ISSN 0132-3474

Номер 6 Ноябрь - Декабрь 2024


# ПРОГРАММИРОВАНИЕ


НАУКА
1727

# СОДЕРЖАНИЕ

**Номер 6, 2024**

## ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ



## АНАЛИЗ ДАННЫХ

# Contents

**No. 6, 2024**

## INFORMATION SECURITY



## DATA ATALISYS

*ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 3–11*

## ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

*УДК 004.056.55*

# ВЫСОКОСКОРОСТНОЙ АЛГОРИТМ СКАЛЯРНОГО УМНОЖЕНИЯ ДЛЯ ПРОЕКТИРОВАНИЯ НЕЙРОННЫХ СЕТЕЙ, СОХРАНЯЮЩИХ КОНФИДЕНЦИАЛЬНОСТЬ

**© 2024 г. М. А. Лапина[a,*], Е. М. Ширяев[a,**], М. Г. Бабенко[a,***], И. Истамов[b,****]**

[a] *Северо-Кавказский центр математических исследований, Северо-Кавказский федеральный университет*
*355017 Ставрополь, ул. Пушкина, 1, Россия*

[b] *Самаркандский государственный университет имени Шарофа Рашидова*
*140104 Самарканд, Университетский бульвар, 15, Узбекистан*

**E-mail: mlapina@ncfu.ru*
***E-mail: eshiriaev@ncfu.ru*
****E-mail: mgbabenko@ncfu.ru*
*****E-mail: istamovismoilzoda@gmail.com*

Поступила в редакцию 10.07.2024 г.
После доработки 16.07.2024 г.
Принята к публикации 26.07.20204 г.

В силу юридических ограничений либо ограничений, связанных с внутренней информационной политикой компаний, зачастую бизнес не доверяет конфиденциальную информацию публичным облачным провайдерам. Одним из механизмов, позволяющих обеспечить безопасность конфиденциальных данных в облаках, является гомоморфное шифрование. Для проектирования решений, использующих нейронные сети, в данных условиях используются нейронные сети, сохраняющие конфиденциальность. Они эксплуатируют механизм гомоморфного шифрования, позволяя таким образом обеспечить безопасность коммерческой информации в облаке. Основным сдерживающим фактором использования нейронных сетей, сохраняющих конфиденциальность, является большая вычислительная и пространственная сложность алгоритма скалярного умножения, который является базовым для вычисления математической свертки. В работе предлагается алгоритм скалярного умножения, который позволяет уменьшить пространственную сложность с квадратичной до линейной, а также уменьшить время вычисления скалярного умножения в 1.38 раза.

*Ключевые слова*: матричные операции; искусственные нейронные сети; полностью гомоморфное шифрование; CKKS; TenSEAL; нейронные сети, сохраняющие конфиденциальность

**DOI:** 10.31857/S0132347424060012, **EDN:** DZISPQ

## 1. ВВЕДЕНИЕ

Методы искусственного интеллекта (ИИ) [1] набирают все большую популярность в последние годы. Если в 2000-х гг. ИИ зачастую интересовались только исследовательские круги, то в последние десятилетия ИИ набирает популярность во всех областях человеческой деятельности. Если проанализировать научно-технические достижения, то можно отметить следующее. На рост популярности методов ИИ в большей степени повлияло развитие децентрализованных вычислительных архитектур, в том числе облачных вычислений, аппаратных ускорителей, а также общая тенденция увеличения вычислительной мощности устройств. Методы ИИ сегодня применяются в производстве для повышения эффективности автоматизации, в медицине и финансовых структурах для анализа больших данных и т. д. С ростом популярности языковых моделей и запуском GPT с открытым исходным кодом методы ИИ охватили еще большее количество сфер человеческой деятельности [2].

Однако, как и в конце XX в., когда появился Интернет/Всемирная паутина, методы ИИ стали предметом различных дискуссий [3], как с точки зрения закона и законотворчества, так и с точки зрения безопасности данных. Задачи, решаемые ИИ, как правило, сопряжены с обработкой больших объемов данных, а в случае с теми же языковыми моделями – очень больших объемов данных. Большие данные [4] могут содержать информацию с ограниченным доступом, например, если ИИ используется компаниями для

обработки данных пользователей, в медицинской организации – персональные данные пациентов, муниципальной/государственной – данные граждан, внутренние документы, в финансовых организациях – данные клиентов, информация о счетах, биржевые котировки и т. д. Все вышеперечисленные данные зачастую являются конфиденциальными и охраняются законом, например, в Российской Федерации это Федеральный закон от 27 июля 2006 г. № 152-ФЗ “О персональных данных”, который направлен на усиление контроля за обработкой и распространением личной информации граждан [5]. Если методы ИИ используются внутри закрытой сети, вопросы безопасности можно решить стандартными методами, но создание и поддержка высокопроизводительной закрытой сети требует больших вычислительных ресурсов и финансирования. Поэтому зачастую эффективнее обратиться к поставщику услуг облачных вычислений, что и делает большинство компаний. При аренде вычислительных мощностей сеть становится общедоступной, что создает риск компрометации конфиденциальных данных. Хотя поставщики облачных услуг гарантируют безопасность данных, хранящихся в облаке, гарантировать безопасность вычислений на данный момент практически невозможно, поскольку можно хранить данные в зашифрованном виде, но не обрабатывать.

