МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

на тему «Применение эвристических алгоритмов к задачам поиска максимально нелинейных векторных булевых функций»

Обучающийся М.Ф. Тагойбекзода
(Инициалы Фамилия) (личная подпись)

Руководитель к.ф.-м.н. О.В. Лелонд
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема бакалаврской работы: «Применение эвристических алгоритмов к задачам поиска максимально нелинейных векторных булевых функций».

Выпускная квалификационная работа посвящена описанию и оценке работы различных эвристических методов нахождения нелинейных сбалансированных векторных функций алгебры логики.

При выполнении бакалаврской работы на языке Java выполнены оценки наибольшей нелинейности.

Выпускная квалификационная работа состоит из введения, трёх разделов, заключения и списка литературы.

В первом разделе описана постановка двух экстремальных задач.

Во втором разделе описываются следующие методы поиска для решения поставленных задач: генетический алгоритм, метод Метрополиса, метод поиска путём постепенного восхождения к вершине.

В третьем разделе описаны результаты применения рассмотренных ранее алгоритмов.

Заключение содержит выводы по выполненной работе.

Бакалаврская работа содержит 6 формул, 11 таблиц. Список использованной литературы включает 20 источников. Объём бакалаврской работы – 47 страниц.

Abstract

Topic of the bachelor's thesis: "Application of heuristic algorithms to the problems of finding maximally non-linear vector Boolean functions."

The final qualifying work is devoted to the description and evaluation of the work of various heuristic methods for finding non-linear balanced vector functions of the algebra of logic.

When performing a bachelor's work in the Java language, estimates of the greatest nonlinearity were made.

The final qualifying work consists of an introduction, three sections, a conclusion and a list of references.

The first section describes the formulation of two extremal problems.

The second section describes the following search methods for solving the tasks: genetic algorithm, Metropolis method, search method by gradual ascent to the top.

The third section describes the results of applying the previously considered algorithms.

The conclusion contains conclusions on the work performed.

Bachelor's work contains 6 formulas, 11 tables. The list of used literature includes 20 sources. The volume of bachelor's work is 47 pages.

Содержание

Введение

Векторные булевы функции выступают основной составляющей при разработке криптографических систем. Для обеспечения криптостойкости векторная булева функция должна иметь некоторые показатели. Ключевыми показателями криптостойкости являются нелинейность и сбалансированность. Функции должны обладать этими показателями каждая в отдельности и совместно. Целью выпускной квалификационной работы является описание и оценка работы различных эвристических методов нахождения нелинейных и сбалансированных векторных функций алгебры логики. Необходимо рассмотреть нахождение таких функций с помощью алгоритма поиска путём постепенного восхождения на вершину, с помощью алгоритма имитации отжига, с помощью генетического алгоритма. Приведённые методы уже применялись для нахождения векторных булевых функций, но не сравнивались между собой.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- определить границы максимального отклонения векторной булевой функции от аппроксимирующей прямой линии;
- выявить векторные функции алгебры логики, у которых отклонение от аппроксимирующей линии близко к ранее определённым границам максимального отклонения;
- выявить векторные функции алгебры логики, обладающие сбалансированностью, с отклонением от аппроксимирующей прямой линии, приближающимся к определённым ранее границам;
- построить алгоритм и выполнить сравнительный анализ методов, способствующих реализации указанных выше задач.

1 Ограничения наибольшей нелинейности (n,m)-функции алгебры логики

1.1 Постановка задачи

Для достижения цели исследования необходимо найти векторные функции алгебры логики, которые имеют высокую нелинейность. Это экстремальная задача с ограничением на сбалансированность.

Другая задача, которая представляет собой экстремальную задачу без такого ограничения, состоит в поиске сбалансированных векторных функций алгебры логики, которые имеют высокую нелинейность. Для постановки этих задач нужно дать следующие определения.

Функция алгебры логики от n аргументов – преобразование $B_n \to B$, в котором $B = \{0,1\}$ представляет собой множество, B_n – двоичный вектор над множеством B, который включает от 0 до 1 вектора n-й длины и действия попарного исключающего «или» и внутреннего произведения.

Исходя из этого, предлагается следующее определение векторной функции алгебры логики или (n, m)-функции, в которой n – число входных аргументов, а m – количество выходных аргументов.

Векторная логическая функция или логическая (n, m)-функция с n параметрами – это множество $F(x) := (f_1(x), f_2(x), \dots, f_m(x))$ с заданным отношением порядка, в которых $x = (x_1, x_2, \dots, x_n)$ и $f_i(x)$ – логические функции, $i = 1, \dots, m$.

Обозначим элементарные логические функции строчными латинскими буквами, а векторные логические функции – заглавными.

Каждое $u \in 0: 2^n - 1$ путём последовательного разделения пополам однозначно представляется следующим образом:

$$u = u_{n-1}2^{n-1} + u_{n-2}2^{n-2} + \ldots + u_1 2 + u_0, \quad (1)$$

где каждый u_i – это единица или ноль.

В связи с этим двоичный n-мерный вектор заменим числом.

К примеру, двоичный n-мерный вектор (0,0,1,0,1) будем заменять его числовым представлением – 5.

Исходя из контекста, можно определить число бит, необходимых для представления числа в двоичной системе счисления.

Количество значений аргументов функции алгебры логики конечно, поэтому всегда есть возможность определить элементарную функцию алгебры логики через вектор скалярных величин размером 2^n, который состоит из единиц и нолей $(f(0), f(1), \ldots, f(2^n - 2), f(2^n - 1))$, и функцию алгебры логики.

В результате будет сформирована прямоугольная таблица $\begin{pmatrix} f_1(0), f_1(1), \ldots, f_1(2^n - 1) \\ f_2(0), f_2(1), \ldots, f_2(2^n - 1) \\ \cdots \\ f_m(0), f_m(1), \ldots, f_m(2^n - 1) \end{pmatrix}$, состоящая из m строк и 2^n столбцов.

Поэтому обозначим пространство логических функций как V_{2^n}, а пространство (n, m)-функций – $V_{2^n}^m$.

В поле B исключающее «или» обозначим символом $\oplus$, произведение p и q – pq.

Попарное исключающее «или» векторов из V_n, образующих пару, тоже обозначим символом $\oplus$.

Для какого пространства используется какое-либо действие, станет понятно на основе контекста.

Следующий термин используется для того, чтобы определить нелинейность логической функции.

Кодовое расстояние $d(f, g)$ – это количество позиций в векторах значений, в которых f и g от n аргументов отличаются.

Нелинейность элементарной функции алгебры логики $f(x_1,\dots,x_n)$ от n аргументов N_f вычисляется как $\min\limits_{(a_0,\dots,a_n)\in V_{n+1}} d(f(x_1,\dots,x_n), a_0 \oplus a_1x_1 \oplus \dots \oplus a_nx_n)$, где d(·,·) – кодовое расстояние.

Распространим определение нелинейности для векторных логических функций.

С этой целью на базе векторной логической функции зададим следующую элементарную логическую.

Допустим, $\theta \in V_m$, $F(x) = (f_1(x),\dots,f_m(x))$ – векторная логическая (n,m)-функция.

Из этого следует, что логическая функция $\theta \cdot F(x) := \theta_1 f_1(x) \oplus \theta_2 f_2(x) \oplus \dots \oplus \theta_m f_m(x)$.

Нелинейность векторной логической (n,m)-функции $F(x) := (f_1(x), f_2(x),\dots,f_m(x))$ от n аргументов представляет собой целое число $N_F := \min\limits_{\theta\in V_m\setminus 0^m}\{N_{\theta\cdot F}\}$, где $\theta \cdot F := \bigoplus\limits_{i=1}^{m} \theta_i f_i$.

На основании введённых понятий можно выполнить постановку экстремальной задачи по нахождению максимума нелинейности векторных логических функций по формуле (2):

$$N_F \to \max_F,\ F \in V_{2^n}^m \tag{2}$$

Кроме прямого метода определения нелинейности элементарной логической функции применяется более удобное для расчётов преобразование Адамара.

Преобразование Адамара функции алгебры логики f от n аргументов представляет собой целое число, являющееся функцией на V_n, которая определяется выражением $W_f(u) := \sum_{x\in V_n}(-1)^{f(x)\oplus\langle x,u\rangle}$, в котором $\langle x,u\rangle = x_1u_1 \oplus \dots \oplus x_nu_n$.

