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 Д.Л. Головко
(Инициалы Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Т.С. Якушева
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

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

Бакалаврская работа посвящена разработке эвристических алгоритмов и применению их для задач поиска максимально нелинейных векторных булевых функций.

В ходе выполнения исследований по бакалаврской работе была постановлена задача максимально нелинейных векторных булевых, были рассмотрены и реализованы эвристические алгоритмы, а также проведен сравнительный анализ выбранных алгоритмов и приведен результат работы этих алгоритмов.

Во введении прописывается актуальность темы, написана цель и задачи бакалаврской работы.

В первом разделе поставлена задача максимально нелинейных векторных булевых функций, построены границы ограничений максимальной нелинейности.

Во втором разделе описываются эвристические алгоритмы, после описания алгоритмы сравниваются и выбирается два наиболее подходящих для поставленной задачи, далее эти алгоритмы реализуются в MATLAB.

В третьем разделе проводится сравнительный анализ выбранных алгоритмов и приводится их результат работы.

В заключении прописаны выводы по всей работе.

Бакалаврская работа состоит из введения, трех разделов, заключения и списка используемой литературы.

В ней присутствуют 10 рисунков, 5 таблиц. Список используемой литературы состоит из 20 источников. Общий объем выпускной квалификационной работы составляет 46 страниц.

Abstract

The title of the bachelor's thesis is *Application of heuristic algorithms to the problems of maximum nonlinear vector boolean functions search*.

The research is devoted to the development of heuristic algorithms and their application for the tasks of searching for maximally nonlinear vector boolean functions.

When doing a research, the formulation of the problem of maximally nonlinear vector booleans was analyzed, heuristic algorithms were analyzed and implemented, and a comparative analysis of the selected algorithms is carried out and the result of these algorithms is presented.

The introduction reveals the relevance of the research and gives a brief description of the work done.

The first chapter the formulation of the problem of maximally nonlinear vector boolean functions is analyzed, the limits of maximum nonlinearity constraints are constructed.

The second chapter heuristic algorithms are described, after the description, the algorithms are compared and the 2 most suitable for the task are selected, then these algorithms are implemented in MATLAB.

The third chapter comparative analysis of the selected algorithms is carried out and their results are presented.

In conclusion, conclusions of the entire word are drawn.

The bachelor's thesis consists of an introduction, three chapters, a conclusion and a list of used literature.

It contains 10 figures, 5 tables. The list of used literature consists of 20 sources. The total volume of the final qualifying work is 46 pages.

Содержание

Введение

Векторные булевы функции применяются в криптографических приложениях, ради чего они обязаны владеть нужными особенностями для того, чтобы гарантировать стойкость к известным видам криптоанализа. Такие функции должны обладать относительно простой структурой. Одним из условий обеспечения устойчивости против криптографических атак является высокая или максимальная нелинейность. Именно это и интересует нас в рамках выпускной квалификационной работы.

Для реализации нашей работы, мы найдем и построим верхние и нижние границы максимальной нелинейности. Благодаря построенным границам, после реализации эвристических алгоритмов, мы сможем их сравнить.

Цель выпускной квалификационной работы – сравнительный анализ эвристических алгоритмов решение задачи поиска максимально нелинейных векторных булевых функций.

Для поставленной цели необходимо выполнить следующие задачи:

1. Описать задачу поиска максимально нелинейных векторных булевых функций;
2. Найти и построить верхние и нижние границы максимальной нелинейности;
3. Сформулировать существующие эвристические алгоритмы решения данной задачи;
4. Реализовать эти алгоритмы;
5. Провести сравнительный анализ реализованных алгоритмов.

1 Исследование предметной области и уточнение границ максимальной нелинейности

1.1 Постановка задачи максимально нелинейных векторных булевых функций

В первом разделе данной квалификационной работы мы формулируем задачу поиска максимально нелинейных векторных булевых функций. Эту задачу мы можем реализовать как с ограничениями, так и без них. Для того чтобы это сделать мы должны для начала выделить и проанализировать несколько нужных определений.

Определение 1. «Булева функция от n переменных – отображение $V_2 \to F_2$,где F_2 – поле из двух элементов, нуля и единицы, V_2 – векторное пространство над полем F_2, состоящее из 0 – 1 векторов длины n с операциями покомпонентного сложения по модулю два и умножения на скаляр» [15].

Первое определение дает нам представление о том, что такое булева функция, благодаря этому теперь мы будем обозначать ее (n, m), где n – количество входных данных, а m – количество выходных данных.

Далее дадим определение векторной булевой функции:

Определение 2. «Векторной булевой функцией или булевой (n,m)-функцией от n переменных называется упорядоченный комплект функций $F(x) \coloneqq (f_1(x), f_2(x), \dots, f_m(x))$, где $x = (x_1, x_2, \dots, x_n)$ и $f_i(x)$ − булевы функции, i = 1, ..., m» [15].

Для удобства дальнейшей работы будем обозначать векторные булевы функции латинскими заглавными буквами, а латинскими строчными буквами будем обозначать одномерные булевы функции.

Любое число $u \in 0 : 2n-1$ посредством последовательного деления на два можно единственным образом представить в виде:

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

где все коэффициенты u_i равны нулю или единице.

В дальнейшем, число в десятичной системе будем обозначать с помощью единиц и нулей вектора n. Так, например, число 6 в шестибитном представление будет равняться (0, 0, 0, 1, 1, 0). Нули и единицы – это количество битов, которые представлены в двоичной системе счисления.

Мы можем представить одномерную булеву функцию в виде вектора значений длины 2^n из нулей и единиц $f(0), f(1), \ldots, f(2^n - 2), f(2^n - 1))$ и векторную (n, m) в виде матрицы, строки которой образуются одномерными булевыми функциями, где n принимает конечное число значений. Тогда размерность этой матрицы составляет $m \times 2^n$, рассмотрим следующий пример:

$$\begin{pmatrix} f_1(0), f_1(1), \ldots, f_1(2^n - 1) \\ f_2(0), f_2(1), \ldots, f_2(2^n - 1) \\ \ldots \quad \ldots \quad \ldots \quad \ldots \\ f_m(0), f_m(1), \ldots, f_m(2^n - 1) \end{pmatrix}. \qquad (2)$$

Исходя из этого примера, получаем пространство булевых функций, которое можно обозначить V_{2^n}, а пространство векторных булевых функций (n, m) - функций – $V_{2^n}^m$.

«Операции в поле F_2 будем обозначать «·» – умножение по модулю 2 и «$\oplus$» – сложение по модулю 2, кроме того, нам понадобиться и операция «$\vee$». При этом в алгебраических выражениях мы часто будем для удобства опускать знак «·», подразумевая, что a·b = ab. Для покомпонентного сложения по модулю 2 векторов из V_n мы используем то же обозначение «$\oplus$». Всякий раз из контекста будет ясно, к элементам какого пространства применяется эта операция» [15].

Теперь нам необходимо установить понятие нелинейности, для этого будем использовать расстояние Хемминга.

Определение 3. «Расстоянием Хемминга *d(f, g)* между булевыми функциями f и g от n переменных называется число позиций в векторах значений, в которых функции различны» [15].

Далее, в определении 4 разберем и рассмотрим, одномерную булеву функцию.

Определение 4. «Нелинейность булевой функции $\mathrm{f}(\mathrm{x}_1, \dots,)\mathrm{x}_\mathrm{n}$ от n переменных это $\mathrm{N_f} := \min_{(\mathrm{a}_0,\dots,\mathrm{a}_\mathrm{n})\in \mathrm{V}_{\mathrm{n}-1}} \mathrm{d}(\mathrm{f}(\mathrm{x}_1, \dots, \mathrm{x}_\mathrm{n}),\ \mathrm{a}_0 \oplus \mathrm{a}_1\mathrm{x}_1 \oplus \dots \oplus \mathrm{a}_\mathrm{n}\mathrm{x}_\mathrm{n})$ – расстояние Хемминга» [15].

Введем новую одномерную булеву функцию на основе векторной булевой функции, чтобы расширить понимание нелинейности для класса векторных булевых функций. Пусть $\theta \in V_m$, $F(x) = (f_1(x), \dots , f_m(x))$ – векторная (n, m)-функция. И получим новую булеву функцию:

$$\theta \cdot F(x) := \theta_1 f_1(x) \oplus \theta_2 f_2(x) \oplus \dots \oplus \theta_m f_m(x). \qquad (3)$$

Определение 5. «Нелинейностью векторной булевой (n, m)-функции $\mathrm{F(x)} := (\mathrm{f}_1(\mathrm{x}), \mathrm{f}_2(\mathrm{x}), \dots, \mathrm{f}_\mathrm{m}(\mathrm{x}))$ от n переменных называется целое число $\mathrm{N_f} := \min_{\theta\in \mathrm{V_m}\setminus 0^\mathrm{m}}\{\mathrm{N}_{\theta\cdot \mathrm{F}}\}$, где $\theta \cdot \mathrm{F} := min$» [15].

Проанализировав вышеприведенные определения, можно составить экстремальную задачу для максимизации нелинейности векторных булевых функций:

$$N_F \rightarrow max_F\ , F \in V_{2^n}^m\ , \qquad (4)$$

Для того чтобы вычислить нелинейность одномерной булевой функции есть несколько способов. Мы можем применить преобразование Уолша–Адамара. С математической точки зрения это преобразование будет более удобное.

Определение 6. «Преобразованием Уолша–Адамара булевой функции f от n переменных называется целочисленная функция на $\mathrm{V_n}$, которую можно определить следующим равенством:

$$W_f(u) := \sum_{x \in V_n}(-1)^{f(x)\oplus\langle x,y\rangle}, \text{ где } \langle x,y\rangle = x_1u_1 \oplus \dots \oplus x_nu_n. \quad (5)$$

Каждой одномерной функции от n переменных мы можем сопоставить вектор, состоящий из коэффициентов Уолша–Адамара, он выглядит следующим образом: $f \to (W_f(0), W_f(1), \dots, W_f(2^n-2), W_f(2^n-1))$, его мы будем называть спектром Уолша – Адамара функции f» [15].