Таким образом, возникает проблема обработки конфиденциальных данных ИИ в публичных сетях. В качестве решения проблемы можно рассмотреть такой криптографический примитив, как полностью гомоморфное шифрование (ПГШ) [6], он позволяет выполнять гомоморфные операции сложения и умножения над зашифрованными данными. Этого достаточно, например, для работы нейронных сетей (НС) [7], когда нам нужно обрабатывать входные данные с помощью уже обученных нейронов. В этой области также существуют нерешенные проблемы, например, с аппроксимацией некоторых функций активации. Кроме того, хотя для матриц операции можно реализовать на основе сложения и умножения (вычитание реализуется как сложение положительных и отрицательных чисел), из-за особенностей схем ПГШ возникает большая избыточность данных, что приводит к неэффективной работе конфиденциальных НС. Цель данной работы – исследовать матричное умножение в контексте гомоморфного шифрования для повышения эффективности использования памяти, разработать новый алгоритм и протестировать его эффективность для конфиденциальных НС.

Работа организована следующим образом: в разделе 2 рассматриваются конфиденциальные НС и методы их организации, в разделе 3 представлено исследование конфиденциального умножения матриц, в разделе 4 анализируются полученные результаты на основе экспериментального исследования, в разделе 5 подводятся итоги проделанной работы и описываются будущие работы.

## 2. НЕЙРОННЫЕ СЕТИ, СОХРАНЯЮЩИЕ КОНФИДЕНЦИАЛЬНОСТЬ

### *2.1. Искусственная нейронная сеть*

Математическая модель искусственных нейронных сетей представляет собой линейную композицию взаимосвязанных нейронов с нелинейными функциями активации (рис. 1).

Входной слой обрабатывает входные данные, скрытый слой выполняет вычисления, а выходной слой отвечает за выходную информацию. Искусственная НС, как и биологическая НС, работает посредством активации нейронов, т. е., когда значение внутри нейрона достигает функции активации, следующему нейрону передаются ее значение. Каждый нейрон имеет свое базовое значение (вес). Тогда состояние нейрона $S$ можно описать как

$$S = \sum_{i=0}^{n} x_i w_i,$$

где $x_i$ – значение $i$-го входа нейрона; $w_i$ – вес $i$-го входа; $n$ – количество входов нейрона. Передача значений следующему нейрону осуществляется посредством функции активации:

$$Y = f(S),$$

где $f(S)$ – функция активации. Например, сигмоидальная функция активации [8], определяется как

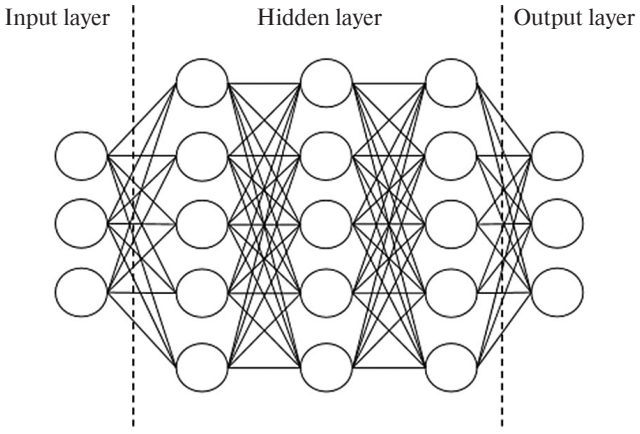



**Рис. 1.** Модель искусственной нейронной сети.

$$f(S) = \frac{1}{1 + e^{-aS}}. \quad (1)$$

С математической точки зрения, НС – это многопараметрическая задача нелинейной оптимизации, где веса нейронов скрытого слоя представляют собой параметры, а нейроны выходного слоя – ограничения. Чтобы нейронная сеть работала правильно, ее нужно обучить. Обучение происходит путем изменения внутренних значений весов нейронов. Существует несколько способов обучения, как с учителем, так и без него. Самый популярный способ обучения с учителем – минимизация ошибок. На основе этого типа обучения строятся сети с обратным распространением, которые используются для поиска закономерностей, прогнозирования и качественного анализа. Анализируя рис. 1, можно заметить, что если ввести операцию умножения матриц, то можно повысить эффективность обработки данных, как это и делается в большинстве случаев. Чтобы строить НС, сохраняющие конфиденциальность, необходимо ввести понятие криптографического примитива ПГШ.