Каждой элементарной функции от n аргументов можно сопоставить вектор, который состоит из коэффициентов Уолша:

$$f \to \left(W_f(0), W_f(1), \dots, W_f(2^n-2), W_f(2^n-1)\right). \quad (3)$$

Этот вектор представляет собой спектр Уолша для функции f.

Для связи преобразования Адамара и нелинейности элементарной функции алгебры логики используется следующее утверждение.

Нелинейность функции алгебры логики N_f рассчитывается по формуле $\frac{1}{2}(2^n - W_{max})$, в которой W_{max} – наибольшее по модулю значение спектра Уолша для функции f [2].

Спектр Уолша векторной логической функции F представляет собой прямоугольную таблицу, состоящую из $2^m - 1$ строк и 2^n столбцов вида $\{W_{\theta\cdot \mathrm{F}}(0), W_{\theta\cdot \mathrm{F}}(1), \dots, W_{\theta\cdot \mathrm{F}}(2^n-1)\}_{\theta \in V_m \backslash 0^m}$. Любая строка этой таблицы представляет собой спектр Уолша элементарной функции $\theta \cdot \mathrm{F}, \theta \in V_m \backslash 0^m$.

Допустим, требуется найти спектр Уолша и нелинейность векторной логической (2,3)-функции F:

$$\begin{array}{cc} & F \\ f_1 & 1,1,0,0 \\ f_2 & 1,1,1,0 \\ f_3 & 1,1,1,1 \end{array} \to \begin{array}{cc} & \theta\cdot \mathrm{F} \\ f_1 & 1,1,0,0 \\ f_2 & 1,1,1,0 \\ f_1 \oplus f_2 & 0,0,1,0 \\ f_3 & 1,1,1,1 \\ f_3 \oplus f_1 & 0,0,1,1 \\ f_3 \oplus f_2 & 0,0,0,1 \\ f_3 \oplus f_2 \oplus f_1 & 1,1,0,1 \end{array} \to \begin{array}{cc} & \text{спектр} \\ & 0,\ 0,\ -4,\ 0 \\ & -2,\ -2,\ -2,\ 2 \\ & 2,\ -2,\ 2,\ 2 \\ & -4,\ 0,\ 0,\ 0 \\ & 0,\ 0,\ 4,\ 0 \\ & 2,\ 2,\ 2,\ -2 \\ & -2,\ 2,\ -2,\ -2 \end{array} \begin{array}{c} N_{\theta\cdot F} \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{array} \quad (4)$$

В соответствии с определением нелинейность векторной логической функции F – это наименьшее значение $N_{\theta\cdot F}$, поэтому $N_F = 0$.

Согласно определению нелинейности и определению спектра Уолша-Адамара (n,m)-функции можно вывести утверждение, которое формулируется следующим образом.

Нелинейность векторной (n, m)-функции алгебры логики N_F рассчитывается по формуле $\frac{1}{2}(2^n - W_{max})$, в которой $W_{max} := max_{u \in V_n, \theta \in V_m \backslash 0_m}(|W_{\theta \cdot F}(u)|)$ – наибольшее по модулю значение спектра Уолша (n, m)-функции алгебры логики F.

Из этого утверждения следует, что задача (2) соответствует задаче минимизации наибольшего по модулю значения элемента спектра Уолша:

$$\max_{u \in V_n, \theta \in V_m \backslash 0_m}(|W_{\theta \cdot F}(u)|) \to \min_F, F \in V_{2^n}^m \quad (5)$$

Рассмотрим определение сбалансированной функции алгебры логики.

Функция алгебры логики f от n аргументов сбалансирована тогда и только тогда, когда вес функции $wt(f) = 2^{n-1}$. $wt(f)$ – число единиц в векторе значений функции алгебры логики f.

С помощью преобразования Адамара определяется, является ли логическая функция сбалансированной.

Функция алгебры логики f обладает сбалансированностью тогда и только тогда, когда $W_f(0) = 0$.

(n, m)-функция алгебры логики обладает сбалансированностью тогда и только тогда, когда функция алгебры логики принимает каждое значение в V_m 2^{n-m} раз.

(n, m)-функция алгебры логики $F(x) = (f_1(x), f_2(x), \dots, f_m(x))$ обладает сбалансированностью тогда и только тогда, когда обладает сбалансированностью любая ненулевая линейная комбинация f_i, т.е. для каждой $\theta \in V_m \backslash 0_m$ функция $\theta \cdot F$ является сбалансированной [6].

Исходя из этого определения, может быть сформулирована экстремальная задача с ограничением на сбалансированность:

$$\max_{u \in V_n, \theta \in V_m \backslash 0_m}(|W_{\theta \cdot F}(u)|) \to \min_F, F \in V_{2^n}^m \quad (6)$$

при ограничении

$$W_{\theta \cdot F}(0) = 0 \text{ для каждого } \theta \in V_m \backslash 0_m. \quad (7)$$

Дадим определения, необходимые для определения границ наибольшей нелинейности векторной (n, m)-функции алгебры логики.

В результате определения этих границ будет представлена таблица для разных n и m.

Если существует векторная (n, m)-функция алгебры логики со значением нелинейности N_F и не существует такой функции G, для которой значение нелинейности N_G было бы больше значения нелинейности N_F, то значение нелинейности N_F представляет собой наибольшее значение нелинейности $N(m, n)$.

1.2 Верхняя оценка наибольшей нелинейности (n,m)-функции

Для определения границ наибольшей нелинейности векторной (n, m)-функции алгебры логики необходимо рассмотреть следующие утверждения.

Утверждение 1. Для нелинейности любой функции алгебры логики, соответственно и для любой (n, m)-функции алгебры логики может быть применена следующая оценка сверху [6]:

$$N(n, m) \leq 2^{n-1} - 2^{\frac{n-2}{2}}, n \geq 2 \quad (8)$$

Имеются различные виды элементарных функций алгебры логики, достигающие наибольшего значения N_F. Одним из таких видов функций являются бент-функции.

Бент-функция представляет собой функцию алгебры логики, у которой количество аргументов n чётно. Бент-функция достигает наибольшего значения $N_F = 2^{n-1} - 2^{\frac{n-2}{2}}$.

Определение бент-функции распространяется на логические (n, m)-функции.

(n, m)-бент-функция алгебры логики $F = (f_1(x), f_2(x), \ldots, f_m(x))$ представляет собой бент-функцию тогда и только тогда, когда все линейные комбинации $f_1, f_2, \ldots, f_m$, отличные от нуля, представляют собой бент-функции.

Отношение (8) в некоторых случаях представляет собой равенство.

Утверждение 2. (n, m)-бент-функция алгебры логики существует тогда и только тогда, когда $n \geq 2m$, при этом n – чётное число, что равносильно $N(n, m) = 2^{n-1} - 2^{\frac{n-2}{2}}$, если $n \geq 2m$, при этом n – чётное число [17].

Представленное далее неравенство (9) [7] сильнее, чем неравенство (8).

Утверждение 3 [7].

$$N(n, m) \leq 2^{n-1} - \frac{\left(3 \cdot 2^n - 2 - 2\frac{(2^n-1)(2^{n-1}-1)}{2^m-1}\right)^{\frac{1}{2}}}{2}, m > n - 2 \qquad (9)$$

Значение правой части неравенства (9) может не являться целым числом.

В связи с этим в качестве оценки сверху значение правой части неравенства (9) можно округлять в меньшую сторону.

Утверждение 4 [7, 16]. $N(n, m) = 2^{n-1} - 2^{\frac{n-1}{2}}$, если n нечётно и $n = m$.

Утверждение 5 [19]. $N(n, m) = 2^{n-1} - 2^{\frac{n-1}{2}}$, если $n = 3, 5, 7$ и $n > m$.

Для верхней и нижней оценок наибольшей нелинейности применяется кодирование с помощью двоичных блочных кодов.

Линейный блочный (n, k)-код – это подмножество C линейного пространства V_n размером k.