Далее, для того чтобы связать преобразование Уолша–Адамара и нелинейность одномерной булевой функции, воспользуемся следующей леммой:

Лемма 1. «Нелинейность булевой функции можно вычислить как $\mathrm{N_f} = \frac{1}{2}(2^{\mathrm{n}} - \mathrm{W_{max}})$, где $\mathrm{W_{max}}$ – максимальное по модулю значение спектра Уолша - Адамара функции f» [15].

Определение 7. «Спектром Уолша−Адамара для (n, m)-функции F будем называть матрицу размерности $2^{\mathrm{m}} - 1 \times 2^{\mathrm{n}}$ вида $\mathrm{W_{\theta\cdot F}}(0), \mathrm{W_{\theta\cdot F}}(1), \dots, \mathrm{W_{\theta\cdot F}}(2^{\mathrm{n}}-1)_{\theta\in \mathrm{V_m}\setminus 0^{\mathrm{m}}}$, где каждая строчка спектр Уолша–Адамара для одномерной функции $\theta \cdot \mathrm{F},\ \theta \in \mathrm{V_m}\setminus 0^{\mathrm{m}}$» [15].

На первом рисунке покажем спектр и нелинейность для векторной (2,3) - функции F.

	F		$\theta \cdot F$		спектр				$N_{\theta\cdot F}$
			f_1	$1,1,0,0$	*0,*	*0,*	*-4,*	*0*	0
			f_2	$1,1,1,0$	*-2,*	*-2,*	*-2,*	*2*	1
f_1	$1,1,0,0$	$\to$	$f_1 \oplus f_2$	$0,0,1,0$	*2,*	*-2,*	*2,*	*2*	1
f_2	$1,1,1,0$		f_3	$1,1,1,1$	*-4,*	*0,*	*0,*	*0*	0
f_3	$1,1,1,1$		$f_3 \oplus f_1$	$0,0,1,1$	*0,*	*0,*	*4,*	*0*	0
			$f_3 \oplus f_2$	$0,0,0,1$	*2,*	*2,*	*2,*	*-2*	1
			$f_3 \oplus f_2 \oplus f_1$	$1,1,0,1$	*-2,*	*2,*	*-2,*	*-2*	1

Рисунок 1 – Вычисление спектра и нелинейности

Минимальное значение из столбца Nθ•F является нелинейностью функции F по определению, значит нелинейность F равна нулю.

Из определения 5, где мы выяснили нелинейность, вытекает следующая лемма.

Лемма 2. «Нелинейность векторной булевой (n, m)-функции F можно вычислить как $N_F = \frac{1}{2}(2^n - W_{max}$, где $W_{max} := max_{u \in V_n, \theta \in V_m \setminus 0^m}(|W_{\theta \cdot F}(u)|)$ – максимальный по модулю элемент спектра Уолша–Адамара векторной функции F» [15].

После рассмотрения Леммы 2, наша задача максимизации нелинейности векторной булевой функции (4) эквивалентна задаче минимизации максимального по модулю значения элемента спектра Уолша–Адамара:

$$\max_{u \in V_n, \theta \in V_m \setminus 0^m}(|W_{\theta \cdot F}(u)|) \rightarrow \min_F,\ F \in V_{2^n}^m \qquad (6)$$

1.2 Границы максимальной нелинейности

Для построения оценки максимальной нелинейности (n, m)-функции будем использовать леммы, теоремы и утверждения. В конце мы построим таблицу различных n и m ограничений максимальной нелинейности (n, m)-функции.

Определение 8. «Если $\exists$ (n, m)-функция F с нелинейностью N_f (одинаковые оба) и не $\exists$ функции $G : N_G > N_F$, то величину N_F называют максимальной нелинейностью $N(m, n)$» [15].

Для нелинейности произвольной булевой функции f, а значит и для произвольной векторной (n, m)-функции справедлива верхняя оценка.

Лемма 3.

$$«N(n,m) \leq 2^{n-1} - 2^{(n-2)/2}, n \geq 2, \quad (7)$$

Есть классы булевых одномерных функций, которые могут достигать максимума нелинейности, например, бент –функции» [15].

Введем и определимся с тем, что такое бент–функция и для чего она нам нужна.

Определение 9. «Бент–функция – булева функция от четного числа переменных n, достигающая максимума нелинейности $2^{n-1} - 2^{(n-2)/2}$» [15].

Далее обобщим понятие бент–функции для векторных (n, m)-функций следующим образом:

Определение 10. «Векторная (n,m)–бент – функция $F = (f_1(x), f_2(x), \dots, f_m(x))$ называется бент–функцией, если каждая ненулевая линейная комбинация $f_1, f_2, \dots, f_m$ является бент–функцией» [15].

Указанное нами неравенство (7) может обращаться в равенство при следующих условиях:

Лемма 4. «Векторная (n, m)-бент–функция существует тогда и только тогда, когда $n \geq 2m$ и n четно. Это эквивалентно тому, что $N(n,m) = 2^{n-1} - 2^{(n-2)/2}$, если $n \geq 2m$ и n четно» [15].

Доказано, что приведенное ниже неравенство (8) сильнее, чем неравенство (7):

Лемма 5. «Граница Сидельникова–Шабата–Воденая:

$$N(n,m) \leq 2^{n-1} - \frac{(3 \cdot 2^n - 2 - 2^{\frac{(2^n-1)(2^{n-1}-1)}{2^m-1}})^{1/2}}{2}, m > n-2. \quad (8)$$

Правая часть в неравенстве (6) не обязательно целочисленная. Вследствие этого мы можем использовать округление в меньшую сторону в качестве оценки сверху» [15].

В лемме 6 рассмотрим, при каких значениях n и m функция четная или нечетная:

Лемма 6. «$N(n,m) = 2^{n-1} - 2^{(n-1)/2}$, если n нечетно и $n = m$» [15].

В утверждение 2 доказан следующий факт:

Утверждение 2. «Для n = 3, 5, 7, n > m верно, что N (n, m) = $2^{n-1} - 2^{(n-1)/2}$» [15].

В дальнейшем для того, чтобы оценить максимальную нелинейность снизу и сверху, нам понадобятся линейные двоичные коды.

Определение 11. «Линейный двоичный (n, k) код – это линейное подпространство C размерности k векторного пространства V_2» [15].

Векторы из C называют кодовыми словами, n – длиной слова, k – размерностью пространства кодовых слов, кодовым расстоянием $D(C)$ – минимальное расстояние между кодовыми словами.

Чтобы нам не повторяться, в дальнейшем будем говорить о линейных кодах без уточнения, что они двоичные, потому что использовать мы будем только двоичные коды.

Линейный код, который имеет параметры: длину слова n, размерность пространства кодовых слов k, кодовое расстояние D будем стандартно обозначаться как (n ,k, D) –код.

Теорема 1. «Если ни при каком $D \geq N_f$ не существует линейного $(2n, 1+m, D)$ кода, тогда не существует (n, m) – функции с нелинейностью равной N_f » [15].

Будем использовать теорему 1, если нужно узнать параметры, при которых линейный код не существует.

Теорема 2. «Для любого кода длины n, размерности k с минимальным кодовым расстоянием D выполняется: $n - k \geq \log_2(\sum_{i=0}^{[D-1/2]} C_n^i)$» [15].

Из теорем 1 и 2 вытекает следующее следствие.

Следствие 1. «Если мы можем выполнить следующее неравенство: $2^n - (n + 1 + m) < \log_2(\sum_{i=0}^{[D-1/2]} C_n^i)$, то не существует ($n$, m) – функции с нелинейностью равной D» [15].

Для следующей леммы нам нужно ввести новое определение.

Определение 12. «Код называется самодополняющим, если из того, что он содержит вектор ϑ, следует, что он содержит вектор $1^n - \vartheta$» [15].

Например, если самодополняющий код содержит вектор (0, 1, 1, 1, 1, 1), он также должен содержать вектор (1, 0, 0, 0, 0, 0).

Лемма 7. «Если не существует линейного самодополняющего $(2^n, n + 1 + m,\ D)$ кода с параметром $D \geq N_F$, тогда не существует (n, m)-функции с нелинейностью равной N_F» [15].

Для теоремы 3 нам нужно установить дополнительное обозначение. Количество кодовых слов в коде C с весом i отметим как A_i, полином Кравчука определим как $P_k(i) \coloneqq \sum_{0 \leq j \leq k} (-1)^j C_i^j C_{l-i}^{k-j}$, где $C_n^k \coloneqq \frac{n(n-1)(n-2)\cdots(n-(k-1))}{k!}$ и $n \in Z, k, i \in N \cup 0$ – обобщенный бином.

Теорема 3. «Не существует линейного самодополняющего кода с параметрами $(l = 2^n, \left[\log_2(2 + S_{opt})\right], D)$, где S_{opt} – оптимальное значение функции для следующей задачи линейного программирования:

$$2(A_D + A_{D+1} + \ldots + A_{l/2-1}) + A_{l/2} \rightarrow max \qquad (9)$$

при ограничениях:

$$\sum_{D \leq i \leq l/2-1} A_i (P_k(i) + P_k(l-i) + A_{l/2} \geq -2C_l^k \qquad (10)$$

для четных $k,\ 2 \leq k \leq l$:

$$A_i \geq 0, D \leq i \leq l/2» \text{ [15]}. \qquad (11)$$

Также отметим важное, очевидно втекающее из определения нелинейности векторной функции свойство.

Замечание 1. «При m > 1 имеет место неравенство $N(m, n) \leq N(m - 1, n)$» [15].

Утверждение 3. «Для $m = 1, n \geq 15$ верно, что $N(n,n) > 2^{n-1} - 2^{(n-1)/2}$» [15].

Лемма 8. «Если n четно, 1 < n < 2m и m ≤ n, то N (n, m) ≥ 2n−1 − 2n/2.

Известно, что всякую булеву функцию можно изобразить в виде алгебраической нормальной формы (АНФ или полином Жегалкина) – это форма представления булевой функции в виде полинома с коэффициентами вида 0 и 1, в котором в качестве произведения применяется операция конъюнкции, а в качестве сложения – сложение по модулю 2. Степенью булевой функции называют степень исходного полинома» [15].

«Векторы значений множества булевых функций от n переменных степени не больше r образуют код, который называется кодом Рида – Маллера r – го порядка длины 2^n, $RM_{r,n}$. Заметим, что код $RM_{r',n}$ включает в себя код $RM_{r,n}$, если $r' > r$, или, как мы будем говорить чаще, $RM_{r',n}$ является надкодом $RM_{r,n}$»[15].

