cases}$$

При $2 \leqslant i \leqslant n$ получаем равенство $2(\gamma'_i - \gamma_i) = 2\alpha_i - (1 + (\beta_{i-1}, -\beta_{i-1}))$. Так как $\beta_i + \beta'_i = 1$, то $\gamma'_i - \gamma_i = \alpha_i - 1 + \beta_{i-1}$. Это уравнение разрешимо при любых α_i и β_{i-1}. Для нахождения γ_1 и γ'_1 воспользуемся уравнением $\gamma'_1 - \gamma_1 = \alpha_1 - 1 + \beta_n$.

Для доказательства обратного предположим, что для двух различных наборов $\langle \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n \rangle$ и $\langle \alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n \rangle$ чисел 0, 1 выполняется равенство

$$f(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n) = f(\alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n).$$

Это даёт равенство $a_1\alpha_1 + \ldots a_n\alpha_n = a_1\alpha'_1 + \ldots a_n\alpha'_n$ и две системы равенств

$$\begin{cases} \beta_n + \alpha_n = \beta'_n + \alpha'_n, \\ \beta_{n-1} + \alpha_{n-1} = \beta'_{n-1} + \alpha'_{n-1}, \\ \ldots \\ \beta_2 + \alpha_2 = \beta'_2 + \alpha'_2, \\ \beta_1 + \alpha_1 = \beta'_1 + \alpha'_1 \end{cases} \quad \text{и} \quad \begin{cases} \beta_{n-1} + \alpha_n + 2\gamma_n = \beta'_{n-1} + \alpha'_n + 2\gamma'_n, \\ \beta_{n-2} + \alpha_{n-1} + 2\gamma_{n-1} = \beta'_{n-2} + \alpha'_{n-1} + 2\gamma'_{n-1}, \\ \ldots \\ \beta_1 + \alpha_2 + 2\gamma_2 = \beta'_1 + \alpha'_2 + 2\gamma'_2, \\ \beta_n + \alpha_1 + 2\gamma_1 = \beta'_n + \alpha'_1 + 2\gamma'_1, \end{cases}$$

которыми воспользуемся для доказательства равенств $\alpha'_1 = 1 - \alpha_1, \ldots, \alpha'_n = 1 - \alpha_n$.

Перепишем эти системы в более наглядном виде:

$$\begin{cases} \beta_n - \beta'_n = \alpha'_n - \alpha_n, \\ \beta_{n-1} - \beta'_{n-1} = \alpha'_{n-1} - \alpha_{n-1}, \\ \ldots \\ \beta_2 - \beta'_2 = \alpha'_2 - \alpha_2, \\ \beta_1 - \beta'_1 = \alpha'_1 - \alpha_1 \end{cases} \quad \text{и} \quad \begin{cases} (\beta_{n-1} - \beta'_{n-1}) + (\alpha_n - \alpha'_n) + 2(\gamma_n - \gamma'_n) = 0, \\ (\beta_{n-2} - \beta'_{n-2}) + (\alpha_{n-1} - \alpha'_{n-1}) + 2(\gamma_{n-1} - \gamma'_{n-1}) = 0, \\ \ldots \\ (\beta_1 - \beta'_1) + (\alpha_2 - \alpha'_2) + 2(\gamma_2 - \gamma'_2) = 0, \\ (\beta_n - \beta'_n) + (\alpha_1 - \alpha'_1) + 2(\gamma_1 - \gamma'_1) = 0. \end{cases}$$

Предположим, что при некотором i выполняется равенство $\alpha_i = \alpha'_i$. Если $i < n$, то, двигаясь по системам вверх, последовательно получаем

1) $\beta_i = \beta'_i$,
2) $(\alpha_{i+1} - \alpha'_{i+1}) + 2(\gamma_{i+1} - \gamma'_{i+1}) = 0$,
3) $(\alpha_{i+1} = \alpha'_{i+1})$ и $(\gamma_{i+1} = \gamma'_{i+1})$,
4) $\beta_{i+1} = \beta'_{i+1}$,
5) $(\alpha_{i+2} - \alpha'_{i+2}) + 2(\gamma_{i+2} - \gamma'_{i+2}) = 0$,
6) $(\alpha_{i+2} = \alpha'_{i+2})$ и $(\gamma_{i+2} = \gamma'_{i+2})$,

…

$3(n-i))$ $(\alpha_n = \alpha'_n)$ и $(\gamma_n = \gamma'_n)$,
$3(n-i)+1)$ $\beta_n = \beta'_n$,
$3(n-i)+2)$ $(\alpha_1 - \alpha'_1) + 2(\gamma_1 - \gamma'_1) = 0$,
$3(n-i)+3)$ $(\alpha_1 = \alpha'_1)$ и $(\gamma_1 = \gamma'_1)$.

Продолжая двигаться по системам вверх, получим, что наборы

$$\langle \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n \rangle \quad \text{и} \quad \langle \alpha'_1, \ldots, \alpha'_n, \beta'_1, \ldots, \beta'_n, \gamma'_1, \ldots, \gamma'_n \rangle$$

совпадают, что противоречит предположению. Значит, при любом i выполняется равенство $\alpha'_i = 1 - \alpha_i$. Поэтому набор $\langle \alpha_1, \ldots, \alpha_n \rangle$ является решением уравнения (2). Это завершает доказательство теоремы. ■

Следствие 1. Задача «m-инъективный рюкзак» является coNP-полной.

Заключение

В работе доказано, что задача «Неинъективный рюкзак» является NP-полной, а задача «Инъективный рюкзак» — coNP-полной. При этом интересно отметить, что с определённой точки зрения «Инъективных рюкзаков» практически столько же, сколько «Неинъективных».

В работах [10, 11] устанавливается, что число «Инъективных рюкзаков» размерности n с фиксированным максимальным элементом M ограничено снизу величиной

$$n!\Big(\frac{M - 2^{n-1} + 1}{2^{n-2}} - 1\Big)\Big(\frac{M - 2^{n-1} + 1}{2^{n-1}}\Big)^{n-2}, \text{ где полагаем } M \geqslant 2^{n-1},$$

а сверху — $n!C_{M-1}^{n-1}$, где полагаем $M \geqslant n$; C_r^s — число сочетаний. Отметим, что оценки эти достаточно грубые, поскольку нижняя оценка получена фактически для «Супервозрастающих рюкзаков», а если рассмотреть верхнюю оценку для числа «Инъективных рюкзаков», максимальный элемент которых принимает значения от n до M, то получим

$$n!\sum_{k=n}^{M} C_{k-1}^{n-1} = n!\Big(\frac{1}{0!}+\frac{n}{1!}+\frac{(n+1)n}{2!}+\frac{(n+2)(n+1)n}{3!}+\ldots+\frac{(M-1)\ldots(n+1)n}{(M-n)!}\Big) =$$
$$= n!\Big(\frac{(n+2)(n+1)}{2!}+\frac{(n+2)(n+1)n}{3!}+\ldots+\frac{(M-1)\ldots(n+1)n}{(M-n)!}\Big) =$$
$$= n!\frac{M(M-1)\ldots(n+1)}{(M-n)!} = \frac{M!}{(M-n)!},$$

и поскольку $\lim\limits_{M\to\infty}\dfrac{M!}{(M-n)!M^n} = 1$, то «практически все» рюкзаки должны быть инъективными.

Действуя по аналогии, можно установить, что число «m-инъективных рюкзаков» размерности n с фиксированным максимальным элементом M ограничено снизу величиной

$$n!\Big(\frac{M-m^{n-1}+1}{(m-1)m^{n-2}}-1\Big)\Big(\frac{M-m^{n-1}+1}{m^{n-1}}\Big)^{n-2}, \text{ где полагаем } M \geqslant m^{n-1},$$

а сверху — $n!C_k^{n-1}$, где полагаем $M \geqslant n(m-1)$; $k = [(M-m+1)/(m-1)]$.

Если посмотреть на все оценки, как на полиномы от M, то окажется, что каждый полином имеет степень $n-1$. При этом общее число «рюкзаков» размерности n с фиксированным максимальным элементом M составляет $M^n-(M-1)^n = \sum\limits_{k=1}^{n} C_n^k(M-1)^{n-k}$ и также является полиномом степени $n-1$ от M. Это означает, что доля «Инъективных рюкзаков» размерности n среди всех рюкзаков зависит только от n, то есть при фиксированном n эта доля является константой, соответственно является константой и доля «Неинъективных рюкзаков».

ЛИТЕРАТУРА

1. *Karp R. M.* Reducibility among combinatorial problems // Complexity of Computer Computations. N. Y.: Plenum Press, 1972. P. 85–103.
2. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
3. *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1984.
4. *Merkle R. and Hellman M.* Hiding information and signature in trapdoor knapsacks // IEEE Trans. Inform. Theory. 1978. V. 24. No. 5. P. 525–530.
5. *Niederreiter H.* Knapsack-type cryptosystems and algebraic coding theory // Problems Control Inform. Theory. 1986. No. 15(2). P. 159–166.
6. *Chor B. and Rivest R.* A knapsack-type public key cryptosystem based on arithmetic in finite fields // CRYPTO'84. LNCS. 1985. V. 196. P. 54–65.
7. *Осипян В. О.* О криптосистемах с заданным рюкзаком // Материалы VI Междунар. науч.-практич. конф. «Информационная безопасность». Таганрог: Изд-во ТРТУ, 2004. С. 269–271.
8. *Осипян В. О.* О системе защиты информации на основе проблемы рюкзака // Известия Томского политехнического университета. 2006. Т. 309. № 2. С. 209–212.
9. *Woeginger G. J. and Yu Z.* On the equal-subset-sum problem // Inform. Proc. Lett. 1992. V. 42. P. 299–302.

10. *Мурин Д. М.* О порядке роста числа инъективных и сверхрастущих рюкзачных векторов // Моделирование и анализ информационных систем. 2012. Т. 19. № 3. С. 124–135.
11. *Мурин Д. М.* О порядке роста числа инъективных и сверхрастущих векторов и некоторых особенностях сильного модульного умножения // Прикладная дискретная математика. Приложение. 2012. № 5. С. 19–21.

REFERENCES

1. *Karp R. M.* Reducibility among combinatorial problems. Complexity of Computer Computations, N. Y., Plenum Press, 1972, pp. 85–103.
2. *Garey R. and Johnson D.* Computers and Intractability. N. Y., USA, W. H. Freeman & Co, 1979.
3. *Papadimitriou C. H. and Steiglitz K.* Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, 1982.
4. *Merkle R. and Hellman M.* Hiding information and signature in trapdoor knapsacks. IEEE Trans. Inform. Theory, 1978, vol. 24, no. 5, pp. 525–530.
5. *Niederreiter H.* Knapsack-type cryptosystems and algebraic coding theory. Problems Control Inform. Theory, 1986, no. 15(2), pp. 159–166.
6. *Chor B. and Rivest R.* A knapsack-type public key cryptosystem based on arithmetic in finite fields. CRYPTO'84, LNCS, 1985, vol. 196, pp. 54–65.
7. *Osipyan V. O.* O kriptosistemakh s zadannym ryukzakom [About cryptosystems with a given knapsack]. Proc. VI Int. Conf. "Information security". Taganrog, TRTU Publ., 2004, pp. 269–271. (in Russian)
8. *Osipyan V. O.* O sisteme zashchity informatsii na osnove problemy ryukzaka [Information protection system based on the knapsack problem]. Bull. TPU, 2006, vol. 309, no 2, pp. 209–212. (in Russian)
9. *Woeginger G. J. and Yu Z.* On the equal-subset-sum problem. Inform. Proc. Lett., 1992, vol. 42, pp. 299–302.
10. *Murin D. M.* O poryadke rosta chisla in"ektivnykh i sverkhrastushchikh ryukzachnykh vektorov [The order in the growth of the injective and super-increasing vectors knapsacks quantity]. Modelirovanie i Analiz Informatsionnykh Sistem, 2012, vol. 19, no 3, pp. 124–135. (in Russian)

11. *Murin D. M.* O poryadke rosta chisla in"ektivnykh i sverkhrastushchikh vektorov i nekotorykh osobennostyakh sil'nogo modul'nogo umnozheniya [About the order of the increasing in the number of the injective and super-increasing vectors and some particularities in the strong modular multiplying]. Prikladnaya diskretnaya matematika. Prilozhenie, 2012, no. 5, pp. 19–21. (in Russian)

УДК 510.52

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ ПРОБЛЕМЫ ДИСКРЕТНОГО ЛОГАРИФМА[1]

А. Н. Рыбалов

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

Генерический подход к алгоритмическим проблемам предложен А. Мясниковым, И. Каповичем, П. Шуппом и В. Шпильрайном в 2003 г. В рамках этого подхода рассматривается поведение алгоритмов на множествах почти всех входов. Изучается генерическая сложность классической проблемы дискретного логарифма в полях $\mathrm{GF}(p)$, где p — простое. Доказывается, что её естественная подпроблема генерически трудноразрешима (то есть трудна для почти всех входов) при условии, что проблема дискретного логарифма трудноразрешима в классическом смысле.

Ключевые слова: *генерическая сложность, проблема дискретного логарифма, вероятностный алгоритм.*

DOI 10.17223/20710410/33/8

ON GENERIC COMPLEXITY OF THE DISCRETE LOGARITHM PROBLEM

A. N. Rybalov

Sobolev Institute of Mathematics SB RAS, Novosibirsk, 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 behaviour of an algorithm on typical (almost all) inputs and ignores the rest of inputs. Many classical undecidable or hard algorithmic problems become feasible in the generic case. But there are generically hard problems. In this paper, we consider generic complexity of the classical discrete logarithm problem. We fit this problem in the frameworks of generic complexity and prove that its natural subproblem is generically hard provided that the discrete logarithm problem is hard in the worst case.

Keywords: *generic complexity, discrete logarithm problem, probabilistic algorithm.*

Введение

В работе [1] развита теория генерической сложности вычислений. 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

[1]Работа поддержана грантом РФФИ № 15-41-04312.

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. В [1, 2] доказано, что таким поведением обладают многие алгоритмические проблемы алгебры, а в [3] построено генерическое множество, на котором разрешима классическая проблема остановки для машин Тьюринга с лентой, бесконечной в одном направлении. Для многих классических NP-полных проблем существуют полиномиальные генерические алгоритмы [4].