В этом случае кодовыми словами являются векторы подмножества C, длиной кодового слова служит n, размером пространства, которое охватывает

все возможные слова, выступает k, а наименьшим расстоянием между словами будет являться расстояние Хэмминга $D(C)$.

(n, k, D)-код – это линейный двоичный блочный (n, k)-код, который обладает следующими характеристиками: n – длина кодового слова, k – размером пространства, которое охватывает все возможные слова, и D – расстояние Хэмминга.

Утверждение 6. Если ни при каком расстоянии Хэмминга D, большим или равном нелинейности N_F, линейного двоичного блочного $(2^n, n + 1 + m, D)$-кода не существует, тогда также не будет существовать векторная (n, m)-функция алгебры логики с нелинейностью N_F [20].

Для использования теоремы 1 необходимо знать значения, при которых линейный двоичный блочный код не существует.

Утверждение 7. Для любого (n, k, D)-кода с наименьшим расстоянием Хэмминга D верно, что $n - k \geq \log_2 \left(\sum_{i=0}^{\lfloor (D-1)/2 \rfloor} C_n^i \right)$ [8].

Из утверждений 6 и 7 вытекает следствие 1.

Следствие 1. Если верно, что $2^n - (n + 1 + m) < \log_2 \left(\sum_{i=0}^{\lfloor (D-1)/2 \rfloor} C_{2^n}^i \right)$, то не существует векторной (n, m)-функции алгебры логики с нелинейностью $N_F = D$.

Далее требуется ввести понятие самомодифицирующегося двоичного кода.

Двоичный код является самомодифицирующимся тогда и только тогда, когда из того, что в нём имеется вектор v, следует, что в нём имеется и вектор $1^n - v$.

Так, если код является самомодифицирующимся и в нём имеется вектор (1,0,0,0), то в нём имеется и вектор (0,1,1,1).

Утверждение 8. Если линейного самомодифицирующегося двоичного $(2^n, n + 1 + m, D)$-кода, в котором $D \geq N_F$, не существует, то не существует

векторной (n, m)-функции алгебры логики со значением нелинейности N_F [20].

Далее требуется дать некоторые определения. A_i – число слов в двоичном коде C с весом i, многочлен Кравчука будет вычисляться по формуле $P_k(i) := \sum_{0 \le j \le k} (-1)^j C_i^j C_{l-i}^{k-j}$, в которой C_n^k представляет собой обобщение бинома Ньютона и вычисляется по формуле $\frac{n(n-1)(n-2)\cdots(n-(k-1))}{k!}$. В последних формулах $n \in Z, k, i \in N \cup 0$.

Утверждение 9. Не существует линейного самомодифицирующегося двоичного $(l = 2^n, \lceil \log_2(2 + S_{max}) \rceil, D)$-кода, в котором S_{max} представляет собой наибольшее значение линейной функции многих переменных для задачи линейного программирования по поиску максимума [20]:

$$2\left(A_D + A_{D+1} + \cdots + A_{\frac{l}{2}-1}\right) + A_{\frac{l}{2}} \tag{10}$$

при линейных ограничениях в виде

$$\sum_{D \le i \le \frac{l}{2}-1} A_i\left(P_k(i) + P_k(l-i)\right) + A_{\frac{l}{2}} \ge -2C_l^k \tag{11}$$

для чётных значений k, при которых выполняются условия $2 \le k \le l$,

$$A_i \ge 0, D \le i \le \frac{l}{2} \tag{12}$$

Необходимо отметить следующее свойство (n, m)-функции алгебры логики.

Замечание 1. Если $m > 1$, то верно следующее: $N(m, n) \le N(m-1, n)$.

1.3 Нижние границы наибольшей нелинейности (n,m)-функции алгебры логики

Patterson и Wiedemann [19] доказали следующее.

Для $m = 1, \quad n \geq 15$ справедливо неравенство: $N(n, m) > 2^{n-1} - 2^{(n-1)/2}$.

Утверждение 10 [18]. Если n – чётное число, для которого выполняются неравенства $1 < n < 2m$ и $m \leq n$, то $N(n, m) \geq 2^{n-1} - 2^{n/2}$.

Каждая функция алгебры логики задаётся многочленом Жегалкина (algebraic normal form, ANF).

Многочлен Жегалкина – это способ записи функции алгебры логики с помощью многочлена с коэффициентом единица либо нуль. В этом многочлене вместо умножения используется логическое умножение, а вместо суммы – исключающее «или».

Алгебраической степенью функции алгебры логики является алгебраическая степень многочлена Жегалкина.

Элементы множества векторных (n, m)-функций алгебры логики от n аргументов, у которых алгебраические степени меньше или равны r, формируют линейный двоичный блочный код, называемый кодом Рида-Мюллера порядка r с длиной кодового слова 2^n, $RM_{\mathrm{r,n}}$.

При этом двоичный код Рида-Мюллера $RM_{\mathrm{r\prime,n}}$ содержит двоичный код Рида-Мюллера $RM_{\mathrm{r,n}}$, порядок r' больше, чем порядок r, или по-другому $RM_{\mathrm{r,n}}$ представляет собой подкод для $RM_{\mathrm{r\prime,n}}$.

Утверждение 11. Если двоичный код Рида-Мюллера порядка 1 $RM_{1,\mathrm{n}}$ представляет собой подкод для линейного двоичного $(2^n, K, D)$-код, причём $1 \leq m \leq K - n - 1$, то имеется векторная функция алгебры логики, у которой значение нелинейности больше или равно расстоянию Хэмминга D [20].

Двоичный код Рида-Мюллера $RM_{\mathrm{r,n}}$ характеризуется длиной кодового слова 2^n, размером пространства кодовых слов $S = 1 + C_n^1 + \ldots + C_n^r$ и наименьшим расстоянием Хэмминга 2^{n-r} [4]. Из перечисленного вытекает следствие 2.

Следствие 2. Если $1 \le m \le C_n^1 + \ldots + C_n^r - n$, то существует векторная (n, m)-функция алгебры логики с нелинейностью, которая больше или равна 2^{n-r}.

Утверждение 12. Ортогональный двоичный код к расширенному двоичному коду Боуза-Чоудхури-Хоквингема $exBCH_{2t+1,n}^{\perp}$ представляет собой надкод для двоичного кода Рида-Мюллера $RM_{1,\mathrm{n}}$ и характеризуется длиной слова 2^n, размером пространства слов $K = tn + 1$, наименьшим D, которое больше или равно $2^{n-1} - (t-1)2^{n/2}$, если $t > 1$ и $2t - 1 < 2^{\lceil n/2 \rceil} + 1$ [20].

Следствие 3. Если $1 \le m \le (t-1)n$, $t > 1$ и $2t - 1 < 2^{\lceil n/2 \rceil} + 1$, то можно построить векторную функцию алгебры логики со значением нелинейности $D \ge 2^{n-1} - (t-1)2^{n/2}$.

1.4 Таблица границ наибольшей нелинейности векторной (n,m)-функции алгебры логики для разных n и m

Исходя из представленных в параграфах 1.2 и 1.3 замечаний, следствий и иных утверждений, заполним таблицу границ наибольшей нелинейности векторной (n, m)-функции алгебры логики для различных n и m (таблица 1).

Значения в ячейках обозначают границы наибольшей нелинейности векторной (n, m)-функции алгебры логики снизу и сверху.

Если ячейка таблицы содержит только одно значение, границы снизу и сверху одинаковые.

Все границы ссылаются на замечание, следствие или иное утверждение $\alpha, \beta, \ldots, \mu, \nu$, сопутствующее получению этих границ.

Аналогичная таблице 1 таблица для значений n от 3 до 8, m от 1 до 8 представлена в [20].

В таблице 1 добавлен столбец $m = 9$ и строка $n = 9$.

Таблицу 1 можно расширить для других m и n.