Теорема 4. «Если линейный $(2^n, K, D)$ код является надкодом для кода Рида-Маллера первого порядка $RM_{1,n}$ и $1 \leq m \leq K - n - 1$, то существует (n, m) – функция с нелинейностью $N_f \geq D$» [15].

Отметим, что доказательство данной теоремы конструктивное. Известно, что код Рида – Маллера $RM_{r,n}$ имеет следующие параметры: длину 2^n, размерность $S = 1 + C_n^1 + \cdots + C_n^r$ и минимальное кодовое расстояние 2^{n-r} [8]. С учетом этих фактов и теоремы 3 получим:

Следствие 2. «Если $1 \leq m \leq C_n^1 + \cdots + C_n^r - n$, то существует (n, m)-функция с нелинейностью $N_f \geq 2^{n-r}$» [15].

Лемма 9. «Дуальный код к расширенному БЧХ коду, $exDCH_{2t+1,n}^{\perp}$ является надкодом для $RM_{1,n}$ и имеет параметры: длину 2^n, размерность $K = tn + 1$, минимальное кодовое расстояние $D \geq 2^{n-1} - (t-1)2^{n/2}$ при$t > 1$ $и$ $2t - 1 < 2^{[n/2]} + 1$» [15].

Из теоремы 4 и леммы 9 следует следствие 3:

Следствие 3. «$(n,\ m)$ – функция с нелинейностью $D \geq 2^{n-1} - (t-1)2^{n/2}$ может быть построена при $1 \leq m \leq (t-1)n, t > 1$ и $2t-1 < 2^{[n/2]} + 1$» [15].

1.3 Таблица ограничений максимальной нелинейности

Основываясь на определениях, теоремах и утверждениях из нашего вышеприведенного списка ограничений максимальной нелинейности мы можем сформировать таблицу ограничений (таблица 1). В некоторых ячейках полученной таблицы указано два числа – это верхний и нижний предел. Если мы видим одно число, то верхний или нижний пределы совпадают. Каждое из приведенных в таблице ограничений опирается на определение, теорему, лемму или утверждение, которые обозначены через соответствующие степени α, β,..., μ, ν. Конкретные утверждения из которых вытекает то или иное ограничение указанны ниже таблицы 1. Таблица 1 приведена для значений n в промежутке [3;9] и m в промежутке [1;9]. Для дальнейшего изучения таблицу можно будет достроить для больших значений n и m.

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

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^{κ}-5^{α}	$4^{\kappa,\zeta}$	$4^{\kappa,\zeta}$	$4^{\kappa,\zeta}$	$2^{\kappa,\zeta}$	$2^{\kappa,\zeta}$	$2^{\kappa,\zeta}$
5	12^{ϵ}	12^{ϵ}	12^{ϵ}	12^{ϵ}	12^{δ}	8^{κ}-10^{ζ}	8^{κ}-10^{ζ}	8^{κ}-9^{ζ}	8^{κ}-9^{ζ}
6	28^{β}	28^{β}	28^{β}	24^{ι}-28^{α}	24^{ι}-26^{ζ}	24^{λ}-26^{γ}	24^{μ}-25^{γ}	$24^{\mu,\gamma}$	$24^{\mu,\gamma}$
7	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	56^{ϵ}	48^{μ}-54^{η}	48^{μ}-54^{η}
8	120^{β}	120^{β}	120^{β}	120^{β}	112^{λ}-120^{α}	112^{λ}-120^{α}	112^{λ}-120^{α}	112^{λ}-116^{γ}	96^{λ}-115^{γ}
9	242^{ξ}-244^{α}	240^{ν}-244^{α}	240^{ν}-$244^{\alpha\nu}$	240^{ν}-244^{α}	240^{ν}-244^{α}	240^{ν}-244^{α}	240^{ν}-244^{α}	240^{ν}-244^{α}	240^{δ}

α: «Есть классы булевых одномерных функций, которые достигают максимума нелинейности, например, бент –функции» [15]

β: «Векторная (n, m) – бент–функция существует тогда и только тогда $n \geq 2m$ и n четно. Это эквивалентно тому, что $N(n,m) = 2^{n-1} - 2^{(n-2)/2}$, если $n \geq 2m$ и n четно» [15].

γ: «Граница Сидельникова–Шабата–Воденая:

$$N(n,m) \leq 2^{n-1} - \frac{(3 \cdot 2^n - 2 - 2\frac{(2^n-1)(2^{n-1}-1)}{2^m-1})^{1/2}}{2}, m > n-2. \quad (12)$$

Правая часть в неравенстве (6) не обязательно целочисленная. Вследствие этого мы можем использовать округление в меньшую сторону в качестве оценки сверху» [15].

δ: «$N(n,m) = 2^{n-1} - 2^{(n-1)/2}$, если n нечетно и $n = m$» [15].

ϵ: «Для n = 3, 5, 7, n > m верно, что N (n, m) = 2n−1 − 2(n−1)/2» [15].

ζ: «Если ни при каком $D \geq N_f$ не существует линейного $(2n, 1+m, D)$ кода, тогда не существует (n, m)-функции с нелинейностью равной N_f » [15].

η: «Если не существует линейного самодополняющего $(2^n, n+1+m, D)$ кода с параметром $D \geq N_F$, тогда не существует (n, m)-функции с нелинейностью равной N_F» [15].

ι: «Если n четно, 1 < n < 2m и m ≤ n, то N (n, m) ≥ $2^{n-1} - 2^{n/2}$» [15].

κ: «Если линейный $(2^n, K, D)$ код является надкодом для кода Рида-Маллера первого порядка $RM_{1,n}$ и $1 \leq m \leq K - n - 1$, то существует (n, m)-функция с нелинейностью $N_f \geq D$» [15].

λ: «Дуальный код к расширенному БЧХ коду, $exDCH^{\perp}_{2t+1,n}$ является надкодом для $RM_{1,n}$ и имеет параметры: длину 2^n, размерность $K = tn + 1$, минимальное кодовое расстояние $D \geq 2^{n-1} - (t-1)2^{n/2}$ при $t > 1$ и $2t - 1 < 2^{[n/2]} + 1$» [15].

μ: Данная таблица уже была построена в статье [21].

ν: «При m > 1 имеет место неравенство $N(m,n) \leq N(m-1,n)$» [15].

ξ: «достигнуто эвристическим алгоритмом среди некоторых специальных классов функций» [19].

Выводы по разделу

В данном разделе мы провели исследование предметной области, была поставлена задача – описали задачу максимально нелинейных векторных булевых функций, а также нашли и построили границы максимальной нелинейности. Для описания задачи были сформулированы 12 определений, 4 теоремы, 3 следствия, 9 лемм и 3 утверждения.

2 Описание и реализация эвристических алгоритмов

Эвристический алгоритм – это алгоритм решения задачи, включающий практический метод. Данный алгоритм не является точным или оптимальным, но является достаточным для решения поставленной задачи. Алгоритм может помочь в том случае, если решение не может быть найдено точным алгоритмом или ее решение слишком длительное.

Правильность алгоритма для всех возможных случаев не доказана, но известно, что он дает достаточно хорошее решение в большинстве случаев. Он дает неверный результат только в редких случаях, реже случается, что алгоритм дает не точный результат, но приемлемый.

Важно знать про эвристический алгоритм следующие особенности:

- не гарантирует нахождение лучшего решения;
- не гарантирует нахождение лучшего решения, даже если оно есть;
- возможны случаи, в которых алгоритм дает неверное решение;

Алгоритм применяется для решения задач с высокой вычислительной сложностью, в случае если: перебор вариантов невозможен или он занимает слишком много времени и технически невозможно вычислить решение.

По причине того, что отсутствуют общие решения поставленной задачи, эвристические алгоритмы достаточно часто применяются в областях искусственного интеллекта, например, для распознавания образов.

В данном разделе будем рассматривать виды эвристических алгоритмов такие как: генетический алгоритм, алгоритм имитации отжига, алгоритм роя частиц и алгоритм восхождения на вершину. Также нам нужно выявить какие алгоритмы больше всего подходят к описанной в первом разделе задаче, потому что некоторые алгоритмы могут некорректно работать с определенными задачами.

2.1 Генетический алгоритм

Идея генетических алгоритмов начинается с генерации некоторой популяции особей. Каждая особь описывается хромосомой состоящая из конечного числа генов. Гены, как правило, являются минимальными количествами хромосом, которые могут в процессе скрещивания претерпевать определенные изменения. По сути дела каждая хромосома кодирует в той или иной форме вектор варьируемых параметров, работают не с самими параметрами, а с их кодами, хромосомами, где элементом кодирования является ген, строится модель, которая позволяет для любой хромосомы рассчитать значение целевой функции (фитнес–функции).

С помощью алгоритма селекции отбираются более приспособленные особи, которые могут участвовать в скрещивании. На основе селекции в конечном этапе часть особей, которые приспособлены плохо будут уничтожаться, но некоторые обстоятельства предполагают принудительное сохранение элиты, то есть лучших особей, которые сохраняются вне зависимости от того какой жребий им выпал. Процессы отбора строятся на основе вероятностных подходов, чем лучше значение фитнес–функции при минимизации, чем меньше ее значение, тем большая вероятность быть отобранной продолжить свое существование и участвовать в скрещивании.

Если мы с вами будем осуществлять только отбор и скрещивание, то, скорее всего наша популяция очень быстро выродится, сойдется к некоторому локальному минимуму, и не будет исследовать все пространство области допустимых значений. Для того чтобы разнообразить популяцию вводятся мутации, которые представляют собой случайные изменения отдельных генов в хромосомах. В конечном итоге мы получаем новую популяцию, которая лучше прежней и далее мы возвращаемся опять к вычислению фитнес–функции, после каждого вычисления фитнес–функции, вычисляется условие выхода из генетического алгоритма. Обычно таким условием является прекращения изменений в популяции или это изменение

становится слишком малым. В этом случаем мы полагаем, что генетический алгоритм сошелся к некоторому решению или группе решений.