### *2.2. Полностью гомоморфное шифрование и схема CKKS*

Полностью гомоморфное шифрование (ПГШ) – криптографический примитив, развивающий идеи гомоморфного шифрования (ГШ). ГШ позволяет выполнять гомоморфное сложение или гомоморфное умножение над зашифрованным текстом. Примерами ГШ являются асимметричные шифры, такие как RSA [9], ElGamal [10] и др. Криптографы еще в 1980-х гг. предполагали, что полностью гомоморфное шифрование, когда шифр позволяет выполнять и гомоморфное сложение, и гомоморфное умножение, возможно. Первая схема ПГШ была представлена Джентри в его работе 2009 г. [6]. Однако эта схема не была эффективной и обрабатывала двоичные биты с помощью логических операций довольно долго (по сравнению с современными схемами), кроме того, схема имела большие ограничения на количество допустимых операций (количество операций, после которых сообщение может быть восстановлено). За следующие 15 лет сам Джентри [11–13] и его последователи [14–17] разработали новые схемы ПГШ, которые работают с целыми числами, выполняются быстрее и ослабляют ограничения на вид и количество операций.

Следующим шагом в истории ПГШ стала схема CKKS (первоначально HEaaN), которая позволяет обрабатывать рациональные числа [18]. CKKS – это система гомоморфного шифрования, предназначенная для эффективного выполнения приближенных арифметических операций над зашифрованными данными. Она идеально подходит для вычислений с вещественными или комплексными числами над полем $C^{N/2}$. Пространство открытых текстов и пространство шифротекстов имеют одну и ту же область

$$Z_Q[X]/(X^N + 1),$$

где $N$ – чаще всего степень двойки.

Пакетное кодирование $C^{\frac{N}{2}} \leftrightarrow Z_Q[X]/(X^N + 1)$ отображает массив комплексных чисел в многочлен со свойством: $decode(encode(m_1) \otimes encode(m_2)) \approx m_1 \odot m_2$, где $\otimes$ – покомпонентное умножение, а $\odot$ – негациклическая свертка.

Схема CKKS работает по стандарту [19], который содержит рекомендуемые параметры для 128-битных ключей ГШ троичной формы $s \in_u \{-1, 0, 1\}^N$. Шифрование в CKKS осуществляется путем вычисления полиномов Лагранжа в поле комплексных чисел.

Схема использует приближенную арифметику для построения шифротекстов. Рассмотрим заданную арифметику. В начале мы фиксируем основание $p > 0$ и модуль $q_0$, причем $q = pq_0$ при $0 < l \le L$. Целое число $p$ будет использоваться в качестве основы для масштабирования в приблизительных расчетах. В качестве параметра безопасности $\lambda$ выбирается такой параметр, что $M = M(\lambda, q_L)$ для полиномиального кольца. При границах $< l \le L$ уровня шифротекста $l$ определяется вектор в $\mathcal{R}_{q\ell}^{k}$ для фиксированного целого числа $k$.

1. Генерация ключей: процесс шифрования начинается с генерации ключей: открытого ключа $pk$ и закрытого ключа $sk$. Закрытый ключ используется для дешифровки данных, а открытый – для их шифрования.
2. Шифрование: чтобы зашифровать вектор открытого текста $x$, выполняются следующие действия:
   - Дополнение: вектор $m(x)$ дополнен нулями, длина вектора равна заданной степени двойки $N$;
   - Кодирование: вектор открытого текста $x$ кодируется в полином открытого текста $m(x)$, который является полиномиальным представлением сообщения;
   - Гомоморфное шифрование: полином $m(x)$ шифруется с помощью $pk$ для получения

полинома $c(x)$ шифротекста, при этом контролируется уровень шума шифротекста – количество специально вносимых ошибок $e$, удовлетворяющее $|e|_{\infty}^{can} \le e_{Max}$, для выражения $c, sk = m + e_{Max}\left(mod\, q_L\right)$.

3. Дешифрование: для дешифровки полинома $c(x)$ шифротекста выполняются следующие действия:
   - Гомоморфное дешифрование: полином $c(x)$ дешифруется с помощью секретного ключа для получения полинома $m(x) \leftarrow c, sk\left(mod\, q_l\right)$ в пространстве открытых текстов;
   - Декодирование: для получения исходного текстового вектора $x$ текстовый полином $m(x)$ снова преобразуется из полинома в полином сообщений.
4. Гомоморфные операции: CKKS поддерживает несколько приближенных арифметических операций над зашифрованными данными, включая сложение и умножение. Гомоморфное сложение и умножение можно выполнять в пространстве шифротекстов без необходимости их дешифровки:
   - Гомоморфное сложение: при получении двух шифротекстов $c_1(x)$ и $c_2(x)$, представляющих зашифрованные значения $m_1(x)$ и $m_2(x)$ соответственно, выполняется гомоморфное сложение путем сложения соответствующих коэффициентов по модулю: $c(x) = c_1(x) + c_2(x)$, при этом ошибки $e_1$ и $e_2$ также суммируются;
   - Гомоморфное умножение: при наличии двух шифротекстов $c_1(x)$ и $c_2(x)$, представляющих зашифрованные значения $m_1(x)$ и $m_2(x)$ соответственно, гомоморфное умножение выполняется путем преобразования зашифрованных полиномиальных текстов для последующего покомпонентного умножения по модулю исходного модульного текста и обратного преобразования: $c(x) = c_1(x) + c_2(x)$, для умножения выделяются собственные границы ошибок $e_{mult} \in \mathcal{R}$ с $\left|e_{mult}\right|^{can_{\infty}} e_{multMax}(l)$, где $e_{multMax}(l)$ заданная константа.