С точки зрения современной криптографии интересны такие алгоритмические проблемы, которые, являясь (гипотетически) трудными в классическом смысле, остаются трудными и в генерическом смысле, т. е. для почти всех входов. Это объясняется тем, что при случайной генерации ключей в криптографическом алгоритме происходит генерация входа некоторой трудной алгоритмической проблемы, лежащей в основе алгоритма. Если проблема генерически легкоразрешима, то для почти всех таких входов её можно быстро решить и ключи почти всегда будут нестойкими. Поэтому проблема должна быть генерически трудной. В данной работе изучается генерическая сложность классической проблемы криптографии — проблемы дискретного логарифма в полях $\mathrm{GF}(p)$, где p простое. До сих пор не известно полиномиальных алгоритмов её решения; на предположении об её трудноразрешимости основаны многие криптографические алгоритмы [5]. В работе доказывается, что эта проблема генерически неразрешима за полиномиальное время при условии отсутствия полиномиального вероятностного алгоритма её решения в худшем случае. Более того, существует правдоподобная гипотеза о том, что любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, т. е. построить полиномиальный детерминированный алгоритм, решающий ту же задачу. Хотя это пока ещё не доказано, имеются серьёзные результаты в этом направлении [6]. При доказательстве основного результата работы использованы методы, развитые в [7, 8].

1. Генерические алгоритмы

Пусть I есть множество всех входов некоторой алгоритмической проблемы и I_n — множество всех входов размера n. Для подмножества $S \subseteq I$ определим последовательность

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

Заметим, что $\rho_n(S)$ — это вероятность попасть в S при случайной и равновероятной генерации входов из I_n. *Асимптотической плотностью* S назовём предел (если он существует)

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

Множество S называется *генерическим*, если $\rho(S) = 1$, и *пренебрежимым*, если $\rho(S) = 0$. Очевидно, что S генерическое тогда и только тогда, когда его дополнение $I \setminus S$ пренебрежимо.

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

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

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если для всех $x \in I$ $\mathcal{A}(x) = y \in J \Rightarrow f(x) = y$. 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).

2. Проблема дискретного логарифма

Напомним, что проблема дискретного логарифма состоит в вычислении функции $dl : I \to \mathbb{N}$, где I — это множество троек (a, p, g_p), таких, что p — простое число, g_p — фиксированный первообразный элемент в поле $\mathrm{GF}(p)$ и $a \in \mathrm{GF}(p)$, $a \neq 0$. Сама функция dl определяется следующим образом:

$$dl(a, p, g_p) = x \Leftrightarrow g_p^x = a \in GF(p).$$

Под размером входа понимается число разрядов в двоичной записи числа p. В настоящее время неизвестно полиномиальных алгоритмов (даже вероятностных), решающих проблему дискретного логарифма. Это обстоятельство лежит в основе криптостойкости многочисленных криптографических алгоритмов [5].

Для изучения генерической сложности этой проблемы необходимо провести некоторую стратификацию на множестве входов. Рассмотрим любую бесконечную последовательность простых чисел

$$\pi = \{p_1, p_2, \ldots, p_n, \ldots\},$$

удовлетворяющую условию $2^n \leqslant p_n < 2^{n+1}$ для любого n. Такие последовательности существуют благодаря известному постулату Бертрана, доказанному П. Л. Чебышевым. Будем называть такую последовательность *экспоненциальной*. Теперь определим функцию dl_π как ограничение функции dl на множество троек (a, p, g_p), таких, что $p \in \pi$. Заметим, что для этой функции множество всех входов размера n состоит из троек (a, p, g_p) с фиксированными p, g_p и произвольным $a \in \{1, \ldots, p-1\}$. Очевидно, что проблема вычисления dl_π является подпроблемой вычисления dl. Следующая лемма показывает, что некоторые такие подпроблемы так же трудны, как и оригинальная проблема дискретного логарифма.

Лемма 1. Если не существует полиномиального вероятностного алгоритма для вычисления dl, то найдется такая экспоненциальная последовательность простых чисел π, что и для вычисления dl_π нет полиномиального вероятностного алгоритма.

Доказательство. Пусть $P_1, P_2, \ldots$ — все полиномиальные вероятностные алгоритмы. Из предположения о том, что не существует полиномиального вероятностного алгоритма для вычисления dl, следует, что для любого алгоритма P_n существует бесконечно много полей $\mathrm{GF}(p)$, в которых он не может вычислить dl. Из этого следует, что можно выбрать последовательность $\pi' = \{p_1, p_2, \ldots\}$ так, чтобы алгоритм P_n не вычислял dl в поле $\mathrm{GF}(p_n)$ и для любого n выполнялось бы $p_{n+1} > 2p_n$. Последовательность π' можно расширить до экспоненциальной последовательности π, добавив где нужно новые члены. Заметим теперь, что dl_π и будет той функцией, для вычисления которой не существует полиномиального алгоритма. ■

3. Основной результат

Следующий результат говорит о том, что проблема дискретного логарифма остаётся вычислительно трудной и в генерическом случае при условии её трудноразрешимости в худшем.

Теорема 1. Пусть π — любая экспоненциальная последовательность простых чисел. Если существует полиномиальный генерический алгоритм, вычисляющий функцию dl_π, то существует полиномиальный вероятностный алгоритм, вычисляющий dl_π для всех входов.

Доказательство. Пусть существует полиномиальный генерический алгоритм $\mathcal{A}$, вычисляющий функцию dl_π. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, вычисляющий dl_π на всем множестве входов. Алгоритм $\mathcal{B}$ на входе (a, p, g_p) будет работать следующим образом:

1) сгенерировать случайно и равномерно $y \in \{0, \dots, p-1\}$ и вычислить $a' = ag_p^y$;
2) запустить алгоритм $\mathcal{A}$ на (a', p, g_p);
3) если $\mathcal{A}(a', p, g_p) = z \in \mathbb{N}$, то $a' = g_p^z = ag_p^y = g_p^{x+y}$, откуда $x = z - y \bmod (p-1)$ — дискретный логарифм для исходной задачи (a, p, g_p);
4) если $\mathcal{A}(a', p, g_p) = ?$, то выдать 0.

Заметим, что данный полиномиальный вероятностный алгоритм может выдать неправильный ответ только на шаге 4. Докажем, что вероятность этого меньше 1/2. Действительно, $a' = ag_p^y$ при $y \in \{0, \dots, p-1\}$ пробегает все ненулевые элементы поля $\mathrm{GF}(p)$, поэтому множество $\{(a', p, g_p) : y \in \{0, \dots, p-1\}\}$ совпадает с множеством всех входов размера n. Но алгоритм $\mathcal{A}$ генерический, поэтому доля тех входов (a', p, g_p), на которых он выдаёт неопределённый ответ, стремится к 0 с ростом n и с некоторого момента становится меньше 1/2. ■

Непосредственным следствием доказанной теоремы является

Теорема 2. Если для вычисления функции dl не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность π, такая, что для вычисления функции dl_π не существует генерического полиномиального алгоритма.

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. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Average-case complexity for the word and membership problems in group theory // Adv. Math. 2005. V. 190. P. 343–359.
3. *Hamkins J. D. and Miasnikov A. G.* The halting problem is decidable on a set of asymptotic probability one // Notre Dame J. Formal Logic. 2006. V. 47. No. 4. P. 515–524.
4. *Gilman R., Miasnikov A. G., Myasnikov A. D., and Ushakov A.* Report on generic case complexity // Herald of Omsk University. 2007. Special Issue. P. 103–110.
5. *Мао В.* Современная криптография: теория и практика. М.: Вильямс, 2005. 768 с.
6. *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.
7. *Myasnikov A. and Rybalov A.* Generic complexity of undecidable problems // J. Symbolic Logic. 2008. V. 73. No. 2. P. 656–673.
8. *Rybalov A.* Generic complexity of presburger arithmetic // Theory Comput. Systems. 2010. V. 46. No. 1. P. 2–8.

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. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Average-case complexity for the word and membership problems in group theory. Adv. Math., 2005, vol. 190, pp. 343–359.
3. *Hamkins J. D. and Miasnikov A. G.* The halting problem is decidable on a set of asymptotic probability one. Notre Dame J. Formal Logic, 2006, vol. 47, no. 4, pp. 515–524.
4. *Gilman R., Miasnikov A. G., Myasnikov A. D., and Ushakov A.* Report on generic case complexity. Herald of Omsk University, 2007, Special Issue, pp. 103–110.
5. *Mao W.* Modern Cryptography: Theory and Practice. Prentice Hall PTR, 2003.
6. *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.
7. *Myasnikov A. and Rybalov A.* Generic complexity of undecidable problems. J. Symbolic Logic, 2008, vol. 73, no. 2, pp. 656–673.
8. *Rybalov A.* Generic complexity of presburger arithmetic. Theory Comput. Systems, 2010, vol. 46, no. 1, pp. 2–8.

УДК 591.68

РЕШЕНИЕ ЗАДАЧ НА ГРАФАХ С ПОМОЩЬЮ STAR-МАШИНЫ, РЕАЛИЗУЕМОЙ НА ГРАФИЧЕСКИХ УСКОРИТЕЛЯХ

Т. В. Снытникова, А. Ш. Непомнящая

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

Для решения многих задач на графах построены эффективные алгоритмы на ассоциативных параллельных процессорах. Но на данный момент нет широко используемых ассоциативных архитектур. Однако с развитием графических ускорителей появилась возможность реализовывать ассоциативные параллельные модели без существенной потери эффективности вычислений, что позволяет применять ассоциативные алгоритмы на практике. Представляется реализация абстрактной модели ассоциативной параллельной обработки данных (STAR-машина) на графических ускорителях с помощью технологии CUDA. Измеряется производительность реализации и показывается её эффективность для решения задач на графах на примере алгоритма Уоршалла нахождения транзитивного замыкания ориентированного графа. На графе с 5000 вершин последовательный алгоритм Уоршалла выполнялся за 884,622 с, ассоциативная параллельная версия — за 64,454 с (ускорение в 13 раз), а ассоциативная параллельная версия, адаптированная под GPU, — за 0,372 с (ускорение в 2 378 раз).

Ключевые слова: *ассоциативный параллельный процессор, вертикальная обработка данных, SIMD, GPU, ориентированный граф, транзитивное замыкание.*

DOI 10.17223/20710410/33/9

SOLUTION OF GRAPH PROBLEMS BY MEANS OF THE STAR-MACHINE BEING IMPLEMENTED ON GPUS

T. V. Snytnikova, A. Sh. Nepomniaschaya

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

E-mail: snytav@yahoo.com, anep@ssd.sscc.ru

Efficient algorithms have been developed for the solution of many graph problems. The solution is performed with associative parallel processors. At present there are no widely used associative architectures. Nevertheless, the recent development of the GPUs has made it possible to implement the associative parallel models with no significant efficiency loss. It enables to use the associative algorithms in practice. The implementation of the abstract model of the associative parallel data processor is given (the STAR-machine) with the GPUs by means of the CUDA technology. The performance is being analysed as well as the efficiency of the solution of the graph problems. The test case is the Warshall algorithm for finding the transitive closure of a directed graph. A graph with 5000 nodes was processed by the sequential Warshall algorithm for 884,622 s, by the associative parallel version for 64,454 s (speed-up is 13 times) and by the GPU-adopted associative parallel version for 0,372 s (speed-up is 2 378 times).

Keywords: *associative parallel processor, vertical data processing, SIMD, GPU, directed graph, adjacency matrix, transitive closure.*

Введение

Ассоциативные вычисления используют совершенно другой метод хранения, извлечения данных и их обработки по сравнению с общепринятой последовательной технологией. Ассоциативные (контекстно-адресуемые) параллельные процессоры (АПП) типа SIMD принадлежат к классу мелкозернистых параллельных систем с простейшими процессорными элементами и последовательно-поразрядной (вертикальной) обработкой информации. Такие системы называют ещё системами вертикальной обработки. В них файл входных данных физически загружается в матричную (двумерную) память, в которой каждая запись файла занимает отдельную строку и обрабатывается отдельным процессорным элементом. Основным преимуществом таких архитектур является реализация интеллектуальной памяти. Такие системы ориентированы, в первую очередь, на решение задач нечисловой обработки. Сюда относятся теория графов, реляционные базы данных, базы знаний, экспертные системы, обработка сейсмических данных и обработка изображений. Основными достоинствами АПП являются параллелизм по данным на базовом уровне, массовый параллельный поиск по содержимому памяти, двумерные таблицы в качестве основной структуры данных и обработка неупорядоченных данных [1]. Базовые операции поиска ($=$, $<$, $\leqslant$, $>$, $\geqslant$, min, max) и арифметические операции выполняются за время, пропорциональное числу битовых столбцов в таблице, а не числу её строк. К этому классу параллельных архитектур относятся хорошо известная система STARAN [2, 3] и современные компьютерные архитектуры Rutgers CAM, ASPRO, IXM2 [4].

На основе процессора типа STARAN разработаны некоторые модели ассоциативной параллельной обработки данных. В работе [5] сравниваются три модели ассоциативной обработки данных: STAR-машина [6], модель Поттера ASC [1, 7] и ортогональная машина. Отметим, что при разработке STAR-машины также учитывались основные свойства отечественного АПП ЕС-1020.

Заметим, что, с одной стороны, АПП позволяют строить эффективные алгоритмы для различных приложений. С другой стороны, пока нет широко используемых ассоциативных архитектур, позволяющих реализовывать эти алгоритмы. Однако в последнее время широко распространены относительно недорогие графические ускорители, которые используются для различных приложений. Они относятся к типу SIMD, поэтому хорошо подходят для реализации STAR-машины.

Наша цель — построить эффективную реализацию STAR-машины на графических ускорителях, используя технологию CUDA. Это позволит использовать на практике ассоциативные параллельные алгоритмы [8 – 22]. Для этого необходимо построить эффективную реализацию базовых операций языка Star на графическом ускорителе. Отметим, что библиотека стандартных процедур языка Star будет реализована как отдельный модуль. В работе [23] такой подход обсуждается для реализации ассоциативной параллельной модели вычислений MASC, которая является расширением модели ASC.