Чтобы применить алгоритм к поставленной задаче нам могут понадобиться значения или управляемые параметры, которыми будут: N – размер популяции, it – количество итераций, α – доля мутантов в поколении, k – количество генов (элементов матрицы), изменяющихся у мутантов, β – доля отбора с возможностью повторения отбираемых функций (опционно).

В генетическом алгоритме набор решений (в нашем случае (n, m)-функций) изменяется благодаря схемам скрещивания, мутации и отбора, чтобы улучшить характеристики элементов множества решений.

Далее схематично на рисунке 1 опишем работу алгоритма поэтапно.

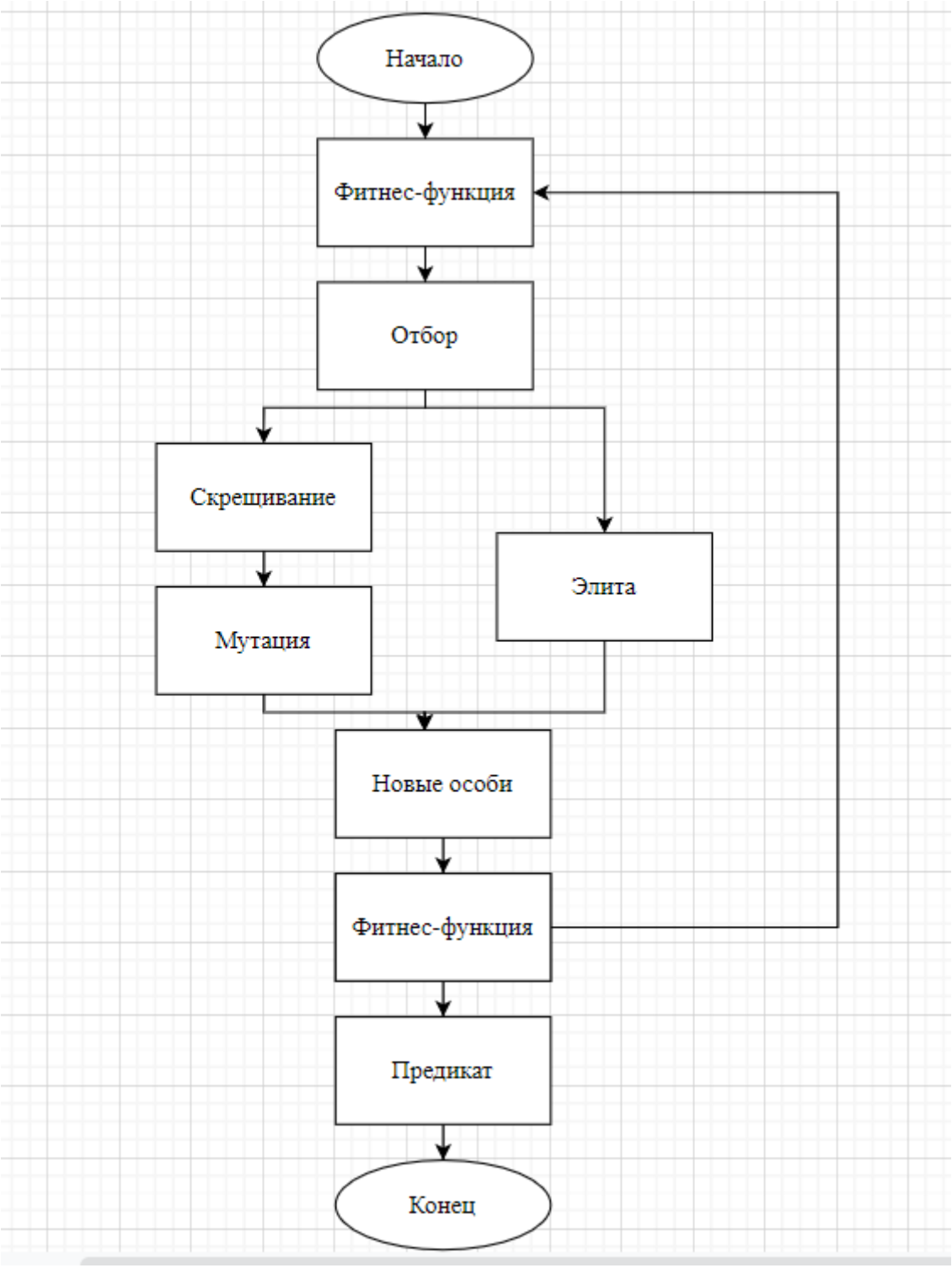

Рисунок 1 – Блок-схема генетического алгоритма

Рассмотрим блок-схему генетического алгоритма для поставленной задачи:

1. Начинается все с фитнес–функции, которую будем обозначать (n, m). Далее нам необходимо сгенерировать функцию $N\ (n,\ m)$. Благодаря этой функции мы сможем сгенерировать первую родительскую пару.
2. После нам нужно скрестить родительскую пару и матрицу ребенка. Для их обозначения используем $b1$, $b2$ и bc соответственно. После скрещивания родителей и ребенка, мы получим гены, которые будут являться ячейкой в матрице. Если мы видим, что родительская пара имеет одинаковое значение, то значит, ген ребенка будет иметь такое же значение, то есть $bc[i][j] = b1[i][j] = b2[i][j]$. Если родительская пара не будет равна между собой, то ген ребенка мы определим сами между значением 0 и 1, эти значения будут иметь одинаковую вероятность. Скрещивая всевозможные пары, породим $N(N-1)$ потомков. Во множества потомков добавим родительские функции, тогда мощность станет равной $N(N+1)$.
3. Следующим этапом является мутация. На данном этапе происходит изменение ячеек k в каждой функции. Изменение будет происходить с помощью $\alpha \bullet N\ (N+1)$ особей, где α – доля мутантов в поколении.
4. На последнем этапе происходит отбор потомков, где нам необходимо определить и выбрать функции с наибольшей нелинейностью. Если значения совпадают, то с помощью функции Уолша–Адамара мы будем отбирать особи, которые имеют наименьшее количество максимальных значений.

Действия 2–4 могут повторяться, все зависит от наступления условия остановки или количества итераций it.

2.2 Алгоритм имитации Отжига

Рассмотрим алгоритм имитации отжига, который использует упорядоченный случайный поиск на основе аналогии с процессом

образования кристаллической структуры в веществе с минимальной энергией при охлаждении.

В основе этого алгоритма лежит физический процесс отжига металла, в результате которого формируется кристаллическая решетка этого металла. То есть металл разогревается до какой-нибудь температуры, атомы начинают относительно свободное движение, колебание, а с остыванием они стремятся к энергетически выгодному состоянию и занимают свои места в кристаллической решетке. Такая идея лежит в основе алгоритма имитации отжига.

Мы рассматриваем одну точку, которая будет совершать движение, колебание по пространству области допустимого значения, причем амплитуда этих колебаний будет снижаться с течением времени, в котором условно говоря происходит уменьшение параметра идентичного температуре, пока эта частица не придет в наиболее энергетически выгодное решение. Ожидается, что это решение будет соответствовать глобальному экстремуму целевой функции, потому что аналогом энергии здесь является значение целевой функции.

Из начальной точки x мы начинаем наш алгоритм и далее случайным образом выбираем в пространстве допустимых значений новую точку x', положение которой зависит от предыдущего состояния и аналога температуры на каждой пятой итерации. Далее в зависимости от разности значений целевой функции в новой и старой точке а также с учетом изменений температуры мы заменяем старую точку на новую. В том случае, если замена произошла, то есть координаты x обновлены, то мы переходим к следующей итерации, увеличиваем k на 1 и проверяем, выполнено ли условия окончания работы алгоритма. Если да, то мы завершаем работу, если нет, то выполняем следующую итерацию. Если же у нас генерация новой точки оказалась неудачной, то есть не удалось ее обновить, то в этом случае остаемся на прежней итерации и пытаемся сгенерировать новую точку, для которой будет возможен переход.