Как сложение, так и умножение приводят к увеличению ошибки аппроксимации $e$, схема CKKS позволяет дешифровать данные, если ошибка находится в определенных пределах. При использовании схемы CKKS важно контролировать рост ошибки, который зависит от количества операций и их порядка. Учитывая особенности арифметики, умножение вносит большую погрешность. Различные программные реализации схемы CKKS предлагают разные способы контроля уровня ошибки.

### *2.3. Нейронные сети, сохраняющие конфиденциальность*

Интерес к нейронным сетям, сохраняющим конфиденциальность (НССК), возник несколько лет назад. В своем обзоре [20] авторы исследуют эту концепцию с теоретической точки зрения, рассматривая основные задачи и проблемы, с которыми сталкиваются исследователи при построении НССК на основе ПГШ. В основе НССК лежит концепция Machine Learning as a Service (MLaaS) [21], которая схожа с концепциями облачных вычислений [22–24]. В статье даны определения операций ПГШ, включая проблемные операции. Помимо умножения матриц, также упоминается бутстраппинг [25], который используется для увеличения количества допустимых операций умножения. Также описаны инструменты для работы с ПГШ [26–28], включая используемые в НССК [29]. Проблема ускорения НССК, работающих в ПГШ, упоминается в статье отдельно. Эта тема также популярна среди исследователей, например, в [30] изучается ускорение операции умножения матриц путем модификации метода Хавели [31]. Однако операции умножения, основанные на этом методе, по-прежнему достаточно ресурсоемки. В следующем разделе подробно рассмотрена операция умножения матриц и способы повышения скорости обработки данных.

## 3. ОПЕРАЦИЯ УМНОЖЕНИЯ МАТРИЦ В ПРИБЛИЖЕННОЙ СХЕМЕ ГОМОМОРФНОГО ШИФРОВАНИЯ

Умножение матриц – базовая операция для многих систем, в том числе и для НС. Рассмотрим ее алгоритм, основанный на умножении квадратных матриц $a$ и $b$ размера $n \times n$:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} \cdot b_{k,j},$$

где $i, j, k \in \overline{1,n}$, $c$ – результат умножения. В открытом виде этот алгоритм довольно прост. Однако в ПГШ его выполнение невозможно, так как мы не можем отдельно обратиться к элементу каждого внутреннего вектора. Рассмотрим подробно метод Хавели [31]. Для выполнения матричного умножения в ПГШ матрицы должны быть закодированы в диагональном представлении. Затем для выполнения умножения необходимо

выполнить несколько операций вращения со вспомогательными матрицами. Рассмотрим пример.

Пусть матрица $A$ размера $m \times m$ представлена в зашифрованном виде $y_0, \ldots, y_{m-1}$, где $y_i = (A_{0,i}, A_{1,i+1}, \ldots, A_{m-1,m+1})$. Сначала, $y_i[j] = A_{j,j+1}$. Далее, кусочное произведение между вектором весов и матрицей $w = vA$, где $v = \{v_0, v_1, \ldots, v_{n-1}\}$ – входной вектор, может быть вычислено как:

$$v_0 = \{x_0, x_1, \ldots, x_{n-1}\} \odot \{y_0, y_1, \ldots, y_{n-1}\} = \{x_0 y_0, x_1 y_1, \ldots, x_{n-1} y_{n-1}\},$$

$$v_1 = \{x_{n-1}, x_0, \ldots, x_{n-2}\} \odot \{y_{n-1}, y_0, \ldots, y_{n-2}\} = \{x_{n-1} y_{n-1}, x_0 y_0, \ldots, x_{n-2} y_{n-2}\},$$

$$\ldots,$$

$$v_{n-1} = \{x_1, x_2, \ldots, x_{n-1}, x_0\} \odot \{y_1, y_2, \ldots, y_{n-1}, y_0\} = \{x_1 y_1, x_2 y_2, \ldots, x_{n-1} y_{n-1}, x_0 y_0\},$$

где $\odot$ – покомпонентное произведение векторов.

Однако есть и другой способ, который подходит для НССК. Возникает вопрос, настолько ли необходим подобный уровень конфиденциальности.