Таблица 1 – Наибольшая нелинейность $N(m,n)$

n \ m	1	2	3	4	5	6	7	8	9
3	2^{ϵ}	2^{ϵ}	$2^{\kappa,\alpha}$	$1^{\kappa,\gamma}$	-	-	-	-	-
4	6^{β}	6^{β}	$4^{\kappa} - 5^{\alpha}$	$4^{\kappa,\zeta}$	$4^{\kappa,\zeta}$	$4^{\kappa,\zeta}$	$2^{\kappa,\zeta}$	$2^{\kappa,\zeta}$	$2^{\kappa,\zeta}$
5	12^{ϵ}	12^{ϵ}	12^{ϵ}	12^{ϵ}	12^{δ}	$8^{\kappa} - 10^{\zeta}$	$8^{\kappa} - 10^{\zeta}$	$8^{\kappa} - 9^{\zeta}$	$8^{\kappa} - 9^{\zeta}$
6	28^{β}			$24^{\iota} - 28^{\alpha}$	$24^{\iota} - 26^{\zeta}$	$24^{\lambda} - 26^{\gamma}$	$24^{\mu} - 25^{\gamma}$	$24^{\mu,\gamma}$	$24^{\mu,\gamma}$
7	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{δ}	$48^{\mu} - 54^{\eta}$	$48^{\mu} - 54^{\eta}$
8	120^{β}				$112^{\lambda} - 120^{\alpha}$			$112^{\lambda} - 116^{\gamma}$	$96^{\lambda} - 115^{\gamma}$
9	$242^{\xi} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	$240^{\nu} - 244^{\alpha}$	240^{δ}

α: Утверждение 1 [6].

β: Утверждение 2 [17].

γ: Утверждение 3 [7] (ограничение Chabaud-Vaudenay).

δ: Утверждение 4 [7, 16].

ϵ: Утверждение 5 [19].

ζ: Утверждение 6 [20] (ограничение линейного кодирования).

η: Утверждения 8, 9 [20] (ограничение линейной оптимизации).

ι: Утверждение 10 [18].

κ: Утверждение 11 (ограничение надкода), Следствие 2 [20] (коды Рида-Мюллера).

λ: Утверждение 11 (ограничение надкода), Утверждение 12, Следствие 3 (коды Боуза-Чоудхури-Хоквингема).

μ: в [20] использована дополнительная информация о расстоянии Хэмминга кодов Боуза-Чоудхури-Хоквингема.

ν: Замечание 1.

ξ: получено методом эвристики для некоторых особых типов векторных функций алгебры логики [9, 10].

Вычисления согласно теореме 3 производились в среде Matlab. Остальные рассмотренные оценки наибольшей нелинейности выполнялись на языке программирования Java.

В первом разделе описана постановка двух экстремальных задач: задачи (5) и задачи (6)-(7). Задача (6)-(7) может быть определена как задача с ограничением на сбалансированность. Задача (5) может быть определена как задача без ограничений. Для решения экстремальных задач с учётом и без учёта ограничений (5)-(7) будут использоваться алгоритмы эвристики.

2 Применяемые эвристические методы

Для решения экстремальной задачи без учёта сбалансированности (5) и экстремальной задачи с ограничением на сбалансированность (6)-(7) будут использованы следующие методы поиска: генетический алгоритм, метод Метрополиса, метод поиска путём постепенного восхождения к вершине. В связи с тем, что каждый из этих методов имеет различия в применении к поставленным задачам, представлены два вида алгоритмов – с учётом и без учёта ограничений.

2.1 Метод Метрополиса

Метод Метрополиса – это эвристический алгоритм нахождения оптимального значения. Метод основан на моделировании процессов, происходящих, когда атомы образовали кристаллическую решётку, их переходы в другие ячейки ещё допустимы. При этом температура постепенно снижается. Достоинство метода Метрополиса заключается в наличии возможности избегать ловушки локальных экстремумов функции, которая подвергается оптимизации, и продолжить искать глобальный экстремум функции, что удаётся путём принятия изменений, которые уменьшают или увеличивают значение требующей оптимизации функции в зависимости от температуры кристаллизации T. С повышением значения T увеличивается вероятность ухудшения [3]. В [8] показан вариант использования метода Метрополиса к экстремальным задачам с учётом и без учёта ограничений на сбалансированность. Характеристиками моделируемого процесса являются показатель снижения параметра T α, число запретов b, число изменяемых на каждой итерации координат k, число итераций на каждом уровне параметра T it.

2.1.1 Метод Метрополиса для несбалансированной задачи

Для инициализации задаётся функция изменения параметра T $T(t)$. Для этого используется формула $T(t) = \alpha \cdot t$, в которой в качестве показателя может использоваться либо 0 либо 1. Наименьшее значение параметра T t_{min} и исходное значение параметра T t_{max} определяются таким образом, чтобы алгоритм в случае роста значения нелинейности до максимального значения перешёл к функции с меньшим значением нелинейности с вероятностями соответственно 10^{-5} и $0{,}95$. Случайным образом выберем или определим векторную (n, m)-функцию алгебры логики F_{glob}, определим такую векторную (n, m)-функцию алгебры логики F, чтобы она была равной значению F_{glob}. Определим такую температуру t, чтобы она была равной t_{max}. Пока фиксированный уровень параметра T t больше t_{min}, для каждого t произведём 3 операции it раз.

Сгенерируем случайное число изменяемых на каждой итерации координат k в прямоугольной таблице векторной (n, m)-функции алгебры логики F. Возможно повторное изменение изменённой ранее координаты.

Если значение W_{max} в прямоугольной таблице спектра Уолша уменьшилось или возросла нелинейность векторной (n, m)-функции алгебры логики F, оставим это изменение в векторной (n, m)-функции алгебры логики F.

Иначе оставим изменения с вероятностью $P(\Delta_W)$, которая вычисляется по формуле $e^{(-\frac{\Delta_{WH}}{2}-1)/t_i}$, в которой $\Delta_W = W'_{max} - W_{max}$, W'_{max} – наибольшее по модулю значение в прямоугольной таблице спектра Уолша изменённой функции, а W_{max} – наибольшее по модулю значение в прямоугольной таблице спектра Уолша для исходной функции.

Если нелинейность векторной (n, m)-функции алгебры логики F стало больше нелинейности векторной (n, m)-функции алгебры логики F_{glob},

зададим значение F_{glob}, равное F. Снизим температуру $t = T(t)$. Последние действия выполняем до тех пор, пока $t > t_{min}$.

2.1.2 Метод Метрополиса для задачи с учётом ограничения на сбалансированность

Для сбалансированной задачи определим векторную (n, m)-функцию алгебры логики F_{glob} таким образом, чтобы она обладала сбалансированностью. Будем менять местами столбцы прямоугольной таблицы векторной (n, m)-функции алгебры логики каждый раз вместо замены её координат. Других изменений нет.

2.1.3 Версии метода Метрополиса

Для совершенствования метода Метрополиса можно добавить табу-поиск. В этом случае если при поиске функций, обладающих сбалансированностью, изменился какой-либо столбец или какая-либо координата, необходимо запретить менять соответственно столбец или координату в течение b итераций, где b – число запретов.

2.2 Поиск восхождением к вершине

Поиск восхождением к вершине является алгоритмом локального поиска. Поиск путём постепенного восхождения к вершине представляет собой итеративный алгоритм. Поиск начинается с решения, найденного случайным образом. Потом выполняется поиск лучшего решения в заданной окрестности решения, найденного случайным образом. Когда оно найдено, метод переходит к нему. Это повторяется до тех пор, пока не будет выявлено ни одного лучшего решения.

В [11, 15] продемонстрирован пример использования поиска путём постепенного восхождения к вершине для нахождения функций алгебры логики с определёнными характеристиками, например обладающих высокой

нелинейностью. k-окрестность векторной (n, m)-функции алгебры логики F – это множество векторных (n, m)-функций алгебры логики, прямоугольные таблицы которых различаются не более чем на k позиций от прямоугольной таблицы векторной (n, m)-функции алгебры логики F. Тогда k-сбалансированная окрестность векторной (n, m)-функции алгебры логики F – это множество векторных (n, m)-функций алгебры логики, полученных из векторной (n, m)-функции алгебры логики F путём перестановки не более k столбцов прямоугольной таблицы функции F. Управляемой характеристикой является параметр k, который задаёт окрестность функции.

2.2.1 Метод поиска путём постепенного восхождения к вершине для задачи без учёта ограничения на сбалансированность

Сгенерируем случайным образом или определим векторную (n, m)-функцию алгебры логики F.

Рассмотрим векторные (n, m)-функции алгебры логики из k-окрестности функции F.