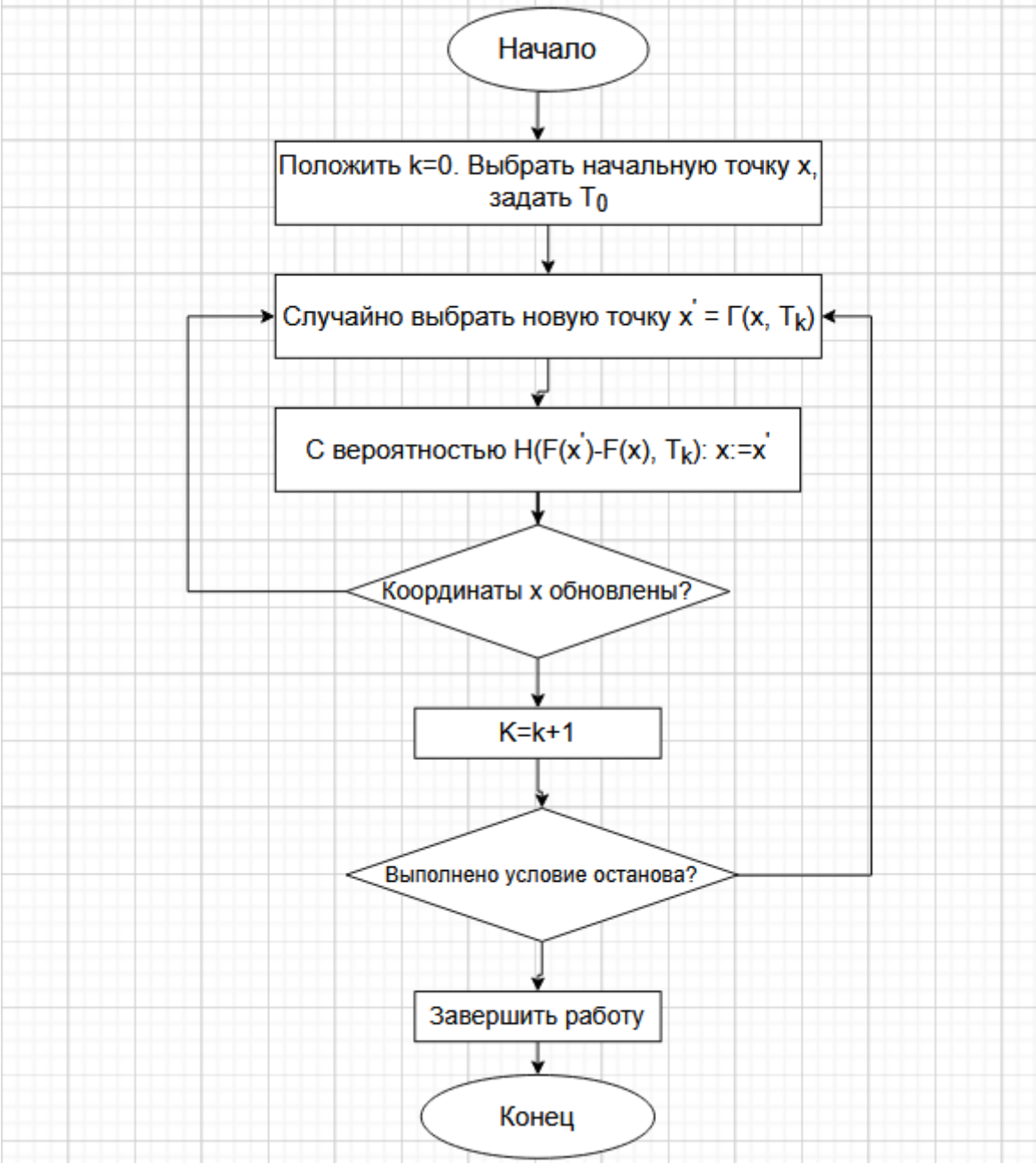

Рисунок 2 – Блок-схема алгоритма имитации отжига

Из свойств алгоритма имитации отжига можно ответить, что он может не включать в себе локальных минимумов и продолжать поиск максимума. На блок-схеме символ *T* обозначает температуру. Данный параметр является характеристикой, которая моделирует процесс и увеличивает значение

целевой функции. Увеличение температуры отражает увеличение вероятности ухудшающегося изменения.

Рассмотрим алгоритм с математической точки зрения:

1. Определим функцию изменения температуры *T(t)*.
2. Зададим минимальную и начальную температуры t_{min} и t_{max} соответственно, таким образом, что переход к функции с меньшей нелинейностью происходил наихудшим случаем соответственно с вероятностями 10^{-4} и 0,99.
3. Зададим или случайно сгенерируем (*n, m*)-функцию F_{glob}, также введем (*n, m*)-функцию F , так что $F = F_{glob}$.

4. «Пусть *it* – количество итераций на каждом уровне температуры. Для каждого фиксированного уровня температуры *t* выполним *it* – раз следующие шаги:

   - Случайным образом изменим *k* координат (не исключена возможность повторного изменения уже измененной координаты) в матрице (*n, m*)-функции *F* .
   - Если значение нелинейности (*n, m*) – функции *F* повысилось или уменьшилось число W_{max} в матрице спектра Уолша – Адамара, оставим данное изменение в (*n, m*) – функции *F* .
   - В противном случае оставим изменения с вероятностью$P\ (\Delta w) = e^{(-\Delta WH/2-1/t_i)}\ P$, *где* $\Delta w = W'_{max} - W_{max}$, W'_{max} – максимальное по модулю значение Уолша – Адамара для измененной функции, а W_{max} – изначальной.

5. Если значение нелинейности (n, m)-функции F стало больше значения нелинейности (n, m)-функции Fglob, изменим функцию Fglob = F .
6. Понизим температуру t = T (t) » [9].

Этапы 4-6 повторяются пока $t > t_{min}$

2.3 Алгоритм роя частиц

Перейдем к следующему алгоритму. Алгоритм роя частиц эмитирует элементы живой природы, например: рой насекомых, косяк рыбы или группы животных, которые перемещаются в пространстве и при этом изменяют свою скорость целенаправленно. Таким образом, чтобы приблизиться к привлекательным для них местам, то есть к месту, где расположена пища. В методах, которые имитируют поведения роя частиц привлекательными местами, являются места, где целевая функция достигает своих экстремальных значений. Общий вид алгоритма роя частиц: в начале, создается группа особей, каждая из которых имеет свои координаты в области допустимых значений и свою начальную скорость. Далее находится значение целевой функции для каждой частицы, в том случае если мы будем иметь с вами какую-то историю по нескольким итерациям этого алгоритма, то находится лучшее решение для каждой частицы, полученное на всех итерациях алгоритма, из них выделяется лучшее решение для всех частиц.

Исходя из текущего состояния, каждой частице, ее истории и истории всех частиц, производится коррекция скорости каждой частицы с использованием этой скорректированной скорости осуществляется перемещение частицы в новую точку. Проверяется выполнение критерия остановки алгоритма и в случае, если он не выполняется, то эти итерации повторяются, в противном случае мы завершаем работу.

Рисунок 3 – Блок-схема алгоритма роя частиц

Интересной частью этого алгоритм является коррекция скорости, именно от этого шага зависит сходимость алгоритма. Изначально коррекция скорости выглядела следующим образом:

$v_{i,t+1} = v_{i,t} + \varphi_p r_p (p_i - x_i, t) + \varphi_g r_g (g_i - x_i, t)$.

Здесь v_i, t – i-я компонента скорости при t-ой итерации алгоритма x_i, t – i-я координата частицы при t – ой итерации алгоритма, p_i – i-я координата лучшего решения, найденного частицей g_i – i-я координата лучшего

решения, найденного всеми частицами rp, rg – случайные числа в интервале (0, 1) φp, φg – весовые коэффициенты, которые надо подбирать под конкретную задачу.

Далее корректируем текущую координату каждой частицы.

$xi, t + 1 = xi, t + vi,t+1$.

2.4 Восхождение на вершину

Рассмотрим алгоритм поиска восхождения на вершину. Этот алгоритм представляет собой цикл, с помощью которого можно постоянно двигаться в возрастание или простыми словами подниматься.

Алгоритм заканчивает работу после того, как достигается вершина или пик, в котором больше нет более высокого значения. Этот алгоритм нельзя сравнивать с деревом поиска, потому что нам необходимо регистрировать только состояние и соответствующее ему значение целевой функции. В алгоритме восхождения на вершину нельзя делать прогнозирование.

Разберем описание алгоритма восхождения на вершину для нашей задачи. Первым шагом мы зададим или же случайно сгенерируем (n, m) – функцию F. Далее будем перебирать функции из k-окрестности (n, m). Последним пунктом в процессе перебора является нахождения функции, у которой нелинейность выше, чем у предыдущей или при одинаковом значении нелинейности в матрице Уолша – Адамара меньше количество максимальных по модулю значений W_{max}, примем ее за новую функцию F и возвратимся к перебиранию из k-окрестности (n, m).

Алгоритм восхождения на вершину остановится только тогда, когда он перепробует все функции и если не найдет улучшений функции F, то алгоритм завершит свою работу, потому что нелинейность функции ограничена сверху.

Рассмотрев все эвристические алгоритмы можно сделать вывод, что не все алгоритмы подходят ко всем задачам. Реализовывать наши алгоритмы мы

будем в MATLAB, поэтому мы взяли генетический алгоритм, т.к. для его реализации мы можем использовать специальный модуль и алгоритм имитации отжига, потому что он быстрее, чем генетический алгоритм и имеет меньший разброс по значениям.

2.5 Реализация генетического алгоритма

В этой части раздела мы реализуем генетический алгоритм, который является алгоритмом из пакета Global Optimization Toolbox программного комплекса MATLAB. Модуль Genetic Algorithm Tool встроен в пакет и он поможет нам без проблем реализовать генетический алгоритм. Этот модуль является более подходящим благодаря тому, что он наиболее удобен.

Реализуем механизм работы алгоритма через использование комплекта Genetic Algorithm Tool.

Для функционирования модуля необходима тестовая выработка библиографических данных и их признаковое описание.

Данный модуль взаимодействует с модулем обучения, оценки и визуализации, создавая итоговое описание признаков классов для дальнейшей подготовки и использования модели классификации на основе представленного набора признаков. Также функциональный модуль использует модуль классификации и преобразования у модуля обучения, а также оценки визуализации.

Для запуска пакета GAT следует в командной строке MATLAB выполнить команду gatool. После этого запустится пакет генетических алгоритмов и на экране появится основное окно утилиты.

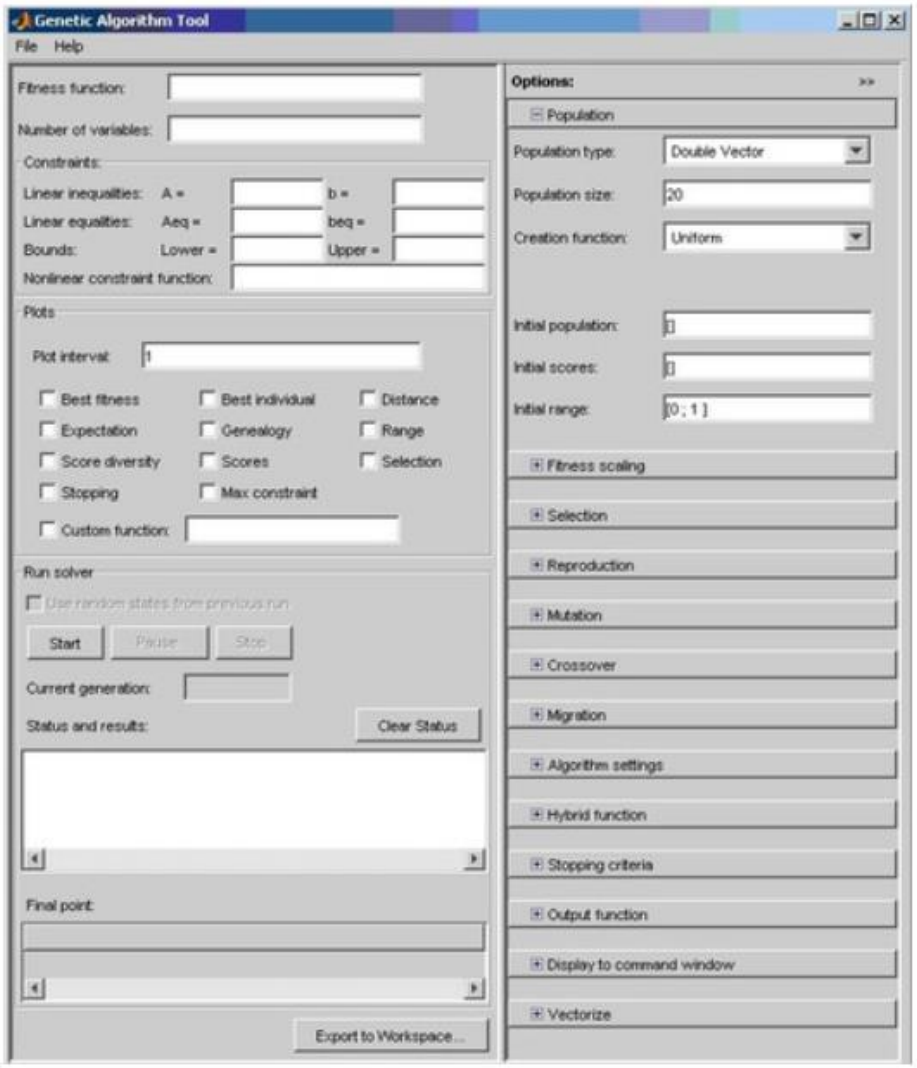

Рисунок 4 – Запуск пакета GAT

В поле Fitness function указывается функция оптимизации в виде @fitnessfun, где fitnessfun.m – название М-файла, в котором предварительно следует описать оптимизируемую функцию. На всякий случай отметим, что М-файл создается в среде MATLAB через меню File->New->M-File. Пример описания некоторой функции my_fun в М-файле:

function z = my_fun(x)