В п. 1 описана STAR-машина, приведены типы данных и базовые операции языка Star, перечислены базовые ассоциативные алгоритмы из библиотеки стандартных процедур и приведены группы ассоциативных алгоритмов (Star-алгоритмов) для решения задач на графах. Приводится ассоциативная параллельная версия алгоритма Уоршалла нахождения транзитивного замыкания ориентированного графа. В п. 2 описана реализация типов данных и базовых операций STAR-машины на языке CUDA C; показана адаптация Star-алгоритма, учитывающая архитектурные отличия STAR-машины и графического ускорителя. В п. 3 дан анализ производительности предложенной реализации STAR-машины на графических ускорителях. Сначала на примере выполнения Star-алгоритма Уоршалла оценивается время работы реализации базовых операций на графическом ускорителе, далее — размер данных, необходимых для хранения графа в форматах STAR-машины. Потом сравнивается время выполнения последовательного алгоритма Уоршалла со временем выполнения Star-алгоритмов, представленных в работе.

1. Модель ассоциативной параллельной обработки данных

STAR-машина — абстрактная модель типа SIMD (Single Instruction Multiple Data) с вертикальной обработкой данных — была представлена в работах [6, 8]. STAR-машина состоит из следующих частей (рис. 1):

— последовательного устройства управления (ПУУ), в котором записаны программа и скалярные константы;

— матричной памяти;

— устройства ассоциативной обработки (УАО), состоящего из p одноразрядных процессорных элементов (ПЭ).

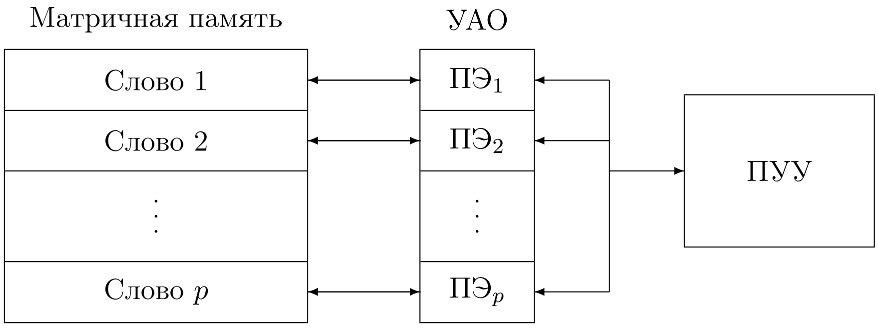


Рис. 1. Модель STAR-машины

Входные данные, записанные в двоичном коде, помещаются в матричную память в виде двумерных таблиц, причём каждая единица данных хранится в отдельной строке и обрабатывается отдельным ПЭ. Строки каждой таблицы нумеруются сверху вниз, а столбцы — слева направо (рис. 2). В матричную память можно загружать несколько таблиц.

Устройство ассоциативной обработки (рис. 3) представляется в виде совокупности h вертикальных одноразрядных регистров длины p. Вертикальный регистр можно представлять как массив, состоящий из одного столбца. Обработка информации происходит следующим образом: из определённой таблицы в определённом порядке извлекаются её одноразрядные столбцы и помещаются в вертикальные регистры устройства ассоциативной обработки, в котором выполняются побитовые операции. Результат выполнения любой операции записывается либо в некоторый регистр, либо в определённый столбец обрабатываемой таблицы, либо в матричную память.

Поясним содержательно работу STAR-машины. Каждый ПЭ может быть либо активным, либо пассивным в зависимости от своего содержимого. Перед выполнением любой программы задаются активные ПЭ, т. е. указываются позиции тех строк заданной таблицы, которые надо анализировать. Каждая команда программы одновременно

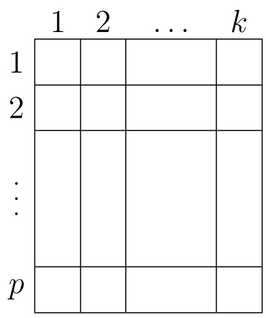


Рис. 2. Матричная память

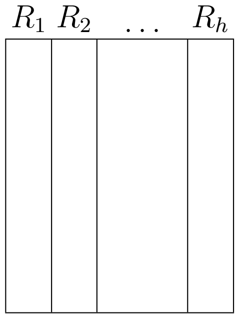


Рис. 3. Устройство ассоциативной обработки

подаётся на все ПЭ, но только активные ПЭ её выполняют. В процессе выполнения некоторых команд некоторые активные ПЭ могут стать пассивными. Результат выполнения программы определяется с помощью активных ПЭ. В качестве примера приведём идею ассоциативного параллельного алгоритма для нахождения строк заданной таблицы T, которые совпадают с заданным образцом v (рис. 4).

v				
	1	0	1	1

T				X	Z
1	0	1	0	1	0
0	0	1	1	0	0
1	0	1	1	1	1
1	0	1	1	0	0
1	0	1	1	1	1
1	0	1	0	1	0

Рис. 4. Нахождение строк таблицы T, совпадающих с образцом v

Сначала с помощью управляющего битового столбца X указываются позиции анализируемых строк таблицы T или активных ПЭ. После считывания k-го бита образца v активными являются ПЭ, которые хранят строки, начинающиеся подцепочкой длины k образца v. После завершения программы активные ПЭ указывают позиции строк таблицы T, которые совпадают с образцом v. На рис. 4 после считывания последнего бита образца v первый и шестой процессорные элементы станут пассивными. Результат указывается с помощью битового столбца Z. В работе [8] приведена процедура MATCH, которая реализует ассоциативный алгоритм поиска строк, совпадающих с образцом.

Для STAR-машины в [6, 8] разработан язык высокого уровня Star, похожий на язык Pascal. Чтобы моделировать обработку информации в матричной памяти, в языке Star имеются три типа данных: `slice` (далее слайс), `word` (слово) и `table` (таблица). С помощью переменной типа `slice` моделируется доступ к таблице по столбцам, а с помощью переменной типа `word` — доступ по строкам. С каждой переменной типа `table` ассоциируется бинарная таблица из n строк и k столбцов, где $n \leqslant p$.

Приведём основные операции для переменной типа `slice`. Пусть X и Y — слайсы, i — целочисленная переменная. Тогда операции имеют следующий вид:

— $SET(Y)$ записывает в слайс Y все единицы;
— $CLR(Y)$ записывает в слайс Y все нули;
— $Y(i)$ выделяет i-ю компоненту в слайсе Y $(1 \leqslant i \leqslant n)$;

— $FND(Y)$ выдаёт порядковый номер i позиции первой (старшей) единицы в слайсе Y, где $i \geqslant 0$;
— $STEP(Y)$ выдаёт такой же результат, как и операция $FND(Y)$, и затем обнуляет старшую единицу;
— $NUMB(Y)$ выдаёт число компонент 1 в слайсе Y;
— $CONVERT(Y)$ выдаёт слово, i-й бит которого совпадает с $Y(i)$;
— $FRST(Y)$ сохраняет первую (старшую) единицу в слайсе Y и обнуляет остальные компоненты.

Общепринятым способом вводятся предикат $SOME(Y)$, а также следующие побитовые логические операции: $X \,\mathrm{and}\, Y$, $X \,\mathrm{or}\, Y$, $\mathrm{not}\, X$, $X \,\mathrm{xor}\, Y$.

Замечание 1. Операция $STEP(Y)$ используется для построения неравномерных циклов.

Для переменных типа `word` выполняются те же операции, что и для переменных типа `slice`. Кроме этого, для слов определены следующие операции:

— $TRIM(i, j, w)$ возвращает подстроку слова w с i-го по j-й биты, где $1 \leqslant i < j \leqslant |w|$;
— $REP(i, j, v, w)$ замещает подстроку $w(i)w(i+1)\dots w(j)$ слова w словом v, где $|v| = j - i + 1$ и $1 \leqslant i < j < |w|$.

Следуя Дж. Поттеру [1], для двух переменных типа `word` определены следующие предикаты: $EQ(v, w)$, $NOTEQ(v, w)$, $LESS(v, w)$, $LESSEQ(v, w)$, $GREAT(v, w)$ и $GREATEQ(v, w)$, а также арифметические функции $ADD(v, w)$ и $SUBT(v, w)$.

Для переменных типа `table` определены следующие две операции:

— $COL(i, T)$ обеспечивает доступ к i-му столбцу таблицы T для выполнения чтения и записи;
— $ROW(i, T)$ обеспечивает доступ к i-й строке таблицы T для выполнения чтения и записи.

1.1. Базовые Star-алгоритмы

Для STAR-машины разработаны различные базовые ассоциативные алгоритмы как для нечисловой, так и для числовой обработки данных. В работе [8] они приведены в виде процедур на языке Star. Процедуры используют управляющий слайс, в котором единицами отмечены *позиции* обрабатываемых строк. К алгоритмам нечисловой обработки, в частности, относятся:

— поиск строк таблицы, совпадающих с образцом;
— сравнение строк с образцом (больше, меньше, равно);
— поиск минимальной строки;
— поиск максимальной строки;
— сравнение таблиц построчно.

К алгоритмам числовой обработки, в частности, относятся:

— прибавление слова w к строкам таблицы;
— построчное сложение таблиц;
— вычитание слова w из строк таблицы;
— построчное вычитание одной таблицы из другой;
— нахождение позиций совпадающих строк.

Эти алгоритмы выполняются на STAR-машине за время, пропорциональное количеству битовых столбцов таблицы T.

1.2. Star-алгоритмы на графах

Для STAR-машины построены как новые ассоциативные параллельные алгоритмы на графах, так и ассоциативные версии ряда известных последовательных алгоритмов. В работах [9, 10] приводятся специальные конструкции и методы ассоциативной обработки данных, которые были разработаны для построения ассоциативных параллельных алгоритмов на графах.

Для *ориентированных* графов разработаны ассоциативные версии алгоритма Уоршалла для нахождения транзитивного замыкания [11], алгоритма Флойда для нахождения кратчайших расстояний между любыми парами вершин [11], алгоритмов Дейкстры [12] и Беллмана — Форда [13] для нахождения кратчайших расстояний и кратчайших путей от заданной вершины до остальных вершин графа, алгоритма Эдмондса [14] для нахождения оптимального ветвления. Для *неориентированных* графов построены ассоциативные версии алгоритмов Краскала и Прима — Дейкстры для нахождения минимального остовного дерева (MST) [15] и алгоритма Габова [16] для нахождения наименьшего остовного дерева с ограничением на степень одной из вершин.

Перечислим группу *динамических алгоритмов* на графах, реализованных на STAR-машине. В [17] построены два ассоциативных параллельных алгоритма для динамической обработки MST после удаления и после добавления одного ребра к неориентированному графу. В [18, 19] построены ассоциативные параллельные алгоритмы для преобразования MST после удаления и после добавления к графу одной вершины вместе с инцидентными рёбрами. В [20] приведены ассоциативные версии алгоритмов Итальяно для динамической обработки транзитивного замыкания ориентированного графа после добавления и после удаления одной дуги. В [21, 22] построены ассоциативные версии алгоритмов Рамалингама для динамической обработки подграфа кратчайших путей с одним стоком после удаления из графа одной дуги и после добавления к графу новой дуги.

1.3. Star-версии алгоритма Уоршалла

Для оценки времени работы реализации базовых операций STAR-машины будем использовать ассоциативную версию алгоритма Уоршалла. При его реализации используются только базовые операции языка Star без привлечения базовых алгоритмов. Описание алгоритма, доказательство его корректности и оценка сложности приведены в [11].

Процедура получает на вход матрицу смежности ориентированного невзвешенного графа с n вершинами и возвращает матрицу путей для транзитивного замыкания графа (листинг 1).

```
proc WARSHALL(n:integer; var P: table);
/*Здесь n - число вершин графа.*/
var X:slice(P); v,w: word; i,k: integer;
begin for k:=1 to n do
   begin
      X:=COL(k,P);
      w:=ROW(k,P);
      while SOME(X) do
      begin
        i:=STEP(X);
        v:=ROW(i,P);
        v=v or w;
```

```
13          ROW(i,P):=v;
14        end;
15     end;
16  end;
```

Листинг 1. Star-версия алгоритма Уоршалла

Модель STAR-машины предполагает организацию доступа как к строкам, так и к столбцам за единицу времени. Но при реализации STAR-машины на графических ускорителях доступ к столбцу осуществляется быстрее, чем к строке. Поэтому модифицируем процедуру WARSHALL так, чтобы минимизировать обращение к строкам (листинг 2).

```
1   proc WARSHALL-C(n:integer; var P: table);
2   var x:word; V, W: Slice; i,k: integer;
3   begin for k:=1 to n do
4      Begin
5         x:=ROW(k,P);
6         W:=COL(k,P);
7         i=STEP(x);
8         while i>0 do
9         Begin
10           V:=COL(i,P);
11           V=V or W;
12           COL(i,P):=V;
13           i:=STEP(x);
14        end;
15     end;
16  end;
```

Листинг 2. Модифицированная Star-версия алгоритма Уоршалла

Рис. 5 иллюстрирует выполнение одной итерации ($k = 2$) процедур WARSHALL и WARSHALL-C. В процедуре WARSHALL в слайс X записан второй столбец, а в слово w — вторая строка матрицы P^1. В цикле (8–14) строки (выделены курсивом), отмеченные единицами в слайсе X, объединяются со словом w. Значения элементов строк, выделенных только курсивом, не изменяются, потому что соответствующие им элементы слова w равны нулю. Поэтому изменить значения могли только те элементы строк, которые находятся в столбцах, отмеченных единицами в слове w. Такие элементы отмечены жирным шрифтом. Аналогично в процедуре WARSHALL-C в слайс W записан второй столбец, а в слово x — вторая строка матрицы P^1. В цикле (8–14) столбцы (выделены курсивом), отмеченные единицами в строке x, объединяются со слайсом W. На этой итерации могли измениться значения только тех элементов столбцов, которые находятся в строках, отмеченных единицами в слайсе W. Эти элементы отмечены жирным шрифтом и их позиции совпадают с позициями элементов, которые могли измениться в процедуре WARSHALL. В утверждении 1 приведено формальное доказательство.

Утверждение 1. Процедуры WARSHALL и WARSHALL-C различаются порядком обхода, но вычисляют одну и ту же логическую функцию.

Доказательство. Для доказательства утверждения 1 восстановим вычисляемую логическую функцию для каждой процедуры.