Если обнаружена векторная (n, m)-функция алгебры логики, значение нелинейности которой выше, чем у векторной (n, m)-функции алгебры логики F, или значения их нелинейности одинаковы, но при этом в прямоугольной таблице спектра Уолша обнаруженной векторной (n, m)-функции алгебры логики число наибольших по модулю значений W_{max} меньше, возьмём эту обнаруженную векторную (n, m)-функцию алгебры логики за новую векторную (n, m)-функцию алгебры логики F. Возвращаемся ко второму шагу.

Поиск завершается, если рассмотрены все векторные (n, m)-функции алгебры логики из k-окрестности, при этом ни в одном случае не найдено лучшее решение.

2.2.2 Метод поиска путём постепенного восхождения к вершине для задачи с учётом ограничения на сбалансированность

Для сбалансированной задачи необходимо заменить два первых шага алгоритма:

– сгенерируем случайным образом или определим сбалансированную векторную (n, m)-функцию алгебры логики F;
– рассмотрим векторные (n, m)-функции алгебры логики из k-сбалансированной окрестности функции F.

В связи с тем, что любая векторная (n, m)-функция алгебры логики из k-сбалансированной окрестности сбалансированной векторной логической (n, m)-функции обладает сбалансированностью, результатом алгоритма будет являться векторная (n, m)-функция алгебры логики, обладающая сбалансированностью.

2.3 Генетический алгоритм

Генетический алгоритм включает взаимодействующие между собой этапы комбинирования, мутирования и селекции. Сначала случайным образом генерируется множество исходной популяции, к элементам которого применяют комбинирование и мутирование. Таким образом формируется множество новых решений. Далее в следующее поколение отбирают лучшие решения с позиции оптимизируемой функции. Указанные шаги повторяют.

Для несбалансированной задачи использован реализованный в среде MatLAB Genetic Algorithm. На его основе были реализованы рассматриваемые далее вариации поиска.

2.3.1 Генетический алгоритм для несбалансированной задачи

Для управления используются такие характеристики как доля мутаций в поколении α, доля селекции с предоставлением возможности повторения

выбираемых функций β (опционально), количество особей N, число итераций it, число ячеек двумерного массива, которые меняются у мутантов, k.

Для улучшения функций из множества решений в рассматриваемом алгоритме набор векторных функций алгебры логики меняется путём комбинирования, мутации и селекции. В генетических алгоритмах целевую функцию обычно называют функцией приспособленности. В качестве функции приспособленности примем нелинейность векторной (n, m)-функции алгебры логики.

Зададим случайным образом исходное множество родителей – N векторных (n, m)-функций алгебры логики.

Комбинирование. Обозначим матрицу функций ребёнка – b_c, а матрицы функций родителей b_1, b_2. Будем перебирать элементы матрицы функции (гены). Если у родителей одинаковые гены, т.е. значение ячейки матрицы (n, m)-функций родителя b_1 равно значению соответствующей ячейки матрицы (n, m)-функций родителя b_2, то соответствующая ячейка матрицы функций ребёнка примет это значение, то есть $b_c[i][j] = b_1[i][j] = b_2[i][j]$. Если $b_1[i][j] \neq b_2[i][j]$, то соответствующая ячейка матрицы (n, m)-функций ребёнка $b_c[i][j]$ выбирается случайным образом между 0 и 1. При этом вероятность распределяется равномерно. Создадим $\frac{N(N-1)}{2}$ потомков путём комбинирования всех допустимых вариантов. После добавления (n, m)-функций родителей мощность множества потомков примет значение $\frac{N(N+1)}{2}$.

Мутирование. Отберём $\alpha \cdot \frac{N(N+1)}{2}$ элементов из множества потомков. У каждой отобранной векторной функции алгебры логики отберём случайным образом k элементов матрицы и поменяем их.

Выберем из множества потомков N векторные функции алгебры логики с максимальным значением нелинейности и примем полученные векторные (n, m)-функции алгебры логики за новое множество родителей. Если значения

нелинейности (n, m)-функций окажутся одинаковыми, необходимо сначала выбрать элементы с минимальным числом наибольших значений спектра Уолша.

Повторим комбинирование, мутирование, выбор $it - 1$ раз.

2.3.2 Генетический алгоритм для экстремальной задачи с учётом ограничения на сбалансированность

Версия алгоритма для экстремальной задачи для задачи с учётом ограничения на сбалансированность рассматривается в [12] и используется с небольшими изменениями. Для управления используется такие характеристики как величина популяции N и число итераций it.

Создание исходного множества векторных функций алгебры логики. Задаём одномерный массив, который включает 2^{n-m} копий всех возможных 2^m векторов размером m. Случайным образом меняя местами элементы массива, сформируем N случайных векторных функций алгебры логики, обладающих сбалансированностью.

Комбинирование-мутации. Пусть $b_1(x)$, $b_2(x)$ – матрицы родительских функций, а $b_c(x)$ – матрица функций ребёнка. Тогда $b_i(x_j)$ – столбец матрицы функций родителя или ребёнка, а $\#(b_i(x_j))$ – число появлений $b_i(x_j)$, где $i = 1, 2$ в $b_c(x)$. Рассмотрим комбинирование-мутации функций $b_1(x)$, $b_2(x)$. Когда $b_1(x_j) = b_2(x_j)$:

- в случае, если при этом $\#(b_1(x_j)) < 2^{n-m}$, тогда $b_c(x_j) = b_1(x_j)$, инкрементируем $\#(b_1(x_j))$;
- иначе, если $b_c(x_j) = b_1(x_k), k \neq j$, при этом $\#(b_1(x_k))$ – минимальное. Инкрементируем $\#(b_1(x_k))$.

Когда $b_1(x_j) \neq b_2(x_j)$: в случае если при этом $\#(b_1(x_j)) = 2^{n-m}$, а $\#(b_2(x_j)) = 2^{n-m}$, тогда $b_c(x_j) = b_1(x_k), k \neq j$, при этом $\#(b_1(x_k))$ – минимальное, инкрементируем $\#(b_1(x_k))$; в случае, если при этом $\#(b_1(x_j)) <$

$\#(b_2(x_j))$, тогда $b_c(x_j) = b_1(x_j)$ и инкрементируем $\#(b_1(x_j))$; в случае, если при этом $\#(b_1(x_j)) > \#(b_2(x_j))$, тогда $b_c(x_j) = b_2(x_j)$ и инкрементируем $\#(b_2(x_j))$; в случае, когда $\#(b_1(x_j)) = \#(b_2(x_j))$ и при этом $\#(b_i(x_j)) < 2^{n-m}$, тогда задаём $b_c(x_j)$ случайным образом равным либо $b_1(x_j)$ либо $b_2(x_j)$. Инкрементируем соответственно либо $b_1(x_j)$ либо $b_2(x_j)$. Этот алгоритм содержит одновременно комбинирование и мутирование. Как правило, либо $b_c(x_j) = b_1(x_j)$ либо $b_c(x_j) = b_2(x_j)$. Но в некоторых случаях $b_c(x_j)$ не принимает значение ни $b_1(x_j)$ ни $b_2(x_j)$. Тогда говорят, что $b_c(x_j)$ – мутирование. Генетический алгоритм предоставляет гарантии, что потомок будет обладать сбалансированностью. Комбинируя все возможные пары, получим $\frac{N(N-1)}{2}$ потомков. Добавим функции родителей во множество потомков, при этом мощность множества потомков примет значение $\frac{N(N+1)}{2}$.

Выберем из полученного множества N векторные функции алгебры логики, которые обладают максимальной нелинейностью (их может быть несколько) и сформируем из выбранных (n, m)-функций новое множество родителей. Если нелинейности совпали, сначала выберем элементы с минимальным числом наибольших значений спектра Уолша.

Повторим комбинирование-мутации и выбор до тех пор, пока число итераций не достигнет значения it.

2.3.3 Вариации генетического алгоритма

Для совершенствования генетического алгоритма может быть применено «встряхивание», заключающееся в том, что если нелинейность не улучшается во время s итераций, $N - 2$ родительских функций определяются заново. При этом две родительские функции, обладающие максимальным значением нелинейности, остаются.