*z = x(1)^2 — 2*x(1)*x(2) + 6*x(1) + x(2)^2 — 6*x(2);*

Нам следует установить параметр Crossover fraction в 0 – это нужно для того, чтобы проконтролировать работу генетического алгоритма без включения операции кроссовера.

На основе данных из функции масштабирования, вкладка Selection позволяет нам выбрать оператор отбора особей родителей.

«В качестве доступных для выбора вариантов оператора отбора предлагаются следующие:

- Tournament – случайно выбирается указанное число особей, среди них на конкурсной основе выбираются лучшие;
- Roulette – имитируется рулетка, в которой размер каждого сегмента устанавливается в соответствии с его вероятностью;
- Uniform – родители выбираются случайным образом согласно заданному распределению и с учетом количества родительских особей и их вероятностей;
- Stochastic uniform – строится линия, в которой каждому родителю ставится в соответствие ее часть определенного размера (в зависимости от вероятности родителя), затем алгоритм пробегает пот линии шагами одинаковой длины и выбирает родителей в зависимости от того, на какую часть линии попал шаг» [4].

Детализация того, как происходит создание новых особей, отображена на вкладке Reproduction, а вот гарантия на количество особей, которые перейдут к следующему поколению – пункт Elite cout. В пункте Crossover fraction указывается доля особей, которые создаются путем скрещивание. Оставшаяся часть формируется путем мутации.

Stopping criteria – пункт, который находится во вкладке критерия, отвечающий за задание состояния, когда алгоритм совершит остановку.

Plot Functions – вкладка позволяет выбирать задаваемые наборы данных, которые выводятся в ходе работы алгоритма, и показывает, насколько корректно выполняется работа и конкретные достигаемые алгоритмом результаты.

«Важными и используемые параметры для отображения являются:

- Best fitness – вывод наилучшего значения оптимизируемой функции для каждого поколения;

- Best individual – вывод наилучшего представителя поколения при наилучшем оптимизационном результате в каждом из поколений;
- Distance – вывод интервала между значениями особей в поколении.

Условия остановки алгоритма. Для определения условий остановки в генетическом алгоритме используются следующие пять условий:

- Generations – алгоритм останавливается тогда, когда число поколений достигает некоего заданного значения Generations;
- Time limit – алгоритм останавливается по истечению некоего заданного времени в секундах Time limit;
- Fitness limit – алгоритм останавливается тогда, когда значение функции пригодности для наилучшей точки для текущего семейства будет меньше или равно Fitness limit;
- Stall generations – алгоритм останавливается в случае, если нет улучшения для целевой функции в последовательности следующих друг за другом поколений длиной Stall generations;
- Stall time limit – алгоритм останавливается в случае, если нет улучшения для целевой функции в течение интервала времени в секундах равного Stall time limit.

Основными параметрами ГА являются:

- вероятность мутации;
- точность получения результата;
- количество итераций алгоритма или количество поколений;
- размер популяции». [12]

Составим таблицу вероятности кроссовера. С помощью таблицы можно выявить отношение текущего населения к следующему поколению, которое было создано операцией кроссовера.

Таблица 2 – Вероятность кроссовера.

		Размер начальной популяции				
		10	50	100	500	1000
Вероятность кроссовера	1	-8,999 -3,619 -0,613	-8,999 -7,705 -4,693	-8,999 -5,837 -2,835	-8,999 -1,061 1,938	-8,999 -7,292 -4,293
	0,8	-8,999 -7,629 -4,628	-8,999 2,351 5,352	-8,999 -0,332 2,669	-8,999 1,516 4,516	-8,999 6,479 9,48
	0,6	-8,999 4,883 7,883	-8,999 -12,305 -9,306	-8,999 1,957 4,955	-8,999 6,74 9,741	-8,999 -13,111 -10,111
	0,3	-8,999 35,127 38,127	-8,999 0,243 3,245	-8,999 -2,898 0,097	-8,999 7,987 10,988	-8,999 23,533 26,534
	0	-8,999 -25,046 -22,045	-8,999 -57,323 -54,323	-8,999 -22,22 -19,22	-8,999 -7,678 -4,678	-8,999 14,207 17,200

По результатам первого запуска видно, что лучшим результатом является N=500, CF – 0, потому что значение f(x) имеет наименьший разброс, чем значения в других ячейках.

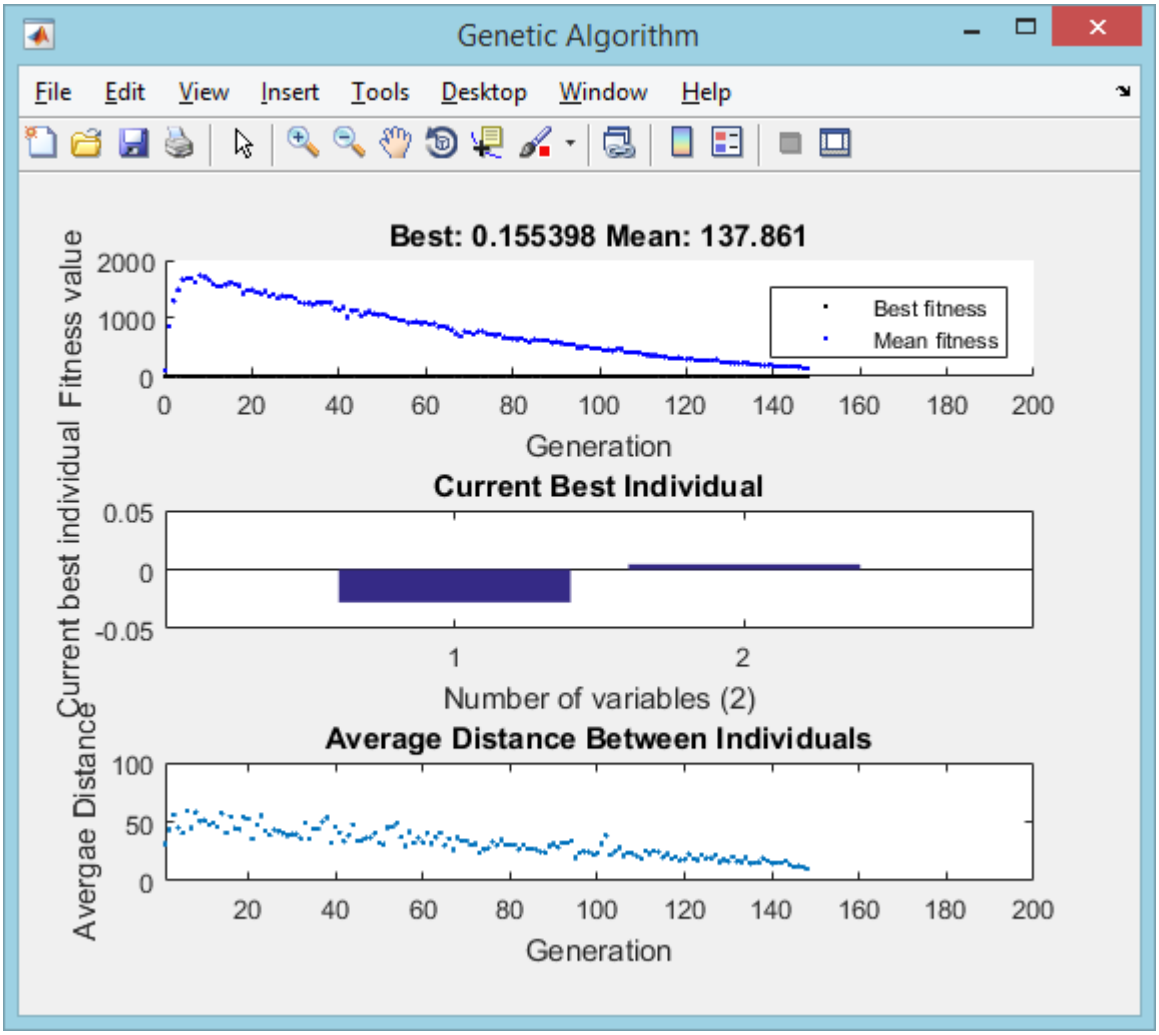

Рисунок 6 – Вывод работы алгоритма

Проанализировав работу алгоритма, мы можем сделать вывод, что при увеличении количества популяции уменьшается количество особей.