Учитывая тот факт, что в настоящее время невозможно обучить НС в ПГШ за приемлемое время, НС обучается в открытом виде. Справедливо заметить, что значения весов являются открытыми и могут быть общедоступными и существует возможность их компрометации. Тогда нет смысла их шифровать, и мы можем применять их в открытом виде. Тогда можно использовать модифицированный алгоритм, основанный на предыдущем, где входная матрица кодируется в диагональном представлении, а веса представляются в виде вектора. Такое умножение рассматривается как умножение матрицы на скаляр, которое реализуется посредством операций умножения и вращения. Рассмотрим пример.

Пусть матрица $A$ размера $n \times n$ представлена в зашифрованном виде $a_0, \ldots, a_{m-1}$, где $a_i = (A_{0,i}, A_{1,i+1}, \ldots, A_{n-1,n+1})$. Сначала, $a_i[j] = A_{j,j+1}$. Следующее произведение $w = vA$, где $v = \{v_0, v_1, \ldots, v_{n-1}\}$ – входной вектор, может быть вычислено как

$$v_0 = x_0\{a_0, a_1, \ldots, a_{n-1}\},$$

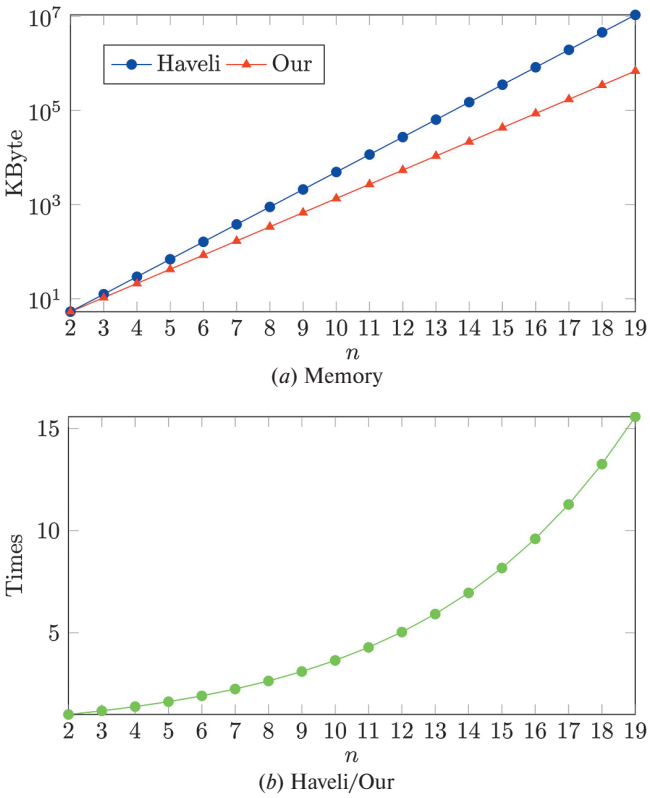
$$v_1 = x_1\{a_{n-1}, a_0, \ldots, a_{n-2}\},$$

$$\ldots,$$

$$v_{n-1} = x_{n-1}\{a_1, a_2, \ldots, a_{n-1}, a_0\}.$$

Этот метод требует $m$ операций вращения, умножения и сложения. Кроме того, вспомогательные матрицы $w$ и $v$ в зашифрованном виде занимают много памяти, как и промежуточные зашифрованные матрицы до получения результата. Из формул видно, что, используя открытые значения, мы не только сокращаем количество операций, но и расход памяти. Для подтверждения выводов было проведено экспериментальное исследование (рис. 2).

Из данных, представленных на рис. 2*a*, можно сделать вывод, что предложенный алгоритм позволяет сократить объем памяти в среднем в 7.89 раза. Линия тренда для данных, представленных на рис. 2*b*, равна $0.0656n^2 - 0.4452n + 2.0912$ с коэффициентом детерминации $R^2 = 0.9925$. Таким образом, можно сделать вывод, что пространственная сложность уменьшилась с $O(n^4)$ до $O(n^2)$ для произведения матриц размера $n \times n$. Учитывая, что для произведения квадратных матриц размера $n \times n$ необходимо вычислить $n^2$ скалярных умножений и $2n^2$ циклических сдвигов векторов, пространственная сложность алгоритма скалярного умножения с зашифрованными данными снижается с $O(n^2)$ до $O(n)$.