Другим вариантом совершенствования генетического алгоритма может быть улучшение селекции функций. Вместо селекции, не учитывающей

повторения выбираемых функций, выбираются $\lfloor \beta \cdot N \rfloor \, \beta \epsilon [0,\ 1]$ векторных функций алгебры логики, которые могут повторяться, при этом остальные $\lceil (1 - \beta) \cdot N \rceil$ векторные функций алгебры логики выбираются таким образом, чтобы они не повторялись. Если во время селекции особи закончатся, породим случайным образом потомков, которых не хватает. Если доля отбора $\beta = 0$, то все выбранные элементы будут отличаться, если доля отбора $\beta = 1$, селекция будет проведены тем же образом, что и без совершенствования.

2.4 Совершенствование используемых эвристических методов

Каждая элементарная функция алгебры логики от n аргументов может быть задана как 0-1 вектором длины 2^n, так и 0-1 вектором длины 2^n, в котором все элементы являются коэффициентами при заданном одночлене многочлена Жегалкина (алгебраическая нормальная форма). Чтобы перейти от вектора функции алгебры логики к алгебраической нормальной форме можно воспользоваться преобразованием Мёбиуса [1]. Отсюда вытекает, что каждую (n, m)-функцию алгебры логики $F = \left(f_1(x), \dots, f_m(x)\right)$ можно задать прямоугольной таблицей алгебраической нормальной формы, в которой каждая строка с номером i представляет собой вектор представления функции алгебры логики f_i в виде алгебраической нормальной формы. В случае применения рассмотренных методов к представленной таким образом функции можно описать новые модификации этих методов.

3 Результаты работы алгоритмов

Результаты применения рассмотренного в параграфе 2.4 совершенствования используемых методов оказались хуже по сравнению с оригинальными вариантами методов для всех методов. Использование «встряхивания» в генетическом алгоритме и табу-поиска в методе Метрополиса способно как улучшить значение нелинейности, полученное в результате применения оригинальных методов, так и сделать его хуже. Указанные модификации использоваться не будут.

Совершенствование селекции функций в генетическом алгоритме при доле селекции β, которая принадлежит промежутку [0.8, 0.9], при поиске функций без учёта сбалансированности в большинстве случаев улучшает нелинейность. Совершенствование селекции функций при доле селекции β, которая чуть больше нуля, при поиске функций с учётом сбалансированности также в большинстве случаев улучшает нелинейность. Это совершенствование алгоритма будет использовано для получения лучших результатов.

Если сравнивать генетический алгоритм, реализованный в среде Matlab, с рассмотренными в предыдущем разделе методами, то он значительно ухудшает нелинейность. Так, при $m = 2,\ n = 9,\ N = 100$ генетический алгоритм достигает нелинейность, равную 228, при этом рассмотренные методы дают нелинейность не ниже 232 (таблица 2). Этот генетический алгоритм использоваться не будет.

Проведём сравнительный анализ рассмотренных методов при различных m и n по нижеперечисленным характеристикам:

– max_N_F – наибольшее полученное значение нелинейности;
– $num\_max\_N_F$ – количество функций, для которых получено это наибольшее значение;

– EN_F – среднее значение нелинейности;
– $Etime$ – среднее значение времени выполнения алгоритма. Для реализации алгоритмов использовался одинаковый язык и одна и та же вычислительная машина.

Таблица 2 – Результат выполнения 100 тестов алгоритмов при $m = 2, n = 9$

Методы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	232	75	231.7	52.55
Метод поиска путём постепенного восхождения к вершине	234	59	233.22	119.78
Генетический алгоритм	234	3	232.11	81.22
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	234	77	233.73	152
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	236	1	233.62	193.27
Случайная генерация	227	4	226.04	100

Обычно число популяций выбирается равным 100.

Если воздействующие характеристики (количество особей в генетическом алгоритме, число итераций, k-окрестности в методе поиска путём постепенного восхождения к вершине) увеличиваются, характеристики max_N_F, $num\_max\_N_F$, EN_F изменяется в сторону улучшения, при этом параметр $Etime$ может значительно увеличиться. Учитывая формулировки

задач, основным показателем является наибольшее и среднее значения нелинейности, при этом значение параметра $Etime$ должно быть адекватным.

Сравним работу рассматриваемых алгоритмов с поиском функции случайной генерацией. Зададим функцию случайной генерацией и, если нелинейность заданной функции больше, чем нелинейность функции F, примем заданную функцию за искомую функцию F. Для управления поиском случайной генерацией используется только характеристика $Etime$.

Рассмотрим сочетания представленных выше методов:

– метод Метрополиса + метод поиска путём постепенного восхождения к вершине – для найденной методом Метрополиса функции используют метод поиска путём постепенного восхождения к вершине;
– генетический алгоритм + метод поиска путём постепенного восхождения к вершине – для найденной генетическим алгоритмом функции используют метод поиска путём постепенного восхождения к вершине.

3.1 Результат работы алгоритмов для несбалансированной задачи

Чтобы уточнить ограничения наибольшей нелинейности, нужно брать значения m и n, при которых ограничения наибольшей нелинейности снизу и сверху различаются. Если рассматриваемый алгоритм даст функцию алгебры логики, значение нелинейности которой превышает её ограничение снизу, то можно утверждать, что нижнее ограничение значения наибольшей нелинейности улучшено.

Проведём сравнительный анализ выполнения алгоритмов для экстремальной несбалансированной задачи при $(m = 6, n = 5)$, $(m = 3, n = 4)$, $(m = 4, n = 6)$ и $(m = 5, n = 8)$.

В таблице 3 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 5$ и $m = 6$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.9$, $it = 1000$, $k = 1$;
– метод поиска путём постепенного восхождения к вершине: $k = 1$;
– генетический алгоритм: количество особей $N = 50$, число повторений $it = 100$, $\alpha = 0.4$, $k = 2$, $\beta = 0.9$.

Таблица 3 – Результат выполнения 100 тестов алгоритмов при $m = 6$ и $n = 5$

Методы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	9	93	8.93	8.74
Метод поиска путём постепенного восхождения к вершине	10	1	8.98	5.18
Генетический алгоритм	10	1	8.95	21.54
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	10	2	9.02	12.59
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	10	4	9.04	26.26
Случайная генерация	9	1	8.01	10

При $m = 6$ и $n = 5$ наибольшая нелинейность $N(5, 6) = 8 - 10$ (таблица 1). Из таблицы 3 видно, что в результате работы каждого алгоритма была найдена функция с более высоким значением нелинейности, поэтому можно

утверждать, что нижнее ограничение наибольшей нелинейности улучшено. Большинство алгоритмов сработали таким образом, что значение нелинейности приняло значение верхнего ограничения наибольшей нелинейности. Таким образом, было выполнено уточнение нижнего ограничения наибольшей нелинейности. $N(5,6) = 10$. Все рассмотренные алгоритмы находят лучшее значение по сравнению с поиском случайной генерацией.

В таблице 4 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 4$ и $m = 3$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.95$, $it = 1000$, $k = 1$;
– метод поиска путём постепенного восхождения к вершине: $k = 3$;
– генетический алгоритм: количество особей $N = 100$, число повторений $it = 100$, $\alpha = 0.4$, $k = 2$, $\beta = 0.8$.

Таблица 4 – Результат выполнения 100 тестов алгоритмов при $m = 3$ и $n = 4$

Методы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	4	100	4	0.78
Метод поиска путём постепенного восхождения к вершине	4	100	4	0.09
Генетический алгоритм	4	100	4	4.5
Метод Метрополиса + восхождение к вершине	4	100	4	0.86

Продолжение таблицы 4

Методы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	4	100	4	4.6
Случайная генерация	4	100	4	4

При $m = 3$ и $n = 4$ наибольшее значение нелинейности $N(4, 3) = 4 - 5$ (таблица 1). Улучшение ограничения наибольшей нелинейности отсутствует. Из таблицы 4 следует, что все рассмотренные алгоритмы дают такой же результат, как и поиск случайной генерацией.

В таблице 5 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 6$ и $m = 4$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.95$, $it = 2100$, $k = 1$;
– метод поиска путём постепенного восхождения к вершине: $k = 2$;
– генетический алгоритм: количество особей $N = 100$, число повторений $it = 100$, $\alpha = 0.4$, $k = 2$, $\beta = 0.8$.