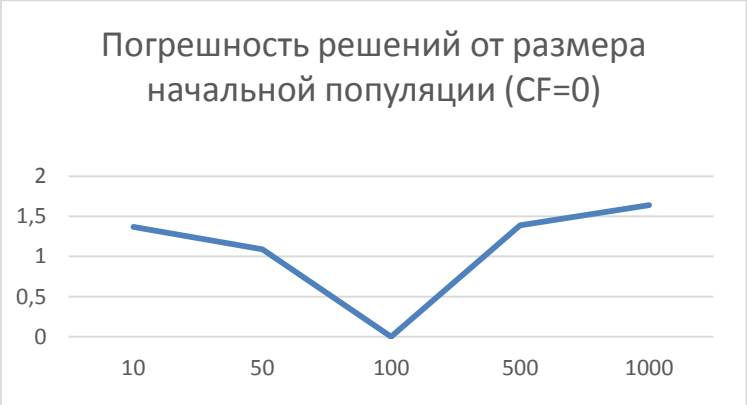

Рисунок 7 – Диаграмма погрешности решений

На рисунке 7 у нас показана погрешность, зависящая от размера популяции. Мы видим, что при популяции равной 100, мы не имеем погрешность, а значит, можем сделать вывод, что лучшим результатом является N=100, CF – 0.

2.6 Реализация алгоритма имитации отжига

Далее реализуем следующий алгоритм. Для этого сначала нам нужно определить описание параметров:

1. Начальное значение управляющего параметра T_0: температура, при которой начинается охлаждение.
2. Параметр затухания управляющего параметра T: выражение для расчета этого ряда температур.
3. Конечное значение управляющего параметра T_f.
4. Длина марковской цепи L_k: количество итераций при любой температуре T.

«Далее опишем наши основные шаги алгоритма:

1. Пусть $T = T_0$. То есть начальная температура отжига, начальное решение x_0 генерируется случайным образом и вычисляется соответствующая мишень E (x0)$E(x_0)$.
2. Пусть T будет следующим T_i в расписании охлаждения.
3. Выполните возмущение в соответствии с текущим решением x_i (метод возмущения может ссылаться на следующий пример), сгенерируйте новое решение x_j, вычислите соответствующее значение целевой функции E (x_j) и вычислите $deltaE = E\ (x_i)$ – E (x_j) .
4. Если $deltaE < 0$, принять новое решение; в противном случае принять вероятность, как указано выше.

5. При температуре *Ti* повторить процесс возмущения и приема *Lk* раз, то есть выполнить 3 и 4.

6. Оцените, достиг ли *T* значения T_f, если да, остановите алгоритм, в противном случае выполните охлаждение» [4].

Примечание генерации новых значений: основными требованиями для создания нового решения являются возможности максимально охватить все области пространства решения. Исходя из этого, новые решения постоянно генерируются при определенной постоянной температуре, вследствие чего есть возможность выхода из текущего региона и последующий поиск других регионов.

«Общие условия сходимости:

1. Начальная температура достаточно высокая.
2. Время теплового равновесия достаточно велико.
3. Конечная температура достаточно низкая.
4. Процесс охлаждения достаточно медленный.

Выбор 3-х параметров:

1. Управляющий параметр Т и начальное значение T_0.
2. Функция распада. Часто используемая функция распада $T_k+1 = a*T_k$, *a* обычно занимает 0,5–0,99’.
3. Длина цепи Маркова. Принцип выбора длины цепи Маркова следующий: при условии, что функция затухания управляющего параметра *T* была выбрана, L_k имеет возможность достичь квазисбалансированного значения для каждого значения управляющего параметра *T*. По опыту, для простых случаев $L_k = 100n$, где *n* – размер проблемы» [5].

На рисунке 8 показана работа алгоритма.

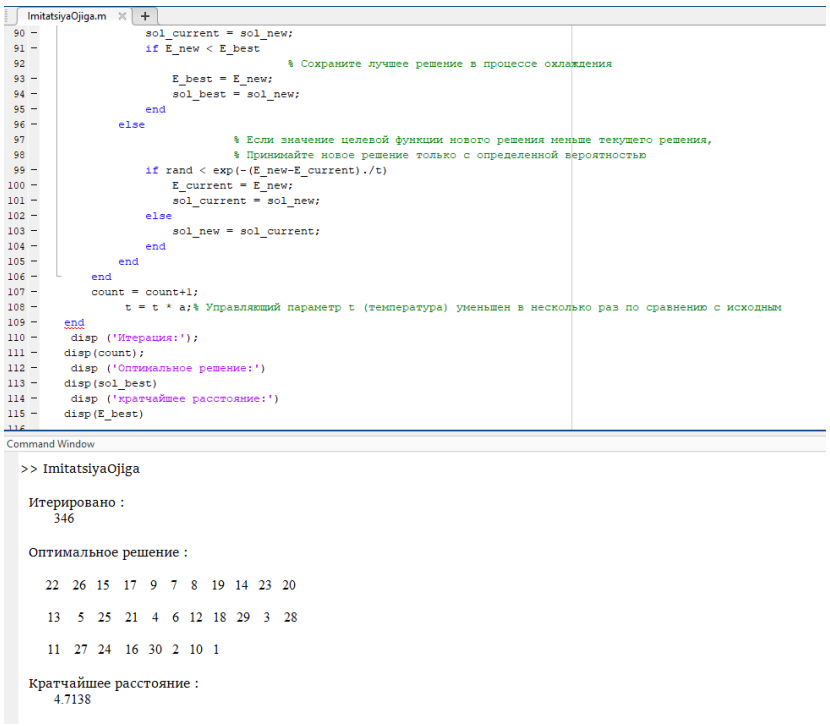

Рисунок 8 – Реализация и вывод работы алгоритма

Так как алгоритм может работать не всегда корректно, необходимо выполнить несколько итераций и, проанализировав эти графики, найти самый корректный вывод программы. Так, например, на рисунке 9 показана работа алгоритма, где изображен не корректный вывод графика. А уже на рисунке 10 результат графика выглядит куда эффективнее.

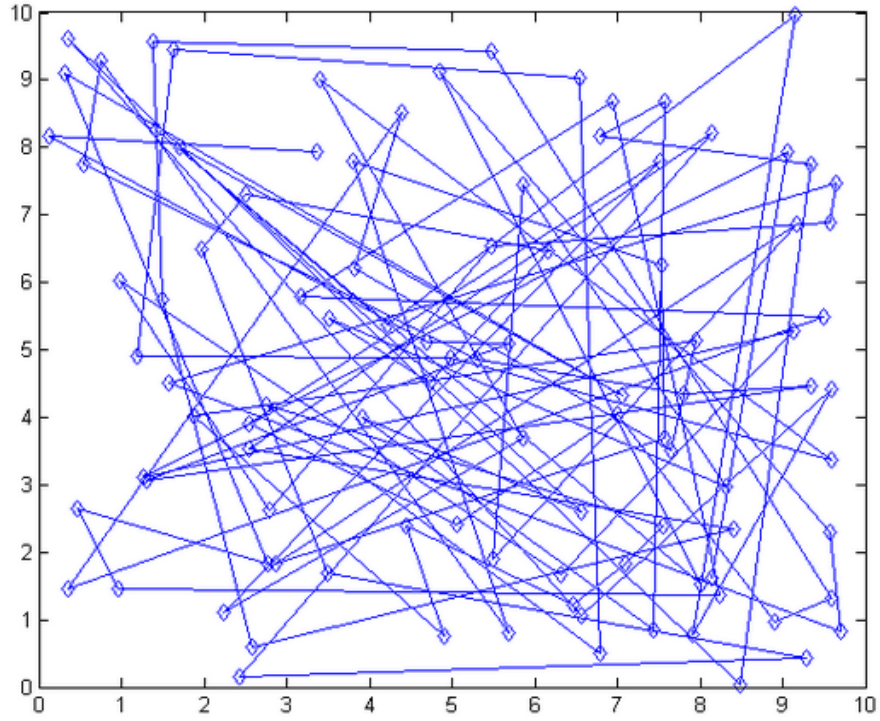

Рисунок 9 – Не корректный вывод графика

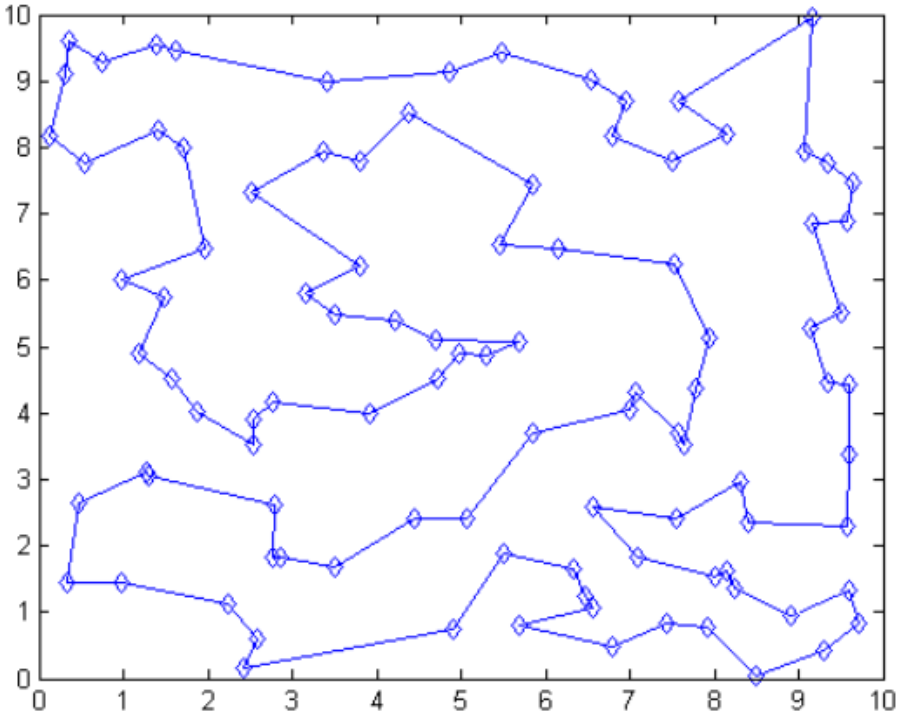

Рисунок 10 – Корректный вывод графика

Выводы по разделу

Во втором разделе мы рассмотрели некоторые виды эвристических алгоритмов, которые применяются для решения задачи поиска максимально нелинейных векторных булевых функций.

Математически описали каждый из видов эвристических алгоритмов, привели их блок-схемы. Из всех алгоритмов рассмотренных в данном разделе мы выбрали для реализации алгоритм имитации отжига и генетический алгоритм.