X	$w=$	**1**	0	**1**	0	0	0
0		0	0	1	0	0	0
0		1	0	1	0	0	0
1		***1***	*1*	***1***	*0*	*0*	*1*
1		***1***	*1*	***1***	*0*	*0*	*0*
0		0	0	1	0	0	0
0		0	0	0	1	1	0

WARSHALL

W	$x=$	***1***	0	***1***	0	0	0
0		*0*	0	*1*	0	0	0
0		*1*	0	*1*	0	0	0
1		***1***	1	***1***	0	0	1
1		***1***	1	***1***	0	0	0
0		*0*	0	*1*	0	0	0
0		*0*	0	*0*	1	1	0

WARSHALL-C

Рис. 5. Результат выполнения итерации ($k = 2$) для Star-версий алгоритма Уоршалла. Обрабатываемые элементы матриц выделены курсивом, элементы, изменяющие свои значения, — жирным шрифтом

Заметим, что $COl(k, P)$ соответствует $\forall j\, P(j, k)$, а $ROW(k, P) - \forall j\, P(k, j)$. Тогда в этих терминах для каждого фиксированного k тело цикла (5–15) процедуры WARSHALL записывается в следующем виде:

$$\forall j\, \forall i\, P^k(i,j) = \begin{cases} P^{k-1}(i,j), & \text{если } P^{k-1}(i,k) = 0, \\ P^{k-1}(i,j) \text{ or } P^{k-1}(k,j), & \text{если } P^{k-1}(i,k) = 1, \end{cases}$$

или

$$\forall j\, \forall i\, P^k(i,j) = P^{k-1}(i,j) \text{ or } \left(P^{k-1}(i,k) \text{ and } P^{k-1}(k,j)\right). \tag{1}$$

Для тела цикла (4–15) процедуры WARSHALL-C имеем

$$\forall j\, \forall i\, P^k(j,i) = \begin{cases} P^{k-1}(j,i), & \text{если } P^{k-1}(k,i) = 0, \\ P^{k-1}(j,i) \text{ or } P^{k-1}(j,k), & \text{если } P^{k-1}(k,i) = 1, \end{cases}$$

или

$$\forall j\, \forall i\, P^k(j,i) = P^{k-1}(j,i) \text{ or } \left(P^{k-1}(j,k) \text{ and } P^{k-1}(k,i)\right). \tag{2}$$

Формулы (1) и (2) совпадают на каждой итерации $k = 1, \ldots, n$. ■

Замечание 2. Теоретическая оценка обеих ассоциативных версий совпадает (не более $O(n^2)$, фактически $O(j)$, где j — количество ненулевых элементов в результирующей матрице, $n \leqslant j \leqslant n^2$). В процедуре WARSHALL операция COL вызывается n раз, а операция ROW — $(n+j)$ раз на чтение и j раз на запись. В процедуре WARSHALL-C операция ROW вызывается n раз на чтение, а операция COL — $(n + j)$ раз на чтение и j раз на запись.

2. Реализация базовых операций языка Star на графических ускорителях

2.1. Представление типов данных

Как отмечено выше, для доступа к матричной памяти используются типы данных `word`, `slice` и `table`.

Так как CUDA C является расширением языка C++, для реализации типов данных используются одноимённые классы. Заметим, что класс `Slice` используется и для типа `word`. Приведём описание этих классов.

```
class Slice{
// основная память CPU
 /* length - длина слайса в битах;
```

```
    N - количество 64-х разрядных элементов для хранения
   слайса */
 unsigned int length,N;
// глобальная память GPU
/* *d_v хранит адрес первого элемента массива длины $N$ */
 Unsigned long long int *d_v;
...}
```

Здесь переменная N равна числу 64-разрядных переменных, необходимых для хранения битового столбца длины `length`. Битовый столбец хранится в массиве `d_v` из N переменных типа `unsigned long long int`, расположенном в глобальной памяти GPU.

Класс `Table` содержит массив из объектов класса `Slice` и несколько указателей.

```
class Table{
/*length - число строк, size - число столбцов */
    unigned int length,size;
    Slice table[size];
    LongPointer *slice_device_pointer_table;
// глобальная память GPU
    LongPointer *d_table, *d_slice_device_pointer_table;
...}
```

Указатель на GPU `*d_table` хранит адрес первого элемента массива `length`·`N` элементов типа `unsigned long long int`, расположенного в глобальной памяти GPU. Указатели `*slice_device_pointer_table` на CPU и `*d_slice_device_pointer_table` на GPU хранят адреса первых элементов массивов из `size` указателей на столбцы `d_table`. С помощью такой системы указателей осуществляется доступ как целиком к таблице, так и к каждому столбцу в отдельности.

2.2. Базовые операции для типа `slice`

Базовые операции над переменными типа `slice` реализованы как одноименные методы класса `Slice`. В свою очередь, каждый метод запускает функцию на GPU, исполняющую операцию. Таким образом, все вычисления производятся на графическом ускорителе параллельно.

Ниже приводятся реализации базовых операций. Отметим, что при операции присваивания указателей на объект класса возможна неоднозначность. Чтобы её избежать, побитовые логические операции реализованы как совмещенные с присваиванием, аналогично подобным операциям языка C++, т. е. $Z := X \,\text{and}\, Y$ записывается как $Z = X$; $Z.AND(Y)$. Операция and реализована следующим образом:

```
void Slice::AND(Slice *Y)
{
    and_long_values<<<N,1>>>(d_v,Y->d_v);
}
__global__ void and_long_values( *d_v, *d_v1)
{
    d_v[blockIdx.x] &= d_v1[blockIdx.x];
}
```

Операции $SET(X)$, $CLR(X)$ и остальные побитовые логические операции реализованы аналогично. Заметим, что каждый из N блоков вычисляет свой элемент массива независимо от других. Поэтому время выполнения этих операций не зависит от длины битового столбца `slice`.

Операция $FND(X)$ выполняется в два этапа. На первом этапе с помощью глобальной процедуры `find` массив `d_v[N]` за несколько шагов сворачивается до одной переменной типа `unsigned long long int`. На каждом шаге массив сворачивается по следующему правилу:

— если `dv[level,i]`$\neq 0$, то в i-й бит элемента массива `dv[level+1]` записывается 1;
— если `dv[level,i]`$= 0$ или $i > N$, то в i-й бит элемента массива `dv[level+1]` записывается 0.

Таким образом, за шаг свёртки длина массива `dv[level+1]` меньше длины массива `dv[level]` в 64 раза и `dv[level+1]` — битовая маска текущего массива `dv[level]`.

На втором этапе глобальная процедура `first_backward` вычисляет результат, разворачивая свёртку (листинг 3).

```
__global__ void first_backward(LongPointer *d_v, int *
    d_first_non_zero, int level)
{
    int f[LEVELS];
    unsigned long long int *dvl,u;
    char lprt[100];
    f[level+1] = 1;
    while(level >= 0)
    {
        dvl = d_v[level];
        int index = f[level+1]-1;
        u=dvl[index];
        f[level]=__ffsll(u)+index*SIZE_OF_LONG_INT;
        level--;
    }
    *d_first_non_zero = f[0];
}
```

Листинг 3. Процедура `first_backward`

Время выполнения операции $FND(X)$ оценивается как $\mathrm{O}(\log_{64}(N))$, где N — длина массива данных слайса.

Операция $STEP(X)$ и предикат $SOME(X)$ реализованы следующим образом. Пусть $i = FND(X)$ и $i > 0$. Тогда в операции $STEP(X)$ i-й бит обнуляется, а в качестве результата возвращается число i. Предикат $SOME(X)$ возвращает `true`. Операция $X = FRST(Y)$ реализуется следующим образом: $CLR(X)$; $i := FND(Y)$; $X(i) := 1$.

Операция $TRIM(i, j, w)$:

```
Slice::TRIM(int i, int j, Slice *w)
{
   trim_long_values<<<N,1>>>(d_v,i,j,w->d_v);
}
__global__ void trim_long_values(*d_v, int i, int j, *d_v1)
{   unsigned long long int head, tail;
    int k, n, index;
    index=blockIdx.x;
```

```
// сдвиг по битам
    k = i % SIZE_OF_LONG_INT -1;
// сдвиг по элементам
    n=(k==0)?(i/SIZE_OF_LONG_INT-1):(i/SIZE_OF_LONG_INT);
    head = d_v1[n+index]>>k;
    tail = d_v1[n+index+1]<<(SIZE_OF_LONG_INT-k);
    head =head|tail;
    d_v[index] = head;
```

Время выполнения операции $TRIM$ не зависит от длины битового столбца `slice`.

Отметим, что операция $CONVERT$ не нуждается в реализации, поскольку класс `Slice` используется для представления как слайсов, так и слов.

Замечание 3. Базовые операции языка Star над слайсами выполняются на графическом ускорителе или за константное время, или за время $\mathrm{O}(\log_{64}(N))$.

2.3. Базовые операции для типа `table`

Операции над типом `table` обеспечивают доступ к столбцам и строкам таблицы для выполнения чтения и записи.

Операция $COL(i, T)$ выполняется вызовом метода `T.col(i)`:

```
Slice * Table::col(int i) {return &(table[i-1]);}
```

Он обеспечивает доступ к столбцу как на чтение, так и на запись.

Для реализации операции $ROW(i, T)$ используются два метода класса `Table`. Метод `SetRow(Slice *s, int i)` записывает слово s в i-ю строку таблицы T. Метод `GetRow(Slice *s, int i)` записывает i-ю строку таблицы в слово s. Процедуры вычисляются на `s.N` блоках по 64 нити, где `s.N` — число битовых элементов в слове s. Поэтому каждый блок производит вычисления над своим элементом слова, а нити — над своим столбцом.

Напомним, что переменная T типа `table` представляется в глобальной памяти графического ускорителя массивом слайсов. Слайсы, в свою очередь, представляются массивами 64-разрядных элементов. Поэтому для доступа к j-му биту i-й строки необходимы следующие индексы:

— `n_col`= $\lceil i/64 \rceil$ — номер элемента массива в столбце;
— `b_col`= $i \bmod 64$ — номер бита в этом элементе;
— `n_row = blockIdx.x` — номер элемента в слове (равен номеру блока);
— `b_row = threadIdx.x` — номер бита в элементе слова (равен номеру нити в блоке);
— `k`= $64\cdot$`blockIdx.x + threadIdx.x` — номер столбца.

В качестве вспомогательных переменных используется массив `tmp[64]` в разделяемой памяти графического ускорителя (у каждого блока свой экземпляр массива, к которому имеют доступ все нити блока) и `col` — указатель на k-й столбец. Запись слова s в таблицу происходит следующим образом:

```
if (s(k)==1)
then
{
   tmp[b_row]=1<<(b_col-1);
   col[n_col]|=tmp[b_row];
}
else
{
```

```
    tmp[b_row]=~(1<<(b_col-1));
    col[n_col]&=tmp[b_row]
}
```

При чтение строки таблицы элементы массива `tmp` для каждого блока вычисляются следующим образом:

```
    bit=(col[n_col]>>(b_col-1))&1;
    tmp[b_row]=bit<<(b_col-1);
    s[n_row]=tmp[0]|...|tmp[63];
```

Для каждого блока вычисляются 64 элемента массива `tmp`. В элементе `tmp[bit_row-1]` бит с номером `b_row` совпадает с i-м битом k-го столбца, остальные биты равны 0. После этого все элементы массива `tmp` объединяются с помощью побитового ИЛИ в один элемент для каждого блока.

Замечание 4. Операции $COL(i,T)$ и $ROW(i,T)$ выполняются за константное время. Тем не менее $COL(i,T)$ выполняется быстрее $ROW(i,T)$.

2.4. Адаптация Star-алгорима Уоршалла к выполнению на графическом ускорителе

Модель STAR-машины предполагает параллельную обработку одного столбца или одной строки, но архитектура графических ускорителей даёт возможность одновременно обрабатывать несколько столбцов или строк. Поэтому некоторые Star-алгоритмы могут быть адаптированы к выполнению на графических ускорителях.

Так, в процедуре WARSHALL-C цикл (8–14) может выполняться параллельно по всем столбцам. Приведём листинг кода, используя в основном язык Star (листинг 4). Для записи процедуры потребуются из CUDA C++ переменная `blockIdx.x`, возвращающая номер исполняемого блока, и директивы `__global__` и `<<<n>>>`, указывающие, что процедура выполняется на n блоках.

```
__global__ Warshall_kernel(P,k)
Begin
// номер блока индексируется с 0, а номер столбца с 1
    i:=blockIdx.X+1;
    W:=COL(k,P);
    V:=COL(i,P);
    if(V(k)=1) then V=V or W;
    COL(i,P):=V;
end;
proc WARSHALL-adapt(n:integer; var P: table);
var x:word; V, W: Slice; i,k: integer;
Begin
  for k:=1 to n do
    Warshall_kernel<<<n>>>(P,k);
end;
}
```

Листинг 4. Адаптированный Star-алгоритм Уоршалла

Такой подход позволяет снизить временну́ю сложность алгоритма до $O(n)$ и значительно уменьшить время его выполнения. Отметим, что не все блоки физически выполняются одновременно, поэтому экспериментальное время счёта отличается от линейного. Но производители графических ускорителей постоянно наращивают количество ядер в них.

3. Анализ производительности

Приведём анализ времени работы базовых операций в зависимости от размера данных, оценим объём памяти, необходимый для представления графов в зависимости от числа вершин и ребер, и сравним время выполнения модифицированного Star-алгоритма Уоршалла и адаптированного Star-алгоритма со временем выполнения других реализаций этого алгоритма на графическом ускорителе. Все расчёты производились на графическом ускорителе NVidia Kepler k-40.

3.1. Время работы базовых операций

Для оценки времени работы базовых операций рассмотрим выполнение Star-алгоритма Уоршалла (см. листинг 1). Эта процедура использует следующие базовые операции языка Star: or, $SOME$, COL (на чтение), ROW (на чтение и на запись) и $STEP$. При этом вызываются следующие глобальные процедуры:

— `or_long_value` для выполнения операции or;
— `find` и `first_backward` для выполнения предиката $SOME$ и операции $STEP$ (вызывают операцию FND);
— `get_row` для выполнения операции ROW на чтение;
— `set_row` для выполнения операции ROW на запись.

Отметим, что операция COL передаёт указатель на столбец, поэтому ее время работы не измеряется.

Для определения времени работы операций Star-алгоритм Уоршалла выполнялся на графах с разным количеством вершин: 100, 1 000, 3 000 и 5 000 вершин. Результаты профилирования показаны в табл. 1. В первом столбце приведено название базовой операции, во втором — имя процедуры, выполняющейся на графическом ускорителе (реализация операции FND вызывает две глобальных процедуры). Отметим, что время выполнения процедуры включает в себя время запуска ядра, поэтому может существенно отличаться для значений порядка нескольких микросекунд. В табл. 1 приведено среднее время запуска; для 100 вершин в скобках показаны минимальное и максимальное время выполнения процедур.