Таблица 5 – Результат выполнения 100 тестов алгоритмов при $m = 4$ и $n = 6$

Методы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	24	70	23.7	9.73
Метод поиска путём постепенного восхождения к вершине	24	86	23.84	6.51
Генетический алгоритм	24	85	23.85	41.8

Продолжение таблицы 5

Методы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	24	97	23.97	15.84
Генетический алгоритм+ метод поиска путём постепенного восхождения к вершине	24	98	23.98	47.09
Случайная генерация	23	1	22.01	15

При $m = 4$ и $n = 6$ значение наибольшей нелинейности $N(6,4) = 24 - 28$ (таблица 1). Как видно из таблицы 5, улучшение ограничения наибольшей нелинейности отсутствует, но каждый рассмотренный алгоритм достиг ограничения снизу нижней границы и показал хорошие результаты по сравнению с поиском случайной генерацией.

В таблице 6 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 8$ и $m = 5$ с нижеперечисленными характеристиками:

– метод поиска путём постепенного восхождения к вершине: $k = 2$;

– метод Метрополиса: $\alpha = 0.95$, $it = 3000$, $k = 1$;

– генетический алгоритм: $\alpha = 0.4$, $\beta = 0.9$, $it = 100$, $k = 1$, $N = 100$.

Количество тестов уменьшено до 20 в связи с долгой работой алгоритмов.

Таблица 6 – Результат выполнения 20 тестов алгоритмов при $n = 8, m = 5$

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	107	9	106.45	256.95
Метод поиска путём постепенного восхождения к вершине	108	15	107.65	1615.02
Генетический алгоритм	108	1	105.5	384.51
Имитация отжига + метод поиска путём постепенного восхождения к вершине	108	14	107.65	1485.17
Генетический + метод поиска путём постепенного восхождения к вершине	108	15	107.7	1900.58
Случайная генерация	103	20	103	250

Для $n = 8, m = 5$ наибольшее значение нелинейности $N(8,5) = 112 - 120$ (таблица 1). В соответствии с таблицей 6 ни один из алгоритмов не достиг нижней границы максимальной нелинейности. Все алгоритмы работают лучше случайной генерации. Среди чистых алгоритмов выделяется метод поиска путём постепенного восхождения к вершине, который при заданных характеристиках выполняет поиск функции лучше, чем метод Метрополиса и генетический алгоритм, и не хуже, чем комбинации алгоритмов. Однако время работы метода поиска путём постепенного восхождения к вершине в числе максимальных.

В таблице 7 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 9$ и $m = 2$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.95$, $it = 3000$, $k = 1$;
– метод поиска путём постепенного восхождения к вершине: $k = 2$;
– генетический алгоритм: $N = 100$, $it = 100$, $\alpha = 0.4$, $k = 1$, $\beta = 0.9$.

Таблица 7 – Результат работы 100 тестов алгоритмов при $n = 9, m = 2$

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	232	75	231.7	52.55
Метод поиска путём постепенного восхождения к вершине	234	59	233.22	119.78
Генетический алгоритм	234	3	232.11	81.22
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	234	77	233.73	152
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	236	1	233.62	193.27
Случайная генерация	227	4	226.04	100

Для $n = 9, m = 2$ значение наибольшей нелинейности $N(9, 2) = 240 - 244$ (таблица 1). Все алгоритмы работают лучше, чем случайная генерация. Среди чистых алгоритмов положительно выделяется метод поиска путём постепенного восхождения к вершине. Лучшую статистику, если не учитывать время работы, демонстрируют комбинированные алгоритмы (таблица 7).

При $n = 8, m = 9$ исследуемыми алгоритмами удалось получить функцию с нелинейностью 100, тогда как теоретическая максимальная нелинейность $N(8, 9) = 96115$.

Почти во всех случаях исследуемые алгоритмы работают лучше, чем случайная генерация. Комбинации алгоритмов обладают лучшей статистикой по функциям, но работают дольше остальных. Среди чистых алгоритмов лучше других ищет функции метод поиска путём постепенного восхождения к вершине.

3.2 Результат работы алгоритмов для сбалансированной задачи

Для сбалансированной векторной булевой (n, m)-функции должно выполняться $n \geq m$. Сравним работу алгоритмов для задачи максимизации нелинейности векторной булевой функции с ограничением на сбалансированность для следующих случаев $(n = 5, m = 5)$, $(n = 6, m = 4)$, $(n = 8, m = 5)$ и $(n = 9, m = 2)$.

Для сбалансированного случая не ставится задача улучшить границу максимальной нелинейности снизу или даже достичь её, так как условие сбалансированности накладывает значительные ограничения на поиск, и максимальная нелинейность для сбалансированных функций может быть ниже, чем максимальная нелинейность произвольной булевой функции.

В таблице 8 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 6$ и $m = 4$ с нижеперечисленными характеристиками:

- метод Метрополиса: $\alpha = 0.95$, $it = 2000$, $k = 2$;
- метод поиска путём постепенного восхождения к вершине: $k = 2$;
- генетический алгоритм: $N = 100$, $it = 100$, $\beta = 0.1$.

Таблица 8 – Результат работы 100 тестов алгоритмов при $n = 6, m = 4$, сбалансированная задача

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	22	100	22	17.86
Метод поиска путём постепенного восхождения к вершине	24	27	22.54	0.24
Генетический алгоритм	22	100	22	31.9
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	24	63	23.26	24.59
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	24	57	23.14	38.77
Случайная генерация	22	100	22	20

Для $n = 6, m = 4$ максимальная нелинейность $N(6, 4) = 24 - 28$ (таблица 1). В сбалансированном случае для $n = 6, m = 4$ метод Метрополиса и генетический алгоритм работают не лучше, чем случайная генерация. Остальные алгоритмы достигли нижней границы максимальной нелинейности (таблица 8).

В таблице 9 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 5$ и $m = 5$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.95$, $it = 1000$, $k = 1$;

– метод поиска путём постепенного восхождения к вершине: $k = 2$;

– генетический алгоритм: $N = 80$, $it = 100$, $\beta = 0$.

Таблица 9 – Результат работы 100 тестов алгоритмов при $n = 5, m = 5$, сбалансированная задача

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	10	98	9.96	9.59
Метод поиска путём постепенного восхождения к вершине	10	85	9.7	0.07
Генетический алгоритм	10	43	8.86	20.8
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	10	100	10	10.55
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	10	84	9.68	20.86
Случайная генерация	10	20	8.4	20

Для $n = 5, m = 5$ максимальная нелинейность $N(5, 5) = 12$ (таблица 1). Все исследуемые алгоритмы работают лучше, чем случайная генерация. Среди чистых алгоритмов при данных управляемых параметрах можно выделить метод Метрополиса (таблица 9).

В таблице 10 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 8$ и $m = 5$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.95$, $it = 3000$, $k = 1$;

– метод поиска путём постепенного восхождения к вершине: $k = 2$;

– генетический алгоритм: $N = 100$, $it = 100$, $\beta = 0.1$.

Таблица 10 – Результат работы 20 тестов алгоритмов при $n = 8, m = 5$, сбалансированная задача

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	106	20	106	250.17
Метод поиска путём постепенного восхождения к вершине	108	8	106.6	52.87
Генетический алгоритм	106	1	104	288.32
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	108	8	106.8	302.37
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	108	7	106.7	346.96
Случайная генерация	104	6	102.6	250

Для $n = 8, m = 5$ максимальная нелинейность $N(8,5) = 112 - 120$ (таблица 1). Все алгоритмы работают лучше, чем случайная генерация. Метод поиска путём постепенного восхождения к вершине работает лучше, чем чистые алгоритмы, и не хуже, чем комбинация алгоритмов, при значительно меньшем времени работы (таблица 10).

В таблице 11 продемонстрированы результаты выполнения алгоритмов в пространстве векторных функций алгебры логики при $n = 9$ и $m = 2$ с нижеперечисленными характеристиками:

– метод Метрополиса: $\alpha = 0.95$, $it = 3000$, $k = 1$;

– метод поиска путём постепенного восхождения к вершине: $k = 2$;

– генетический алгоритм: $N = 100$, $\beta = 0.1$.