Алгоритм имитации отжига и генетический мы реализовали для решения нашей задачи. После реализации мы получили окончательные результаты, которые отразим в третьем разделе для дальнейшего их сравнения.

3 Тестирование

3.1 Сравнительный анализ выбранных алгоритмов

В данном разделе проведем сравнительный анализ двух алгоритмов а именно, генетического и имитации отжига. Оба алгоритма будем запускать в MATLAB.

Проанализируем работу алгоритмов для конкретных значений n и m и сравним их по следующим параметрам:

«Max_N_f – максимум достигнутый алгоритмом нелинейности;

$Num\_\max\_N_f$ – число функций с максимально-достигнутой нелинейностью;

EN_f – средняя нелинейность;

$Etime$ – среднее время работы алгоритма (в секундах)» [8].

Все вышеупомянутые параметры будут входить в состав таблицы для сравнения. Также все программы были написаны на одном языке программирования MATLAB и выполнялись на одном компьютере.

Если мы будем увеличивать некоторые управляемые параметры, результат работы по Max_N_f, $Num\_\max\_N_f$, EN_f улучшиться, но при этом время работы алгоритмов может сильно вырасти. Поэтому мы выделим главный критерий: максимум достигнутый алгоритмом нелинейности при умеренном времени работы.

3.2 Результат работы алгоритмов и их сравнение

Для более корректного результата работы будем вводить разные значения для n и m функций. Если алгоритм найдет булеву функцию с нелинейностью выше, чем ее нижняя граница, можно сделать вывод о том, что мы смогли добиться улучшения границ нижней максимальной нелинейности.

Будем вводить следующие значения для 2-х алгоритмов, чтобы проверить: (n = 3, m = 2)(n = 6, m = 7)(n = 10, m = 3), при этом размер тестовой статистики будет равен 100.

Первыми вводим данные n = 3 , m = 2.

Таблица 3 – Результат работы алгоритма при n =3 , m =2

Алгоритмы	Max_N_f	$Num\_\max\_N_f$	EN_f	$Etime$
Генетический алгоритм	3	100	3	4.26
Алгоритм имитации отжига	3	100	3	0.71

По результату работы алгоритмов, мы видим, что алгоритм имитации отжига работает в разы быстрее, чем генетический алгоритм, причем они имеют одинаковые результаты по колонкам: достигнутый максимум, число функций с максимальным значением и средняя нелинейность. Исходя из этой таблицы, можно сделать вывод:

получив данные при N (размер популяции) = 100, n = 3, m = 2 или N = (3,2), мы можем сказать, что реализованные алгоритмы работают не лучше, чем поиск случайных генераций, все значения приведены в таблице 3.

Вторыми вводим данные n = 6 , m = 7.

Таблица 4 – Результат работы алгоритма при n = 6 , m = 7

Алгоритмы	Max_N_f	$Num\_\max\_N_f$	EN_f	$Etime$
Генетический алгоритм	12	1	11.01	23.38
Алгоритм имитации отжига	11	95	10.95	9.46

Исходя из результатов работы алгоритмов стоит выделить, что алгоритм имитации отжига работает в разы быстрее, чем генетический алгоритм. Однако, генетический алгоритм работает эффективнее, потому что среднее значение нелинейности больше. Исходя из этой таблицы, можно сделать вывод:

получив данные при N = 100, n = 6, m = 7 или N = (6,7), мы видим, что смогли превзойти максимальную линейность, а значит, оба алгоритма работают лучше, чем поиск случайных генераций. Все значения приведены в таблице 4.

Третьими вводим данные n =10 , m =3.

Таблица 5 – Результат работы алгоритма при n =10 , m =3

Алгоритмы	Max_N_f	$Num\_\max\_N_f$	EN_f	$Etime$
Генетический алгоритм	261	5	259.2	84.48
Алгоритм имитации отжига	258	79	257.61	57.66

По результату работы алгоритмов, мы видим, что алгоритм имитации отжига работает в разы быстрее, чем генетический алгоритм, но генетический алгоритм работает эффективнее, потому что среднее значение нелинейности больше, но также можно сделать вывод, что алгоритм имитации отжига имеет меньшей разброс в максимальной нелинейности [256;258], а генетический алгоритм [257;261].

Исходя из этой таблицы, можно сделать вывод:

получив данные при N = 100, n = 10, m = 3 или N = (10,3), в этом случае, алгоритмы не достигли максимальной нелинейности, но они все еще работают лучше, чем поиск случайных генераций. Все значения приведены в таблице 5.

Мы проверили работу алгоритмов при разных входных данных, в таблицах наглядно показана статистика работы выбранных алгоритмов.

Вывод по работе алгоритмов: в большинстве запусков программы с разными входными данными алгоритмы работают лучше, чем поиск случайных генераций. Лучшим алгоритмом среди тестируемых можно назвать алгоритм имитации отжига, так как он показал лучшее время работы в некоторых случаях и более точный результат, а именно эти критерии для нас являются самыми важными в работе.

Выводы по разделу

В третьем разделе мы протестировали работу алгоритмов по всем определяемым нами значениям (n = 3, m = 2)(n = 6, m = 7)(n = 10, m = 3). Также мы сравнили работу генетического алгоритма и алгоритма имитации отжига, все полученные результаты были отражены в таблицах. Мы описали все значения необходимые для работы алгоритмов и по каждому запуску программы сделали выводы.

Заключение

В данной квалификационной выпускной работе мы исследовали предметную область и сформулировали задачу максимально нелинейных векторов булевых функций, для этого рассмотрели основные понятия и ввели необходимые определения, леммы и теоремы. После уточнили границы максимальной нелинейности векторных булевых функций, которые помогли нам определить максимальные значения, чтобы решение задачи было более корректным.

Далее, в ходе исследования алгоритмов мы рассмотрели некоторые виды эвристических алгоритмов решения поставленной задачи. Выбранные нами алгоритмы были успешно реализованы на языке MATLAB. Мы протестировали работу алгоритмов по всем определяемым нами значениям (n = 3, m = 2)(n = 6, m = 7)(n = 10, m = 3) и можно отметить что алгоритм имитации отжига работает в разы быстрее, чем генетический алгоритм, но при этом генетический алгоритм является более эффективным, так как среднее значение нелинейности больше. Также можно сделать вывод, что алгоритм имитации отжига имеет меньшей разброс в максимальной нелинейности [256;258], а генетический алгоритм [257;261].

На основе приведенного сравнения двух алгоритмов можно сделать вывод о том, что алгоритм имитации отжига является более быстрым и имеет меньший разброс по сравнению с генетическим алгоритмом.

Все поставленные задачи выполнены, и цель выпускной квалификационной работы была достигнута.

Список используемой литературы

1. Агафонова И.В. Алгебраическая нормальная форма булевой функции и бинарное преобразование Мёбиуса, 2013.
2. Вишняков П.О. Планирование маршрутов с использованием модифицированного метода «ближайшего соседа» //Математические методы в технике и технологиях - ММТТ. 2014. No 6 (65). С. 63-67.
3. Гладков Л.А., Гладкова Н.В. Особенности использования нечетких генетических алгоритмов для решения задач оптимизации и управления // Известия ЮФУ. Технические науки. 2009. No 4 (93). С. 130-136.
4. Курбатова Н.В., Пустовалова О.Г. Основы MatLab в примерах и задачах. - Ростов-на-Дону: Южный федеральный университет, 2017. - 146 с.
5. Лопатин А. Метод отжига. Стохастическая оптимизация в информатике, 1:133–149, 2005.
6. Марченков С. С. Основы теории булевых функций. - 1 изд. - Москва: Физматлит, 2014. - 134 с.
7. Н. В. Тимофеева Линейная алгебра. Современная алгебра. - Ярославль: ЯрГУ, 2017. - 134 с.
8. Н. Ю. Прокопенко МЕТОДЫ ОПТИМИЗАЦИИ. - Нижний Новгород: ННГАСУ, 2018. - 118 с.
9. Панченко Т. В. Генетические алгоритмы. - Астрахань: Астраханский университет, 2007. - 86 с.
10. Писарук Н. Н. Исследование операций. - Минск: БГУ, 2015. - 298 с.
11. Пожидаев М. С., Сбалансированная эвристика для решения задачи маршрутизации транспорта с учетом грузоподъемности [Text] / Ю. Л. Костюк, М. С. Пожидаев // Вестник ТГУ. УВТиИ. — 2010. — No 3.

12. Скиена С.С Алгоритмы руководство по разработке. - 2-е изд. - Санкт-Петербург : БХВ-Петербург, 2011. - 713 с.

13. Смоленцев Н. К. MATLAB. Программирование на C++, C#, Java и VBA. - 2-е изд. - Москва: ДМК Пресс, 2015. - 495 с.

14. Токарева Н. Н. Нелинейные булевы функции: бент-функции и их обобщения. Издательство LAP LAMBERT Academic Publishing (Saarbrucken, Germany), 2011. - 146 с.

15. Ященко В.В., Логачев О.А., Сальников А.А. Булевы функции в теории кодирования и криптологии. - Москва: МЦНМО, 2004. - 468 с.

16. Carlet C. *Boolean Methods and Models*, chapter Vectorial boolean functions for cryptography. Cambridge University Press, 2008.

17. Christofides N., The vehicle routing problem. In N. Christofides, A.Mingozzi, P. Toth, C. Sandi, editors [Text] / N. Christofides, A. Mingozzi, P. Toth // Combinatorial Optimization. — Wiley, Chichester, 1979. — P. 315–338.

18. Gromicho J., van Hoorn J.J., Kok A.L., Schutten J.M.J., Restricted dynamic programming: A flexible framework for solving realistic VRPs [Text] / J.Gromicho, J.J.van Hoorn, A.L. Kok, J.M.J. Schutten // Vrije Universiteit, Amsterdam, The Netherlands – 2011. – P. 902-909.

19. Maitra S. Boolean functions on odd number of variables having nonlinearity greater than the bent concatenation bound. Boolean Functions in Cryptology and Information Security (NATO ASI Zvenigorod, 2007), pages 173–182, 2008.

20. Haimovich M., Bounds and heuristics for capacitated routing problems [Электронный ресурс] / M. Haimovich, A.H.G. Rinnooy Kan // Mathematics of Operations Research. — 1985. — No 10. — P. 527–542.

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