Таблица 1

Профилирование базовых операций

Операция	Процедура	Время выполнения, мкс			
		100	1 000	3 000	5 000
ROW (чтение строки)	`get_row`	18,6 (18,5 – 19,7)	19,5	19,6	19,8
ROW (запись строки)	`set_row`	4,1 (2,9 – 5,6)	3,9	4,3	4,1
or (побитовое ИЛИ)	`or_long_value`	1,9 (1,9 – 3,0)	2,2	2,0	2,1
FND (номер старшей единицы в слайсе)	`find`	2,5 (2,3 – 2,8)	2,8	3,5	3,8
	`first_backward`	3,2 (3,1 – 4,0)	3,2	3,3	3,4

Замечание 5. Время выполнения базовых операций практически не зависит от размера данных, что соответствует теоретическим оценкам в п. 2: $O(\log_{64}(n))$ для реализации операций FND, $SOME$, $STEP$ и $FRST$ и константное время для реализации остальных операций.

3.2. Размер данных

Исследуем вопрос о необходимом размере памяти для хранения графа, который задаётся в разных форматах.

Для ассоциативных параллельных алгоритмов графы задаются в одном из двух форматов: матрицей весов (матрицей смежности, если граф невзвешенный) или списком рёбер и весов.

При использовании матрицы смежности для графа с n вершинами необходимо $8(n(\lceil n/64 \rceil + 1) + 1)$ байт глобальной памяти графического ускорителя. Соотношение между числом вершин графа и размером глобальной памяти, необходимой для его хранения, показано на рис. 6. Таким образом, для хранения матрицы смежности графа со 180 тысячами вершин необходимо около 3,78 Гбайт.

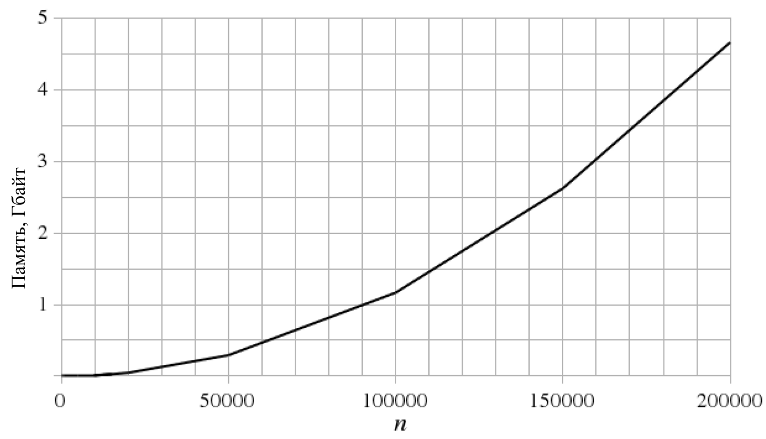


Рис. 6. Соотношение между числом вершин графа и размером глобальной памяти

Для хранения невзвешенного графа с n вершинами и m рёбрами, представленного в виде списка рёбер, необходимо $2(8(\log_2 n(\lceil m/64 \rceil + 1) + 1))$ байт. В этом случае потребуется около 3,34 Мбайт для хранения графа с 10^4 вершин и 10^6 рёбер. Для хранения дорожного графа США (около $24 \cdot 10^6$ вершин и $58 \cdot 10^6$ дуг) потребуется 348 Мбайт глобальной памяти. Файл данных этого графа расположен на сайте [24]. Для примера, графический ускоритель Kepler K40 оснащён 12 Гбайт памяти и 2 880 ядер CUDA [25].

3.3. Сравнение времени работы различных реализаций алгоритма Уоршалла

Для сравнения времени работы различных реализаций алгоритма Уоршалла на графических ускорителях использованы модифицированный и адаптированный Star-алгоритмы Уоршалла. Время работы в секундах последовательного алгоритма Уоршалла, модифицированного (листинг 2) и адаптированного (листинг 4) Star-алгоритмов приведено в табл. 2.

Таблица 2

Время работы последовательного алгоритма Уоршалла и его Star-версий

Количество вершин	1 000	2 000	3 000	4 000	5 000
Последовательный алгоритм	8,037	59,812	197,111	461,785	884,622
WARSHALL-C	4,827	11,161	23,363	42,661	64,454
WARSHALL-adapt	0,003	0,027	0,093	0,236	0,372

В работах [26 – 28] приводятся параллельные реализации алгоритма Уоршалла на GPU, но из-за различия используемых моделей графических ускорителей сравнить время их работы с временем работы Star-алгоритмов затруднительно.

Заключение

В работе приведена абстрактная модель ассоциативной параллельной обработки данных (STAR-машина). Обоснована актуальность реализации этой модели на современных архитектурах. Обоснован выбор графических ускорителей в качестве такой архитектуры. Приведена реализация этой модели на графических ускорителях с помощью технологии CUDA, которая позволяет получить на алгоритме Уоршалла ускорение в 13 раз на графе с 5 000 вершин.

Показано, что некоторые ассоциативные параллельные алгоритмы могут быть адаптированы для выполнения на графических ускорителях, если принимать во внимание архитектурные различия STAR-машины и графических ускорителей. Так, адаптация ассоциативной версии алгоритма Уоршалла под графические ускорители дала ускорение в 2 378 раз на графе с 5 000 вершин.

На данный момент Star-алгоритм записывается в виде процедуры на CPU и методы, реализующие базовые операции, запускают на графическом ускорителе функции, производящие вычисления. Исходя из этого, планируется продолжить работу над реализацией STAR-машины на графических ускорителях так, чтобы Star-алгоритм мог быть представлен в виде процедуры, запускаемой на GPU. Необходимо также сделать библиотеку базовых Star-процедур для выполнения на графических ускорителях. Заметим, что все базовые процедуры могут быть реализованы в виде процедур, запускаемых на GPU, чтобы свести к минимуму количество запусков ядер.

ЛИТЕРАТУРА

1. *Potter J. L.* Associative Computing: a Programming Paradigm for Massively Parallel Computers. Boston: Perseus Publishing, 1991. 304 p.
2. *Rudolph J. A.* A production implementation of an associative array processor — STARAN // AFIPS'72. N. Y.: ACM, 1972. P. 229–241.
3. *Foster C. C.* Content Addressable Parallel Processors. N. Y.: John Wiley & Sons, 1976. 233 p.
4. *Higuchi T., Furuya T., Handa K., et al.* IXM2: a parallel associative processor // ISCA'91. Proc. 18th Annual Intern. Symp. Computer Architecture. N. Y.: ACM, 1991. P. 22–31.
5. *Непомнящая А. Ш., Владыко М. А.* Сравнение моделей ассоциативного вычисления // Программирование. 1997. № 6. С. 41–50.
6. *Nepomniaschaya A. Sh.* Language STAR for associative and parallel computation with vertical data processing // Parallel Computing Technologies. Singapore: World Scientific, 1991. P. 258–265.
7. *Potter J., Baker J., Scott S., et al.* ASC: An Associative-Computing Paradigm // Computer. 1994. No. 27(11). P. 19–25.
8. *Nepomniaschaya A. Sh.* Basic associative parallel algorithms for vertical processing systems // Bulletin of the Novosibirsk Computing Center. 2009. Ser. Comp. Science. No. 9. P. 63–77.
9. *Nepomniaschaya A. Sh.* Constructions used in associative parallel algorithms for undirected graphs. Part 1 // Bulletin of the Novosibirsk Computing Center. 2013. Ser. Comp. Science. No. 35. P. 69–83.
10. *Nepomniaschaya A. Sh.* Constructions used in associative parallel algorithms for directed graphs // LNCS. 2015. V. 9251. P. 201–209.

11. *Nepomniaschaya A. Sh.* Solution of path problems using associative parallel processors // Intern. Conf. Parallel and Distributed Systems. Seoul: IEEE Computer Society Press, 1997. P. 610–617.
12. *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.
13. *Nepomniaschaya A. Sh.* An associative version of the Bellman — Ford algorithm for finding the shortest paths in directed graphs // LNCS. 2001. V. 2127. P. 285–292.
14. *Nepomniaschaya A. Sh.* Efficient implementation of Edmonds' algorithm for finding optimum branchings on associative parallel processors // Proc. 8th Intern. Conf. Parallel and Distributed Systems (ICPADS'01). KyongJu City: IEEE Computer Society Press, 2001. P. 3–8.
15. *Непомнящая А. Ш.* Сравнение алгоритмов Прима — Дейкстры и Краскала с помощью ассоциативного параллельного процессора // Кибернетика и системный анализ. 2000. № 2. С. 19–27.
16. *Nepomniaschaya A. Sh.* Representation of the Gabow algorithm for finding smallest spanning trees with a degree constraint on associative parallel processors // LNCS. 1996. V. 1123. P. 813–817.
17. *Nepomniaschaya A. Sh.* Associative parallel algorithms for dynamic edge update of minimum spanning trees // LNCS. 2003. V. 2763. P. 141–150.
18. *Nepomniaschaya A. Sh.* Associative parallel algorithm for dynamic reconstructing a minimum spanning tree after deletion of a vertex // LNCS. 2005. V. 3606. P. 151–173.
19. *Непомнящая А. Ш.* Ассоциативный параллельный алгоритм для динамической обработки минимального каркаса после добавления к графу новой вершины // Кибернетика и системный анализ. 2006. № 1. С. 19–31.
20. *Nepomniaschaya A. Sh.* Efficient implementation of the Italiano algorithms for updating the transitive closure on associative parallel processors // Fundamenta Informaticae. IOS Press, 2008. V. 89. No. 2–3. P. 313–329.
21. *Nepomniaschaya A. Sh.* Associative version of the Ramalingam decremental algorithm for dynamic updating the single-sink shortest paths subgraph // LNCS. 2009. V. 5698. P. 257–268.
22. *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.
23. *Mingxian J.* Associative operations from MASC to GPU // PDPTA-15 — The 21st Intern. Conf. on Parallel and Distributed Processing Techniques and Applications. Las Vegas: CSREA Press, 2015. P. 388–393.
24. `http://www.dis.uniroma1.it/challenge9/download.s` — 9th DIMACS Implementation Challenge — Shortest Paths. 2010.
25. `http://www.nvidia.ru/object/tesla-server-gpus-ru.html` — Высокопроизводительные вычисления для серверов| Графические процессоры Tesla|NVIDIA. 2016.
26. *Harish P. and Narayanan P.* Accelerating large graph algorithms on the GPU using CUDA // LNCS. 2007. V. 4873. P. 197–208.
27. *Katz G. J. and Kider Jr. T.* All-pairs shortest-paths for large graphs on the GPU // Proc. 23rd ACM SIGGRAPH/EUROGRAPHICS Symp. Graphics Hardware. Sarajevo: Eurographics Association, 2008. P. 47–55.
28. *Lund B. and Smith J. W.* A Multi-Stage CUDA Kernel for Floyd — Warshall. arXiv preprint arXiv:1001.4108

REFERENCES

1. *Potter J. L.* Associative Computing: a Programming Paradigm for Massively Parallel Computers. Boston, Perseus Publishing, 1991. 304 p.
2. *Rudolph J. A.* A production implementation of an associative array processor — STARAN. AFIPS'72, N. Y., ACM, 1972, pp. 229–241.
3. *Foster C. C.* Content Addressable Parallel Processors. N. Y., John Wiley & Sons, 1976. 233 p.
4. *Higuchi T., Furuya T., Handa K., et al.* IXM2: a parallel associative processor. ISCA'91. Proc. 18th Annual Intern. Symp. Computer Architecture. N. Y., ACM, 1991, pp. 22–31.
5. *Nepomnyashchaya A. Sh. and Vladyko M. A.* Sravnenie modeley assotsiativnogo vychisleniya [Compare of associative computation models]. Programmirovanie, 1997, no. 6, pp. 41–50. (in Russian)
6. *Nepomniaschaya A. Sh.* Language STAR for associative and parallel computation with vertical data processing. Parallel Computing Technologies. Singapore, World Scientific, 1991, pp. 258–265.
7. *Potter J., Baker J., Scott S., et al.* ASC: An Associative-Computing Paradigm. Computer, 1994, no. 27(11), pp. 19–25.
8. *Nepomniaschaya A. Sh.* Basic associative parallel algorithms for vertical processing systems. Bulletin of the Novosibirsk Computing Center, 2009, ser. Comp. Science, no. 9, pp. 63–77.
9. *Nepomniaschaya A. Sh.* Constructions used in associative parallel algorithms for undirected graphs. Part 1. Bulletin of the Novosibirsk Computing Center, 2013, ser. Comp. Science, no. 35, pp. 69–83.
10. *Nepomniaschaya A. Sh.* Constructions used in associative parallel algorithms for directed graphs. LNCS, 2015, vol. 9251, pp. 201–209.
11. *Nepomniaschaya A. Sh.* Solution of path problems using associative parallel processors. Intern. Conf. Parallel and Distributed Systems, Seoul, IEEE Computer Society Press, 1997, pp. 610–617.
12. *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.
13. *Nepomniaschaya A. Sh.* An associative version of the Bellman — Ford algorithm for finding the shortest paths in directed graphs. LNCS, 2001, vol. 2127, pp. 285–292.
14. *Nepomniaschaya A. Sh.* Efficient implementation of Edmonds' algorithm for finding optimum branchings on associative parallel processors. Proc. 8th Intern. Conf. Parallel and Distributed Systems (ICPADS'01), KyongJu City, IEEE Computer Society Press, 2001, pp. 3–8.
15. *Nepomnyashchaya A. Sh.* Sravnenie algoritmov Prima — Deykstry i Kraskala s pomoshch'yu assotsiativnogo parallel'nogo protsessora [Compare of Prima — Dijkstra and Kruskal algorithms using associative parallel processor]. Kibernetika i sistemnyy analiz, 2000, no 2, pp. 19–27. (in Russian)
16. *Nepomniaschaya A. Sh.* Representation of the Gabow algorithm for finding smallest spanning trees with a degree constraint on associative parallel processors. LNCS, 1996, vol. 1123, pp. 813–817.
17. *Nepomniaschaya A. Sh.* Associative parallel algorithms for dynamic edge update of minimum spanning trees. LNCS, 2003, vol. 2763, pp. 141–150.
18. *Nepomniaschaya A. Sh.* Associative parallel algorithm for dynamic reconstructing a minimum spanning tree after deletion of a vertex. LNCS, 2005, vol. 3606, pp. 151–173.
19. *Nepomnyashchaya A. Sh.* Assotsiativnyy parallel'nyy algoritm dlya dinamicheskoy obrabotki minimal'nogo karkasa posle dobavleniya k grafu novoy vershiny [An associative parallel

algorithm for dynamic update of a minimum spanning tree after adding a new vertex to a graph]. Kibernetika i sistemnyy analiz, 2006, no. 1, pp. 19–31. (in Russian)