Таблица 11 – Результат работы 100 тестов алгоритмов при $n = 9, m = 2$, сбалансированная задача

Алгоритмы	max_N_F	$num\_max\_N_F$	EN_F	$Etime$
Метод Метрополиса	232	74	231.48	50.42
Метод поиска путём постепенного восхождения к вершине	236	2	232.42	39.75
Генетический алгоритм	230	2	227.51	73.51
Метод Метрополиса + метод поиска путём постепенного восхождения к вершине	234	29	232.58	91.78
Генетический алгоритм + метод поиска путём постепенного восхождения к вершине	234	18	232.55	116.19
Случайная генерация	226	100	226	80

Для $n = 9, m = 2$ максимальная нелинейность $N(9, 2) = 240 - 244$ (таблица 1). Все алгоритмы работают лучше, чем случайная генерация. Лучшую статистику имеет алгоритм восхождения на вершину (таблица 11).

Все алгоритмы в рассмотренных случаях работают лучше, чем случайная генерация. Комбинации алгоритмов, как и в задаче без ограничений, обладают лучшей статистикой по функциям, но имеют наибольшее время работы. Также выделим метод поиска путём постепенного восхождения к вершине, который для сбалансированной задачи всегда работает в несколько раз быстрее по сравнению с другими алгоритмами и имеет статистику по функциям в трех случаях из четырех не хуже, чем комбинации алгоритмов.

Метод поиска путём постепенного восхождения к вершине для задачи без учёта сбалансированности имеет лучшую статистику по функциям среди некомбинированных алгоритмов. Для задачи с учётом сбалансированности метод поиска путём постепенного восхождения к вершине работает в несколько раз быстрее других алгоритмов и часто ищет функции не хуже, чем комбинированные методы поиска.

Заключение

К задачам поиска векторных булевых функций с высокой нелинейностью и поиска сбалансированных векторных булевых функций с высокой нелинейностью были применены эвристические алгоритмы: метод Метрополиса, метод поиска путём постепенного восхождения к вершине, генетический алгоритм и их комбинации. Эти алгоритмы были написаны на языке Java. Также был применён генетический алгоритм Matlab.

Чтобы оценить работу запрограммированных алгоритмов относительно максимально возможного значения нелинейности, были изучены теоретические границы максимальной нелинейности. С помощью исследуемых алгоритмов для случаев $n = 5$, $m = 6$ и $n = 8$, $m = 9$ удалось уточнить теоретические границы максимальной нелинейности снизу.

По работе алгоритмов для обеих задач максимизации нелинейности векторной булевой функции, с учётом требования сбалансированности и без, была собрана статистика для различных n и m. По ней можно сделать вывод, что лучшей статистикой по найденным функциям обладают комбинированные методы поиска, комбинации генетического алгоритма с методом поиска путём постепенного восхождения к вершине и метод Метрополиса с методом поиска путём постепенного восхождения к вершине. Однако, как стоило ожидать, комбинации алгоритмов имеют наибольшее время работы.

Метод поиска путём постепенного восхождения к вершине для задачи без учёта сбалансированности имеет лучшую статистику по функциям среди некомбинированных алгоритмов. Для задачи с учётом сбалансированности метод поиска путём постепенного восхождения к вершине работает в несколько раз быстрее других алгоритмов и часто ищет функции не хуже, чем комбинированные методы поиска.

Список используемой литературы

1. Агафонова, И.В. Алгебраическая нормальная форма булевой функции и бинарное преобразование Мёбиуса / И.В. Агафонова – 2013. – 9 с.
2. Логачёв, О.А. Булевы функции в теории кодирования и криптологии / О.А. Логачёв, А.А. Сальников, В.В. Ященко – М.: МЦНМО, 2004. – 470 с.
3. Лопатин, А.С. Метод отжига. Стохастическая оптимизация в информатике: Межвуз. сб. / Под ред. О.Н. Граничина / А.С. Лопатин – СПб.: Издательство СПбГУ, 2005. – С. 133-149.
4. Мак-Вильямс, Ф.Дж. Теория кодов, исправляющих ошибки / Ф.Дж. Мак-Вильямс, Н.А. Слоэн – М.: Связь, 1979. – 744 с.
5. Токарева, Н.Н. Нелинейные булевы функции: бент-функции и их обобщения / Н.Н. Токарева – Издательство LAP LAMBERT Academic Publishing (Saarbrucken, Germany), 2011. – 180 с.
6. Carlet, C. Boolean Methods and Models, chapter Vectorial boolean functions for cryptography / C. Carlet – Cambridge University Press, 2008.
7. Chabaud, F. Links between differential and linear cryptanalysis. In: De Santis A. (eds) Advances in Cryptology – EUROCRYPT'94. Lecture Notes in Computer Science, vol 950 / F. Chabaud, S. Vaudenay. – Springer, Berlin, Heidelberg, 1994. – Pp. 356-365.
8. Clark, J.A. The design of S-boxes by simulated annealing. New Generation Computing, vol 23(3) / J.A. Clark, J.L. Jacob, S. Stepney. – 2005. – Pp. 219-231.
9. Kavut, S. Generalized rotation symmetric and dihedral symmetric boolean functions – 9 variable boolean functions with nonlinearity 242. In Applied Algebra, Algebraic Algorithms and Error-Correcting Codes / S. Kavut, M.D. Yücel Vaudenay. – Springer, Berlin, Heidelberg, 2007. – Pp. 321-329.
10. Maitra, S. Boolean functions on odd number of variables having

nonlinearity greater than the bent concatenation bound. Boolean Functions in Cryptology and Information Security / S. Maitra – 2008. – Pp. 173-182.

11. Millan, W. Boolean function design using hill climbing methods. In Information Security and Privacy / W. Millan, A. Clark, E. Dawson. – Springer, 1999. – Pp. 1-11.

12. Millan, W. Evolutionary heuristics for finding cryptographically strong s-boxes. In: Varadharajan V., Mu Y. (eds) Information and Communication Security. ICICS 1999. Lecture Notes in Computer Science, vol 1726 / W. Millan, L. Burnett, G. Carter, A. Clark, E. Dawson. – Springer, 1999. – Pp. 263-274.

13. Millan, W. Heuristic design of cryptographically strong balanced boolean functions. In EUROCRYPT 98, LNCS 1403 / W. Millan, A. Clark, E. Dawson. – Springer-Verlag, 1998. – Pp. 489-499.

14. Millan, W. How to improve the nonlinearity of bijective s-boxes. In: Boyd C., Dawson E. (eds) Information Security and Privacy. ACISP 1998. Lecture Notes in Computer Science, vol 1438 / W. Millan. – 1998. – Pp. 181-192.

15. Millan, W. Smart hill climbing finds better boolean functions. In Proceedings of the Workshop on Selected Areas on Cryptography / W. Millan, A. Clark, E. Dawson. – Springer-Verlag, 1997. – Pp. 50-63.

16. Nyberg, K. Differentially uniform mappings for cryptography. In: Helleseth T. (eds) Advances in Cryptology – EUROCRYPT ’93. Lecture Notes in Computer Science, vol 765 / K. Nyberg – Springer, Berlin, Heidelberg, 1994. – Pp. 55-64.

17. Nyberg, K. Perfect nonlinear s-boxes. In: Davies D.W. (eds) Advances in Cryptology – EUROCRYPT ’91. Lecture Notes in Computer Science, vol 547 / K. Nyberg – Springer, Berlin, Heidelberg, 1992. – Pp. 378-386.

18. Nyberg, K. S-boxes and round functions with controllable linearity and differential uniformity. In: Preneel B. (eds) Fast Software Encryption. FSE 1994. Lecture Notes in Computer Science, vol 1008 / K. Nyberg – Springer, Berlin, Heidelberg, 1995. – Pp. 111-130.

19. Patterson, N.J. The covering radius of the (2^{15}, 16) Reed-Muller code is at least 16276. IEEE Trans. Inform. Theory vol IT-29 / N.J. Patterson, D.H. Wiedemann. – 1983. – Pp. 354-356.

20. Wadayama, T. Upper and lower bounds on maximum nonlinearity of n-input m-output boolean function. Designs, Codes and Cryptography, 23(1) / T. Wadayama, T. Hada, K. Wakasugi, M. Kasahara – 2001. – Pp. 23-34.