(*a*) Memory

(*b*) Haveli/Our

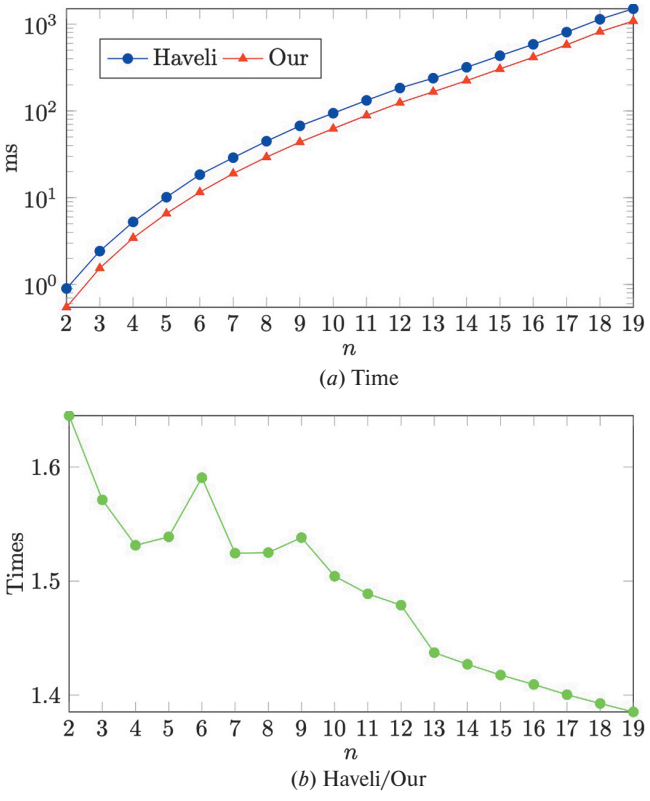
**Рис. 2.** Исследование потребления памяти предлагаемым методом.

Как видно на рис. 2*b*, потребление памяти сократилось с квадратичного закона до линейного. Это дает преимущество в эффективности при работе с HCCK. Однако, учитывая специфику схемы CKKS, необходимо проверить, не повлияли ли внесенные изменения на точность результата. Для этого необходимо построить нейронную сеть и провести исследование. Далее рассмотрим скорость, с которой выполняются вычисления. Стоит отметить, что на рис. 3*a* показана общая скорость выполнения операций, включая шифрование и дешифрование.

Анализируя рис. 3*a*, можно сказать, что предложенный метод выполняет умножение быстрее. Этот эффект достигается как за счет нового подхода к умножению, так и за счет того, что шифрование и дешифрование упрощаются, поскольку умножение выполняется на открытом векторе весов HCCK. Кроме того, мы предлагаем проанализировать соотношение скоростей методов (рис. 3*b*).

Линии тренда зависимости времени от $n$ для алгоритма Хавели $0.0634n^4 - 1.8561n^3 + 20.897n^2 - 88.794n + 118.56$, для предложенного алгоритма $0.0461n^4 - 1.3405n^3 + 14.955n^2 - 63.491n + 84.81$ с коэффициентом детерминации для обеих линий равным $R^2 = 0.9995$ (рис. 3*a*). Асимптотически выигрыш во времени при увеличении $n$ равен

$$\lim_{n\to\infty} \frac{0.0634n^4 - 1.8561n^3 + 20.897n^2 - 88.794n + 118.56}{0.0461n^4 - 1.3405n^3 + 14.955n^2 - 63.491n + 84.81} \approx$$
$$\approx 1.38.$$




(*a*) Time

(*b*) Haveli/Our

**Рис. 3.** Исследование времени вычисления операции умножения матриц.

Время работы алгоритма для произведения квадратных матриц $n \times n$ уменьшилось в среднем в 1.49 раза (рис. 3*b*). С увеличением размера график становится более линейным, что можно объяснить увеличением избыточности, которая зависит от длины вектора, в то время как при малых размерах она зависит как от конструкции вектора, так и от вспомогательных матриц, необходимых для вращения.

В целом можно сказать, что предложенный метод эффективен как с точки зрения потребления памяти, так и скорости вычислений. Стоит отметить, что этот результат достигается за счет снижения конфиденциальности, а именно конфиденциальности весов HCCK, при условии, что веса являются общеизвестными при обучении HCCK в открытом виде.

## 4. ИССЛЕДОВАНИЕ ТОЧНОСТИ

В рамках исследования были проведены эксперименты по обучению и тестированию нейронной сети, а также ее зашифрованной версии на наборе данных MNIST. Целью эксперимента было оценить производительность модели в обычном и зашифрованном режимах, а также изучить влияние гомоморфного шифрования на производительность и точность модели. Аппаратная конфигурация состоит из процессора Intel(R) Xeon(R) CPU E5–2696 v3 с тактовой частотой 2.30 ГГц, 32 ГБ оперативной памяти DDR4 с частотой 2133 МГц и твердотельного накопителя объемом 1 ТБ. Среднее время измерялось путем запуска алгоритмов на платформе 10 000 раз. В ходе эксперимента собирались данные о потерях при обучении и тестировании, а также о точности классификации для каждого класса. Для этого была построена НС на основе следующей математической модели.

Рассмотрим сверточную нейронную сеть (СНС) со следующими слоями и параметрами:

- Входное изображение: $I$, одноканальное изображение.
- Первый сверточный слой ($C_1$): применяет 4 фильтра с размером ядра $7 \times 7$ с шагом 3 и размером 0.
- Первый слой – полносвязный ($F_1$): преобразует упрощенные карты признаков с использованием $H$-нейронов.