20. *Nepomniaschaya A. Sh.* Efficient implementation of the Italiano algorithms for updating the transitive closure on associative parallel processors. Fundamenta Informaticae, IOS Press, 2008, vol. 89, no. 2–3, pp. 313–329.
21. *Nepomniaschaya A. Sh.* Associative version of the Ramalingam decremental algorithm for dynamic updating the single-sink shortest paths subgraph. LNCS, 2009, vol. 5698, pp. 257–268.
22. *Nepomnyashchaya A. Sh.* Assotsiativnaya versiya algoritma Ramalingama dlya dinamicheskoy obrabotki podgrafa kratchayshikh putey posle dobavleniya k grafu novoy dugi [Associative version of the ramalingam algorithm for dynamic update of the shortest-path subgraph after insertion of a new edge to the graph]. Kibernetika i sistemnyy analiz, 2012, no. 3, pp. 45–57. (in Russian)
23. *Mingxian J.* Associative operations from MASC to GPU. PDPTA-15 — The 21st Intern. Conf. on Parallel and Distributed Processing Techniques and Applications, Las Vegas, CSREA Press, 2015, pp. 388–393.
24. `http://www.dis.uniroma1.it/challenge9/download.s` — 9th DIMACS Implementation Challenge — Shortest Paths, 2010.
25. `http://www.nvidia.ru/object/tesla-server-gpus-ru.html`
26. *Harish P. and Narayanan P.* Accelerating large graph algorithms on the GPU using CUDA. LNCS, 2007, vol. 4873, pp. 197–208.
27. *Katz G. J. and Kider Jr. T.* All-pairs shortest-paths for large graphs on the GPU. Proc. 23rd ACM SIGGRAPH/EUROGRAPHICS Symp. Graphics Hardware, Sarajevo, Eurographics Association, 2008, pp. 47–55.
28. *Lund B. and Smith J. W.* A Multi-Stage CUDA Kernel for Floyd — Warshall. arXiv preprint arXiv:1001.4108

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 512.54

ОБ ОДНОМ АЛГОРИТМЕ ВЫЧИСЛЕНИЯ ФУНКЦИЙ РОСТА В КОНЕЧНЫХ ДВУПОРОЖДЁННЫХ ГРУППАХ ПЕРИОДА 5[1]

А. А. Кузнецов

Сибирский государственный аэрокосмический университет, г. Красноярск, Россия

Пусть $B_0(2,5) = \langle a_1, a_2 \rangle$ — наибольшая конечная двупорождённая бернсайдова группа периода 5, порядок которой равен 5^{34}. Для каждого элемента данной группы существует уникальное коммутаторное представление вида $a_1^{\alpha_1} \cdot a_2^{\alpha_2} \cdot \ldots \cdot a_{34}^{\alpha_{34}}$, где $\alpha_i \in \mathbb{Z}_5$, $i = 1, 2, \ldots, 34$. Здесь a_1 и a_2 — порождающие элементы $B_0(2,5)$; $a_3, \ldots, a_{34}$ — коммутаторы, которые вычисляются рекурсивно через a_1 и a_2. Определим фактор-группу группы $B_0(2,5)$ следующего вида: $B_k = B_0(2,5)/\langle a_{k+1}, \ldots, a_{34} \rangle$. Очевидно, что $|B_k| = 5^k$. Предложен новый алгоритм, при помощи которого вычислены функции роста B_k относительно порождающих множеств $\{a_1, a_2\}$ и $\{a_1, a_1^{-1}, a_2, a_2^{-1}\}$ для $k = 15$, 16, 17. На основе полученных данных вычислены диаметр и средний диаметр соответствующих графов Кэли.

Ключевые слова: *функция роста, группа Бернсайда, граф Кэли.*

DOI 10.17223/20710410/33/10

AN ALGORITHM FOR COMPUTATION OF THE GROWTH FUNCTIONS IN FINITE TWO-GENERATED GROUPS OF EXPONENT 5

A. A. Kuznetsov

Siberian State Aerospace University, Krasnoyarsk, Russia

E-mail: alex_kuznetsov80@mail.ru

Let $B_0(2,5) = \langle a_1, a_2 \rangle$ be the largest 2-generator Burnside group of exponent 5. It has the order 5^{34}. There is a power commutator representation of $B_0(2,5)$. In this case, every element of the group can be uniquely represented as $a_1^{\alpha_1} \cdot a_2^{\alpha_2} \cdot \ldots \cdot a_{34}^{\alpha_{34}}$, where $\alpha_i \in \mathbb{Z}_5$, $a_i \in B_0(2,5)$, $i = 1, 2, \ldots, 34$. Here, a_1 and a_2 are generators of $B_0(2,5)$, commutators $a_3, \ldots, a_{34}$ are recursively defined by a_1 and a_2. We define $B_k = B_0(2,5)/\langle a_{k+1}, \ldots, a_{34} \rangle$ as a quotient of $B_0(2,5)$. It is clearly that $|B_k| = 5^k$. A new algorithm for computing the growth function of B_k is created. Using this algorithm, we calculated the growth functions of B_k relative to generating sets $\{a_1, a_2\}$ and $\{a_1, a_1^{-1}, a_2, a_2^{-1}\}$ for $k = 15$, 16, 17.

Keywords: *Burnside group, the growth function.*

[1]Работа поддержана грантом Президента РФ (проект № МД-3952.2015.9).

Введение

Одним из важных инструментов для определения строения группы является изучение её роста относительно фиксированного порождающего множества. Пусть $G = \langle X \rangle$. Шаром K_s радиуса s группы G будем называть множество всех её элементов, которые могут быть представлены в виде групповых слов в алфавите X длины не больше s. Для каждого целого неотрицательного s можно определить функцию $F(s)$ роста группы, которая равна числу элементов группы G относительно X, представимых в виде несократимых групповых слов длины s. Таким образом,

$$F(0) = |K_0| = 1,\ F(s) = |K_s| - |K_{s-1}| \text{ при } s \in \mathbb{N}.$$

Как правило, функцию роста конечной группы представляют в виде таблицы, в которую записывают ненулевые значения $F(s)$.

К сожалению, вычисление функции роста большой конечной группы является хотя и разрешимой, но весьма сложной проблемой. Это связано с тем, что в общем случае задача по определению минимального слова элемента группы, как показали С. Ивен и О. Голдрейх [1], является NP-трудной. Таким образом, в наихудшем случае количество элементарных операций, которые необходимо выполнить для решения указанной задачи, представляет собой экспоненциальную функцию от $|X|$.

Отметим также, что при вычислении функции роста группы мы параллельно выясняем характеристики соответствующего графа Кэли, например диаметр и средний диаметр [2]. Пусть $F(s_0) > 0$, но $F(s_0 + 1) = 0$, тогда s_0 является диаметром графа Кэли группы G в алфавите порождающих X, который будем обозначать $D_X(G)$. Соответственно средний диаметр $\overline{D}_X(G)$ равен $\dfrac{1}{|G|}\sum\limits_{s=0}^{s_0} s \cdot F(s)$.

Как известно, благодаря своим замечательным свойствам графы Кэли нашли применение при моделировании топологий многопроцессорных вычислительных систем (МВС) — суперкомпьютеров [3], а также дата-центров [4]. И вполне возможно, что в недалёком будущем понадобятся знания о сверхбольших графах, которые будут востребованы при проектировании распределённых систем, пиковая производительность которых будет достигать 1 экзафлопс и выше.

Одной из широко применяемых топологий МВС является k-мерный гиперкуб, который задаётся группой $B(k, 2)$ — k-порождённой бернсайдовой группой периода 2. Группа $B(k, 2)$ имеет простую структуру и равна прямому произведению k экземпляров циклической группы порядка 2. Обобщением гиперкуба является n-мерный тор, который порождается прямым произведением n экземпляров циклических подгрупп, порядки которых могут не совпадать. В работе [5] исследованы графы Кэли групп $B(k, 3)$, т. е. групп периода 3, а также проведён сравнительный анализ данных графов с гиперкубами. Анализ выявил, что графы $B(k, 3)$ обладают более предпочтительными характеристиками, чем $B(k, 2)$. Это означает, что при парном сравнении графов $B(k_1, 3)$ и $B(k_2, 2)$ с примерно одинаковым количеством вершин первые будут иметь меньшие диаметры, средние диаметры и степени. В связи с этим представляет интерес задача по исследованию графов Кэли конечных бернсайдовых групп другого периода.

Пусть $B_0(2, 5) = \langle a_1, a_2 \rangle$ — максимальная конечная двупорождённая бернсайдова группа периода 5, порядок которой равен 5^{34} [6]. Используя систему компьютерной алгебры GAP, нетрудно получить *pc*-представление (*Power Commutator presentation*) данной группы [3, 7]. В этом случае каждый элемент $g \in B_0(2, 5)$ может быть однозначно записан в следующем виде:

$$g = a_1^{\alpha_1} \cdot a_2^{\alpha_2} \cdot \ldots \cdot a_{34}^{\alpha_{34}},\ \alpha_i \in \mathbb{Z}_5,\ i = 1, 2, \ldots, 34.$$

Здесь a_1 и a_2 — порождающие элементы $B_0(2,5)$; $a_3, \ldots, a_{34}$ — коммутаторы, которые вычисляются рекурсивно через a_1 и a_2 [6].

Обозначим через B_k фактор-группу $B_0(2,5)$ следующего вида:

$$B_k = B_0(2,5)/\langle a_{k+1}, \ldots, a_{34}\rangle.$$

Очевидно, что $|B_k| = 5^k$ и для всех $g \in B_k$ выполняется

$$g = a_1^{\alpha_1} \cdot a_2^{\alpha_2} \cdot \ldots \cdot a_k^{\alpha_k}.$$

Помимо прикладного интереса, существует и другая причина повышенного внимания исследователей к функциям роста B_k и особенно к $B_{34} = B_0(2,5)$. Это объясняется тем, что полученная информация может оказаться полезной при решении открытой проблемы о конечности $B(2,5)$ — свободной двупорождённой бернсайдовой группы периода 5. Если $B(2,5)$ конечна, то $B_{34} = B(2,5)$. Однако в обозримом будущем вычислить функцию роста B_{34} едва ли возможно, поскольку количество её элементов очень велико: $5^{34} = 582076609134674072265625 \approx 5 \cdot 10^{23}$.

К. А. Филипповым в работе [8] вычислена функция роста группы B_{14} в алфавите $A_2 = \{a_1, a_2\}$. Аналогичная задача для симметричного порождающего множества $A_4 = \{a_1, a_1^{-1}, a_2, a_2^{-1}\}$ решена Ч. Симсом [9]. В обоих случаях решение было получено при помощи длительных компьютерных вычислений. Отметим также, что Симс в своём алгоритме использовал элегантный трюк, основанный на применении к B_{14} группы автоморфизмов, сохраняющих длину группового слова. Это дало возможность свести задачу к рассмотрению только 1/16 части элементов группы. Подобный трюк для несимметричного порождающего множества менее эффективен, поскольку позволяет сократить количество элементов не более чем в 2 раза.

Дальнейшее изучение функций роста групп B_k для $k > 14$ сталкивается с серьёзными вычислительными трудностями ввиду их больших порядков.

В данной работе представлен эффективный алгоритм для вычисления функций роста указанных групп, который даёт возможность преодолеть существующий барьер. Компьютерная реализация алгоритма позволила вычислить функции роста групп B_k для $k = 15$, 16 и 17.

В п. 1 рассматривается базовый алгоритм для вычисления функции роста конечной группы, а также исследуется его сложность. В п. 2 алгоритм модифицируется для исследования групп B_k. В п. 3 приведены результаты компьютерных вычислений по модифицированному алгоритму.

1. Описание базового алгоритма

В качестве отправной точки далее представлен общий алгоритм A-I, вычисляющий функцию роста произвольной конечной группы G, заданной фиксированным порождающим множеством X.

Лемма 1. Алгоритм A-I корректен, т. е. он за конечное число шагов вычислит функцию роста любой конечной группы G, заданной фиксированным порождающим множеством X.

Доказательство. По построению A-I выражает каждый элемент группы G в виде группового слова наименьшей длины в алфавите X. После каждого прохода от п. 2 до п. 8 множество K_s представляет собой шар радиуса s группы G относительно X. Поскольку количество элементов группы конечно, через конечное число шагов, при некотором $s = s_0 + 1$, все $g \in G$ будут записаны в виде слов, длины которых меньше

Алгоритм 1. A-I

Вход: $X = \{x_1, x_2, \ldots, x_m\}$ — порождающее множество G
Выход: функция роста $F(s)$ группы G

1: $s := 0$, $F(0) := 1$, $K_0 := \{e\}$, $P := K_0$
2: $s := s + 1$
3: $K_s := K_{s-1}$
4: **Для всех** $x \in X$ и $p \in P$
5: $\quad g = x \cdot p$ (или $g = p \cdot x$)
6: $\quad$ **Если** $g \notin K_s$, **то**
$\qquad K_s := K_s \cup \{g\}$
7: $P := K_s - K_{s-1}$
8: $F(s) := |P|$
9: **Если** $F(s) > 0$, **то**
$\quad$ переход в п. 2,
10: **иначе**
11: $\quad s_0 := s - 1$, переход в п. 12
12: $D_X(G) := s_0$, $\overline{D}_X(G) := \dfrac{1}{|K_{s_0}|} \sum\limits_{s=0}^{s_0} s \cdot F(s)$, выход

s_0+1. Получим $F(s_0+1) = 0$ и $|K_{s_0}| = |G|$, что означает остановку алгоритма. При этом $D_X(G) = s_0$ и $\overline{D}_X(G) = \dfrac{1}{|K_{s_0}|} \sum\limits_{s=0}^{s_0} s \cdot F(s)$. ■

Несмотря на то, что алгоритм A-I хорошо известен специалистам, автору не удалось найти публикаций, в которых были бы вычислены оценки его временной сложности. Например, в системе компьютерной алгебры GAP реализован указанный алгоритм, вычисляющий функцию роста группы подстановок. Однако в документации к системе отсутствуют сведения о его сложности. Для устранения указанного пробела воспользуемся машиной с произвольным доступом к памяти — RAM-машиной, а также асимптотическим анализом сложности [10]. Далее термины «временная сложность» и «сложность» будем считать равнозначными; кроме этого, введём обозначения:

— $T(|G|)$ — сложность алгоритма A-I;
— $\mathrm{O}(f(|G|))$ — верхняя асимптотическая оценка сложности;
— $\Omega(f(|G|))$ — нижняя асимптотическая оценка сложности;
— $\Theta(f(|G|))$ — одновременно верхняя и нижняя оценки сложности.

Нам понадобятся следующие вспомогательные утверждения.

Лемма 2. В процессе работы алгоритма A-I необходимо вычислить и проверить $|X| \cdot |G|$ групповых слов.

Доказательство. При вычислении слов длины $s \geqslant 1$ следует обработать $F(s-1)|X|$ элементов. В итоге получим

$$\sum_{s=1}^{D_X(G)+1} F(s-1)|X| = |X| \sum_{s=1}^{D_X(G)+1} F(s-1) = |X| \cdot |G|.$$

Лемма доказана. ■

Лемма 3. Пусть $N(|G|)$ — суммарное количество проверок вида $g \notin K_s$ алгоритма А-I. Тогда верно следующее неравенство:

$$\frac{1}{2}|G|^2 + \left(|X| - \frac{3}{2}\right)|G| + 1 \leqslant N(|G|) \leqslant \left(|X| - \frac{1}{2}\right)|G|^2 + \frac{1}{2}|G|.$$

Доказательство. Рассмотрим сначала те групповые слова, которые войдут в K_s. Всего таких элементов, не считая единицы, $(|G| - 1)$. Для первого из них необходимо осуществить одну проверку (с единицей группы), для второго — две и т. д. В итоге получим $1 + 2 + \ldots + (|G| - 1) = |G|(|G| - 1)/2$ операций.

Теперь рассмотрим те слова, которые не войдут в K_s. Согласно лемме 2, количество таких элементов равно $|X| \cdot |G| - (|G| - 1)$. В лучшем случае каждый из этих элементов будет проверен один раз и исключён, а в худшем — для каждого из них потребуется осуществить $|G|$ сравнений.

Таким образом, общее число проверок в худшем и лучшем случаях равно $|G|(|G| - 1)/2 + |X| \cdot |G| - (|G| - 1)$ и $|G|(|G| - 1)/2 + (|X| \cdot |G| - (|G| - 1))|G|$ соответственно. После элементарных преобразований получим нужное неравенство. ■

Теорема 1. $T(|G|) \in \Omega(|G|^2)$ и $T(|G|) \in \mathrm{O}(|X| \cdot |G|^2)$.

Доказательство. При $|G| \to \infty$ сложностью операций всех пунктов алгоритма А-I, кроме четвертого, следует пренебречь. Поскольку количество элементарных операций, необходимое для вычисления произведения двух элементов группы, конечно и ограничено $\Theta(1)$, то суммарная сложность $M(|G|)$ п. 5 ограничена $\Theta(|X| \cdot |G|)$. Согласно лемме 3, получим $|G|^2/2 \leqslant N(|G|) \Rightarrow N(G) \in \Omega(|G|^2)$ и $N(|G|) \leqslant |X| \cdot |G|^2 \Rightarrow$ $\Rightarrow N(|G|) \in \mathrm{O}(|X| \cdot |G|^2)$. Так как $T(|G|) = M(|G|) + N(|G|)$, получим $T(|G|) \in \Omega(|G|^2)$ и $T(|G|) \in \mathrm{O}(|X| \cdot |G|^2)$. ■

Следствие 1. Если $|X| \ll |G|$, то $T(|G|) \in \Theta(|G|^2)$.

Следствие 2. Если $|X| \sim |G|$, то $T(|G|) \in \mathrm{O}(|G|^3)$.

Если порядок группы G достаточно большой, то наиболее критическими участками алгоритма А-I будут пп. 5 и 6, т. е. умножение элементов группы и проверка, позволяющая определить, встречался ранее такой элемент или нет.

2. Модификация алгоритма А-I для исследования групп B_k

В данном случае $X = A_2$ или A_4. Рассмотрим п. 5 алгоритма.

Существуют два способа умножения элементов в группах B_k: собирательный процесс [3, 7] и метод полиномов Холла [11]. Вычислительные эксперименты показали, что второй способ позволяет значительно быстрее собирательного процесса (по крайней мере на порядок) умножать элементы в группах B_k [12].

Следующий вопрос: как эффективнее умножать порождающие $x \in X$ на элементы группы $p \in B_k$ — слева (1 способ) или справа (2 способ), т. е. $x \cdot p$ или $p \cdot x$? Кроме этого, была предложена идея при умножении слева использовать пять вариантов полиномов в зависимости от значения степени α_1 элемента p (3 способ). Для ответа на данный вопрос необходимо вычислить суммарное число операций $t(k)$ в полиномах Холла, необходимое для выполнения одного произведения. В полиномах имеются три вида действий: сложение, умножение и остаток от деления числа на 5. На рис. 1 представлены графики $t(k)$ для каждого из трёх способов (в предположении, что указанные действия имеют одинаковую сложность равную 1).

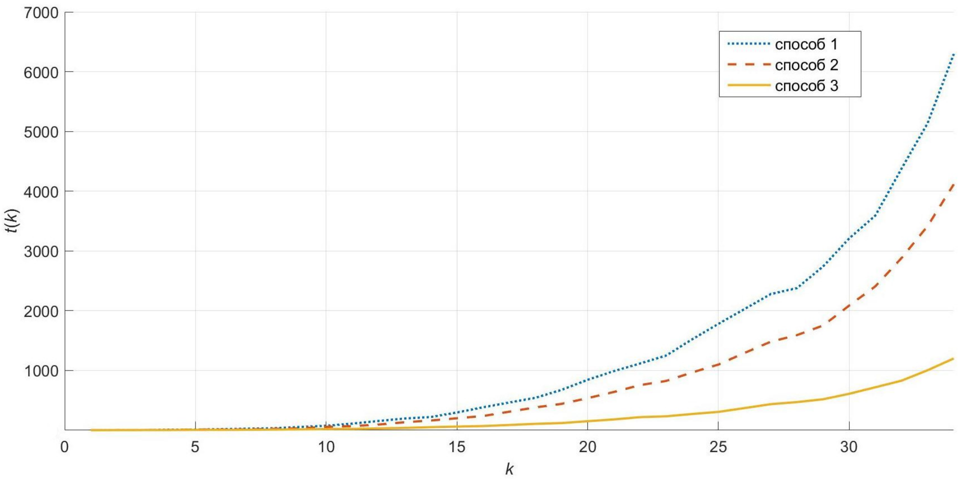


Рис. 1. Графики сложности $t(k)$ для одного произведения

Рис. 1 наглядно показывает, что третий способ самый эффективный. Этот факт был учтён при реализации п. 5 алгоритма.

Теперь обратимся к п. 6 алгоритма А-I. Для его оптимизации потребуется каждому элементу $g \in B_k$ присвоить уникальный порядковый номер. Пусть $g = a_1^{\alpha_1} \cdot a_2^{\alpha_2} \cdot \ldots \cdot a_k^{\alpha_k}$ — произвольный элемент B_k. Определим биективное отображение ϕ следующего вида:

$$g \stackrel{\phi}{\longleftrightarrow} n_g = \alpha_1 \alpha_2 \ldots \alpha_k.$$

Здесь n_g представляет собой целое неотрицательное число, записанное в пятеричной системе счисления, которое возьмём в качестве порядкового номера g. Легко увидеть, что n_g пробегает все значения от 0 до $(5^k - 1)$.

Модифицируем А-I следующим образом. В п. 1 добавим булев вектор V размерности 5^k, все элементы которого первоначально равны 0. Для удобства индексация элементов V начинается с 0, т. е. последний элемент вектора имеет индекс $(5^k - 1)$. Заменим п. 6 алгоритма А-I на следующий:
6: **Если** $V_{n_g} = 0$, **то** $V_{n_g} := 1$ и $K_s := K_s \cup \{g\}$.

Данная модификация позволяет избежать трудоёмкой процедуры поиска элемента g в множестве K_s. Будем считать, что для выполнения п. 6 алгоритма требуется не более $t(g)$ элементарных операций.

Несмотря на то, что порядки B_k ограничены сверху, имеет смысл рассмотреть асимптотическую оценку сложности модифицированной версии алгоритма, поскольку данные группы содержат достаточно большое число элементов при $k > 8$.

Теорема 2. Пусть $T(|B_k|)$ — вычислительная сложность модифицированного алгоритма А-I. Тогда верно, что $T(|B_k|) \in \Theta(|B_k|)$.

Доказательство. При вычислении асимптотической оценки следует учитывать сложность только наиболее трудоёмких фрагментов алгоритма, т. е. пп. 5 и 6. Поскольку $T(|G|) = (t(k) + t(g))|X| \cdot |B_k|$, $|X| \ll |B_k|$, $t(k) \ll |B_k|$ и $t(g) \ll |B_k|$, получим $T(G) \in \Theta(|B_k|)$. ■

Также отметим, что в п. 5 все элементы g вычисляются независимо друг от друга, поэтому этот участок алгоритма можно легко распараллелить. В этом случае сначала параллельно вычисляются все произведения g, а затем для каждого элемента получившегося массива последовательно выполняется п. 6.

Легко увидеть, что перечисленные модификации алгоритма A-I применительно к группам B_k значительно повышают его производительность.

3. Компьютерная реализация алгоритма

Модифицированный алгоритм A-I реализован на языке `C++` и апробирован на 4-процессорном компьютере (на каждом по 16 ядер) с общей памятью объёмом 256 Гбайт. В результате вычислены функции роста групп B_k для $k = 15$, 16 и 17 для порождающих множеств $A_2 = \{a_1, a_2\}$ и $A_4 = \{a_1, a_1^{-1}, a_2, a_2^{-1}\}$. Затраченное на расчёты время примерно равно 1 час для $k = 15$, 5 часов — для $k = 16$ и около суток — для $k = 17$. На рис. 2–7 представлены графики функций роста указанных групп.

В таблице приведены характеристики графов Кэли групп B_k, вычисленные в настоящей работе, а также, для полноты картины, уже известные. Значения средних диаметров округлены до ближайших целых.

Отметим, что для решения задачи маршрутизации в графе Кэли, после того как будут получены минимальные слова группы в результате работы алгоритма A-I, можно будет воспользоваться алгоритмом из [4], который имеет сложность $\mathrm{O}(D_X^2(G))$.