- Второй полносвязный слой ($F_2$): сопоставляет скрытый слой с выходным слоем с $O$-нейронами.

Математические операции, выполняемые СНС, выглядят следующим образом:

1. Работа первого сверточного слоя может быть определена как
$$C_1(I) = Conv2d(I, K_1, S_1, P_1),$$
где $K_i = 7 \times 7$ – размер ядра, с шагом $S_1 = 3$ и размером $P_1 = 0$.
2. Выходной сигнал $C_1$ проходит через функцию активации и, возможно, другие операции, такие как объединение или нормализация, после чего сглаживается и поступает в первый слой с полным подключением.
3. Работа первого полносвязанного слоя может быть определена как
$$F_1(X) = XW_{F_1} + b_{F_1},$$
где $X$ – входной вектор для $F_1$; $W_{F_1}$ и $b_{F_1}$ представляют собой веса и смещения $F_1$ соответственно.
4. Работа второго полносвязанного слоя аналогично определяется как
$$F_2(Y) = YW_{F_2} + b_{F_2},$$
где $Y$ – входной вектор $F_2$, полученный из выхода $F_1$; $W_{F_2}$ и $b_{F_2}$ – веса и смещения $F_2$ соответственно. Эта модель описывает структуру СНС, подчеркивая последовательность от обработки на сверточном слое до генерации конечного вывода через полносвязанные слои.

СНС была выбрана в качестве модели потому, что ПГШ обладает свойствами как гомоморфизма, так и автоморфизма, благодаря которым вращение зашифрованных матриц для реализации матричного умножения реализовать достаточного просто. Кроме того, эти свойства позволяют достаточно эффективно выполнять математическую операцию свертки [32]. Далее рассмотрим

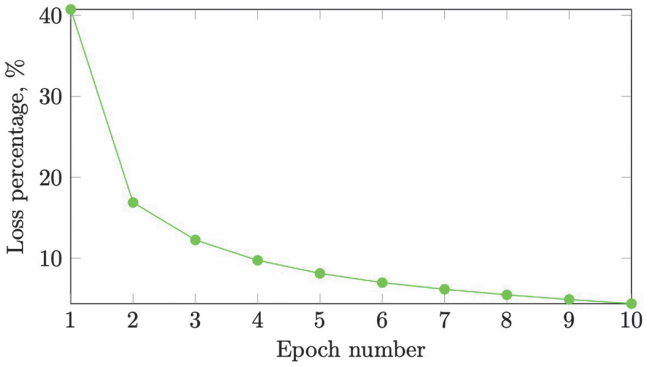



**Рис. 4.** Исследование функции потерь для PPNN.

результаты работы полученной модели. А именно, данные о потерях (рис. 4).

На рис. 4 показана динамика потерь нейронной сети в процессе обучения для 10 эпох. По оси абсцисс откладывается номер теста, равный количеству эпох (от 1 до 10), а по оси ординат – количество потерь. На графике видно уменьшение потерь с каждой последующей эпохой, что свидетельствует об адекватном поведении модели при использовании нового алгоритма обучения.

На рис. 5 показана точность классификации модели на тестовых данных для каждого из 10 классов. По оси абсцисс представлены классы (от 0 до 9), а по оси ординат – процент точности для каждого класса. График помогает наглядно представить, как модель справляется с классификацией различных категорий, выявляя классы, в которых модель работает лучше или хуже.

Результаты экспериментов показывают, что нейронная сеть демонстрирует высокую точность как в обычном, так и в зашифрованном режимах, причем в зашифрованном режиме общая точность несколько повышается. Это говорит о том, что применение гомоморфного шифрования не оказывает существенного негативного влияния на способность модели к классификации. Кроме того, тот факт, что в некоторых классах зашифрованная СНС показывает более

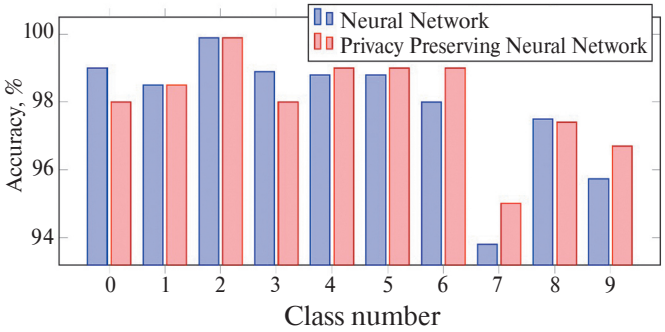



**Рис. 5.** Исследование точности PPNN для различных классов.

точные результаты, требует дополнительных исследований.

## 5. ЗАКЛЮЧЕНИЕ

Результатом исследования по адаптации НССК для работы с ПГШ стали результаты, которые подчеркивают потенциал и ограничения данного подхода. Исследование показывает, что, изменив метод умножения зашифрованных матриц, можно успешно использовать НССК при сохранении эффективности обработки данных и потребления памяти.