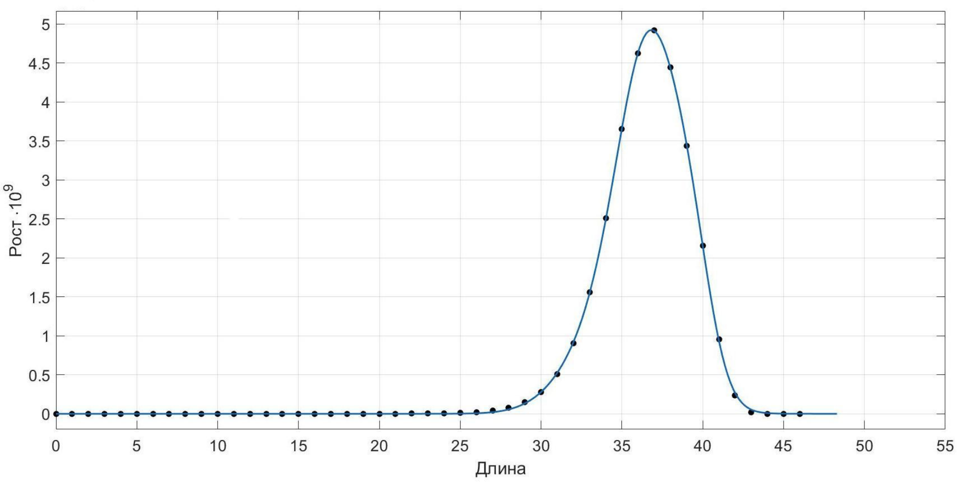


Рис. 2. График функции роста группы B_{15} в алфавите A_2

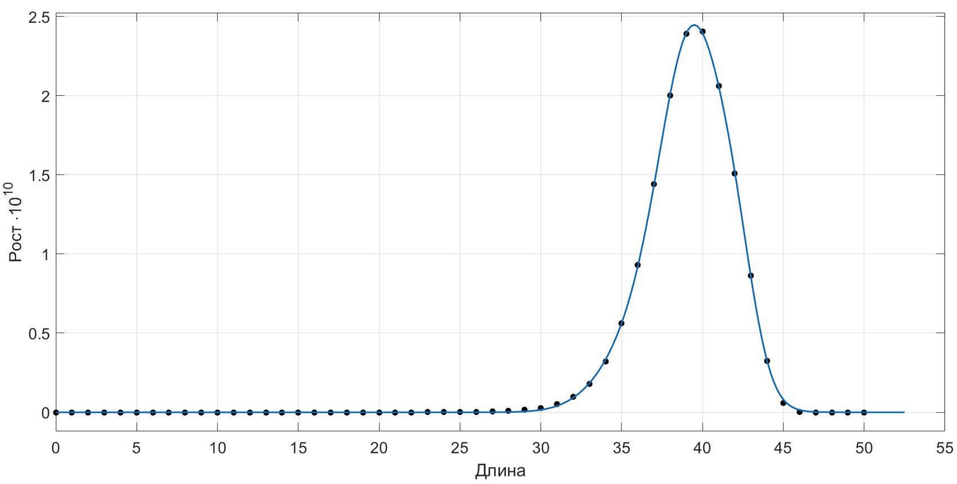


Рис. 3. График функции роста группы B_{16} в алфавите A_2

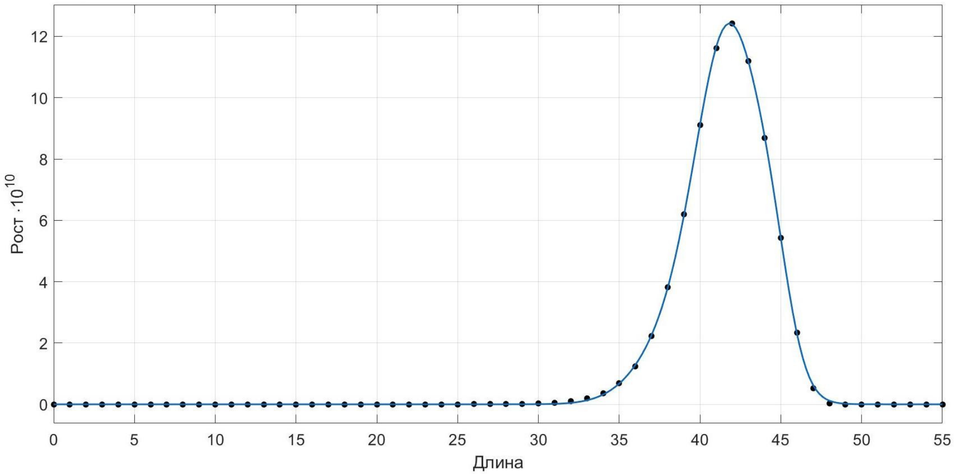


Рис. 4. График функции роста группы B_{17} в алфавите A_2

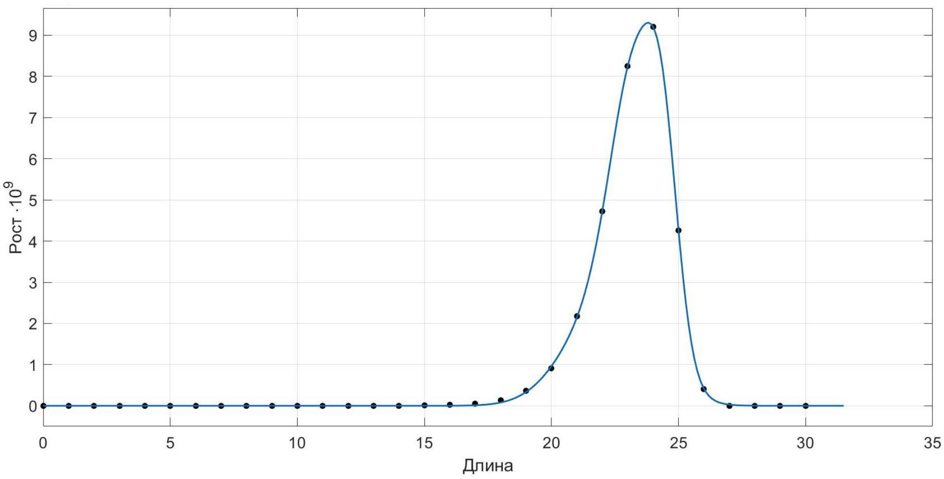


Рис. 5. График функции роста группы B_{15} в алфавите A_4

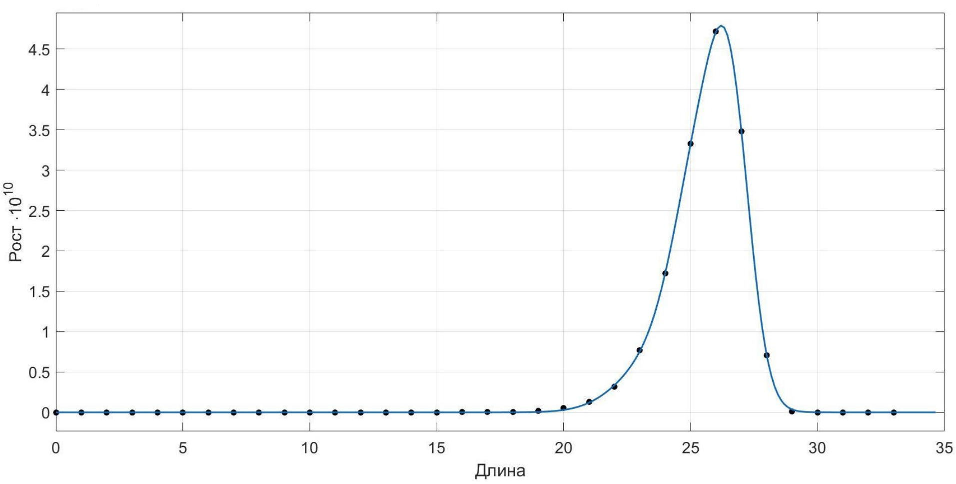


Рис. 6. График функции роста группы B_{16} в алфавите A_4

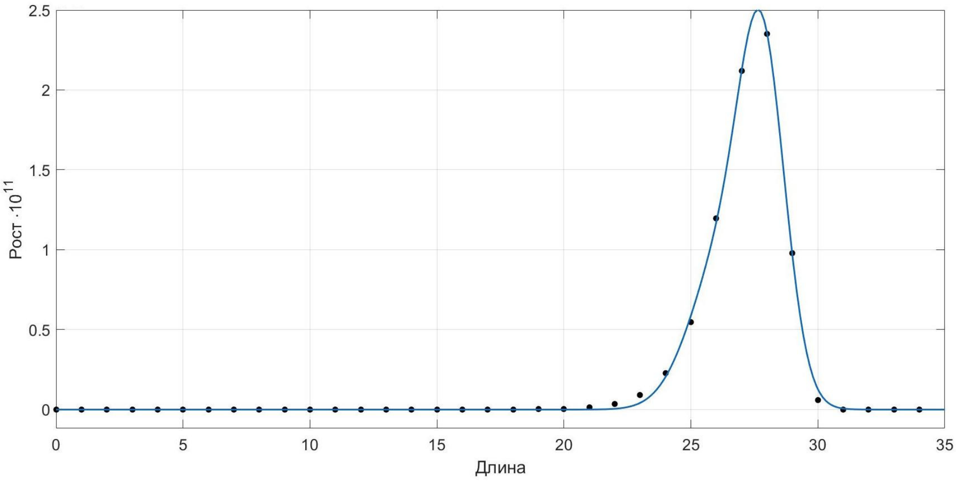


Рис. 7. График функции роста группы B_{17} в алфавите A_4

Характеристики графов Кэли групп B_k

k	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$D_{A_2}(B_k)$	8	10	13	20	20	25	30	31	32	35	40	40	45	46	50	55
$\overline{D}_{A_2}(B_k)$	4	6	8	12	14	16	20	22	24	27	29	31	34	36	39	42
$D_{A_4}(B_k)$	4	6	9	10	13	15	19	20	20	24	26	28	30	30	33	34
$\overline{D}_{A_4}(B_k)$	2	4	5	7	9	10	13	14	15	17	19	21	22	23	26	27

ЛИТЕРАТУРА

1. *Even S. and Goldreich O.* The Minimum Length Generator Sequence is NP-hard // J. Algorithms. 1981. No. 2. P. 311–313.
2. *Кузнецов А. А., Кузнецова А. С.* Параллельный алгоритм для исследования графов Кэли групп подстановок // Вестник СибГАУ. 2014. №1. С. 34–39.
3. *Holt D., Eick B., and O'Brien E.* Handbook of Computational Group Theory. Boca Raton: Chapman & Hall/CRC Press, 2005. 514 p.
4. *Camelo M., Papadimitriou D., Fàbrega L., and Vilà P.* Efficient routing in data center with underlying Cayley graph // Proc. 5th Workshop on Complex Networks CompleNet. 2014. P. 189–197.
5. *Кузнецов А. А.* Графы Кэли бернсайдовых групп периода 3 // Сибирские электронные математические известия. 2015. Т. 12. С. 248–254.
6. *Havas G., Wall G., and Wamsley J.* The two generator restricted Burnside group of exponent five // Bull. Austral. Math. Soc. 1974. No. 10. P. 459–470.
7. *Sims C.* Fast multiplication and growth in groups // Proc. 1998 Intern. Symp. Symbolic Algebraic Computation. 1998. P. 165–170.
8. *Филиппов К.А.* О диаметре Кэли одной подгруппы группы $B_0(2, 5)$ // Вестник СибГАУ. 2012. №1. С. 234–236.
9. *Sims C.* Computation with Finitely Presented Groups. Cambridge: Cambridge University Press, 1994. 628 p.
10. *Скиена С.* Алгоритмы. Руководство по разработке. СПб.: БХВ-Петербург, 2014. 720 с.
11. *Hall P.* Nilpotent Groups: Notes of Lectures Given at the Canadian Mathematical Congress Summer Seminar, University of Alberta, 12–30 August, 1957. London: Queen Mary College, 1969.

12. *Кузнецов А. А., Кузнецова А. С.* Быстрое умножение элементов в конечных двупорождённых группах периода пять // Прикладная дискретная математика. 2013. №1. С. 110–116.

REFERENCES

1. *Even S. and Goldreich O.* The Minimum Length Generator Sequence is NP-hard. J. Algorithms, 1981, no. 2, pp. 311–313.
2. *Kuznetsov A. A. and Kuznetsova A. S.* Parallel'nyy algoritm dlya issledovaniya grafov Keli grupp podstanovok [A parallel algorithm for study of the Cayley graphs of permutetion groups]. Vestnik SibGAU, 2014, no. 1, pp. 34–39. (in Russian)
3. *Holt D., Eick B., and O'Brien E.* Handbook of Computational Group Theory. Boca Raton, Chapman & Hall/CRC Press, 2005. 514 p.
4. *Camelo M., Papadimitriou D., Fàbrega L., and Vilà P.* Efficient routing in data center with underlying Cayley graph. Proc. 5th Workshop on Complex Networks CompleNet, 2014, pp. 189–197.
5. *Kuznetsov A. A.* Grafy Keli bernsaydovykh grupp perioda 3 [The Cayley graphs of Burnside groups of exponent 3]. Siberian Electronic Math. Reports, 2015, vol. 12, pp. 248–254. (in Russian)
6. *Havas G., Wall G., and Wamsley J.* The two generator restricted Burnside group of exponent five. Bull. Austral. Math. Soc., 1974, no. 10, pp. 459–470.
7. *Sims C.* Fast multiplication and growth in groups. Proc. 1998 Intern. Symp. Symbolic Algebraic Computation, 1998, pp. 165–170.
8. *Filippov K.A.* O diametre Keli odnoy podgruppy gruppy $B_0(2,5)$ [About Cayley diameter of one subgroup of the group $B_0(2,5)$]. Vestnik SibGAU, 2012, no. 1, pp. 234–236. (in Russian)
9. *Sims C.* Computation with Finitely Presented Groups. Cambridge, Cambridge University Press, 1994. 628 p.
10. *Skiena C.* Algoritmy. Rukovodstvo po razrabotke [Algorithms. Guide for the Development]. SPb., BKhV-Petersburg Publ., 2014. (in Russian)
11. *Hall P.* Nilpotent Groups: Notes of Lectures Given at the Canadian Mathematical Congress Summer Seminar, University of Alberta, 12–30 August, 1957. London, Queen Mary College, 1969.
12. *Kuznetsov A. A. and Kuznetsova A. S.* Bystroe umnozhenie elementov v konechnykh dvuporozhdennykh gruppakh perioda pyat' [Fast multiplication in finite two-generated groups of exponent five]. Prikladnaya Diskretnaya Matematika, 2013, no. 1, pp. 110–116. (in Russian)

СВЕДЕНИЯ ОБ АВТОРАХ

ГОРОДИЛОВА Анастасия Александровна — аспирантка Института математики им. С. Л. Соболева СО РАН, г. Новосибирск. E-mail: **gorodilova@math.nsc.ru**

ДУРНЕВ Валерий Георгиевич — профессор, доктор физико-математических наук, заведующий кафедрой компьютерной безопасности и математических методов обработки информации Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **durnev@uniyar.ac.ru**

ЗЕТКИНА Алена Игоревна — магистрантка кафедры мировой экономики и статистики Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **aizetkina@gmail.ru**

ЗЕТКИНА Оксана Валерьевна — доцент, кандидат экономических наук, доцент кафедры мировой экономики и статистики Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **oks_68@mail.ru**

ЗУБОВ Анатолий Юрьевич — кандидат физико-математических наук, доцент, старший научный сотрудник Московского государственного университета им. М. В. Ломоносова, г. Москва. E-mail: **zubovanatoly@yandex.ru**

ИЛЬЕВ Артем Викторович — аспирант Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **artyom_iljev@mail.ru**

ИЛЬЕВ Виктор Петрович — доктор физико-математических наук, доцент, старший научный сотрудник Института математики им. С. Л. Соболева СО РАН, г. Новосибирск; профессор Омского государственного университета, г. Омск.
E-mail: **iljev@mail.ru**

КУЗНЕЦОВ Александр Алексеевич — доктор физико-математических наук, профессор, директор института Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнёва, г. Красноярск.
E-mail: **alex_kuznetsov80@mail.ru**

МАГОМЕДОВ Абдулкарим Магомедович — доктор физико-математических наук, профессор, заведующий кафедрой дискретной математики и информатики Дагестанского государственного университета, г. Махачкала.
E-mail: **magomedtagir1@yandex.ru**

МУРИН Дмитрий Михайлович — кандидат физико-математических наук, старший преподаватель кафедры компьютерной безопасности и математических методов обработки информации Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **nirum87@mail.ru**

НЕПОМНЯЩАЯ Анна Шмилевна — кандидат физико-математических наук, старший научный сотрудник Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **anep@ssd.sscc.ru**

РЫБАЛОВ Александр Николаевич — кандидат физико-математических наук, старший научный сотрудник Института математики им. С. Л. Соболева СО РАН, г. Новосибирск, Россия. E-mail: **alexander.rybalov@gmail.com**

СНЫТНИКОВА Татьяна Валентиновна — ведущий инженер Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск.
E-mail: **snytav@yahoo.com**

СОШИН Данил Андреевич — сотрудник ФГУП «НИИ «Квант», г. Москва.
E-mail: **danil_re@list.ru**

ФОМИЧЕВ Владимир Михайлович — доктор физико-математических наук, профессор, профессор Финансового университета при Правительстве Российской Федерации, профессор НИЯУ МИФИ, ведущий научный сотрудник ФИЦ ИУ РАН, г. Москва.
E-mail: **fomichev@nm.ru**

Журнал «Прикладная дискретная математика» включен в перечень ВАК рецензируемых российских журналов, в которых должны быть опубликованы основные результаты диссертаций, представляемых на соискание учёной степени кандидата и доктора наук, в базу данных Web of Science (Russian Science Citation Index — RSCI), а также в перечень журналов, рекомендованных УМО в области информационной безопасности РФ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте journals.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также и правила подготовки рукописей статей в журнал.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*