Анализ результатов тестирования НССК показал, что модель демонстрирует улучшение производительности с каждой эпохой, что видно по снижению потерь при тестировании. Это говорит о том, что НССК адекватно адаптируется к зашифрованным данным и эффективно их обрабатывает. Результаты тестирования модели показали высокую точность классификации как для каждого из классов, так и в целом, что подтверждает эффективность модели в задачах классификации. Интересно отметить, что зашифрованная версия модели показала сопоставимую, а в некоторых случаях даже более высокую точность, что говорит о том, что применение ПГШ не оказывает существенного негативного влияния на способность модели к классификации. Тем не менее следует отметить, что использование приближенной ПГШ требует дальнейших исследований для оптимизации баланса между безопасностью, конфиденциальностью и производительностью модели. Важно изучить влияние различных типов аппроксимации функции активации на точность и общую производительность модели, а также разработать методы улучшения производительности НССК.

В статье предложен алгоритм скалярного умножения, позволяющий уменьшить пространственную сложность с $O(n^2)$ до $O(n)$ и сократить время вычисления скалярного умножения в 1.38 раза.

Результаты данного исследования открывают новые перспективы для разработки безопасных НС, особенно в тех областях, где требуется обработка конфиденциальных данных, а также подчеркивают важность продолжения исследований в этой области для достижения оптимального сочетания безопасности, конфиденциальности и эффективности в НССК. В будущем планируется исследовать и разработать методы выполнения других операции в НССК для повышения эффективности вычислений, потребления памяти, точности и конфиденциальности.

## 6. ИСТОЧНИК ФИНАНСИРОВАНИЯ

Исследование выполнено за счет гранта Российского научного фонда № 19-71-10033, https://rscf.ru/project/19-71-10033/.

## СПИСОК ЛИТЕРАТУРЫ

1. *Hunt E.B.* Artificial intelligence. Academic Press, 2014.
2. *Radford A. et al.* Improving language understanding by generative pre-training. OpenAI, 2018.
3. *Wamser F. et al.* Traffic characterization of a residential wireless Internet access // Telecommunication Systems. Springer, 2011. V. 48. P. 5–17.
4. *Sagiroglu S., Sinanc D.* Big data: A review // 2013 international conference on collaboration technologies and systems (CTS). IEEE, 2013. P. 42–47.
5. *О персональных данных* [Electronic resource]. http://pravo.gov.ru/proxy/ips/?docbody&nd=102108261 (accessed: 16.06.2024)
6. *Gentry C.* A fully homomorphic encryption scheme. Stanford university, 2009.
7. *Yegnanarayana B.* Artificial neural networks. PHI Learning Pvt. Ltd., 2009.
8. *Pratiwi H. et al.* Sigmoid activation function in selecting the best model of artificial neural networks // Journal of Physics: Conference Series. IOP Publishing, 2020. V. 1471. № 1. P. 012010.
9. *Rivest R.L., Shamir A., Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. 1978. V. 21. № 2. P. 120–126.
10. *ElGamal T.* A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE transactions on information theory. IEEE, 1985. V. 31. № 4. P. 469–472.
11. *Gentry C.* Fully homomorphic encryption using ideal lattices // Proceedings of the forty-first annual ACM symposium on Theory of computing. Bethesda MD USA: ACM, 2009. P. 169–178.
12. *Van Dijk M. et al.* Fully Homomorphic Encryption over the Integers // Advances in Cryptology – EUROCRYPT 2010 / ed. Gilbert H. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. V. 6110. P. 24–43.
13. *Gentry C., Halevi S.* Implementing gentry's fully-homomorphic encryption scheme // Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15–19, 2011. Proceedings 30. Springer, 2011. P. 129–148.
14. *Brakerski Z.* Fully homomorphic encryption without modulus switching from classical GapSVP // Annual Cryptology Conference. Springer, 2012. P. 868–886.
15. *Brakerski Z., Vaikuntanathan V.* Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages // Advances in Cryptology – CRYPTO 2011 / ed. Rogaway P. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. V. 6841. P. 505–524.

16. *Brakerski Z., Gentry C., Vaikuntanathan V.* (Leveled) Fully Homomorphic Encryption without Bootstrapping // ACM Trans. Comput. Theory. 2014. V. 6. № 3. P. 1–36.
17. *Dijk M. van et al.* Fully homomorphic encryption over the integers // Annual international conference on the theory and applications of cryptographic techniques. Springer, 2010. P. 24–43.
18. *Cheon J.H. et al.* Homomorphic encryption for arithmetic of approximate numbers // International conference on the theory and application of cryptology and information security. Springer, 2017. P. 409–437.
19. *Homomorphic Encryption Standardization* – An Open Industry / Government / Academic Consortium to Advance Secure Computation [Electronic resource]. https://homomorphicencryption.org/ (accessed: 10.12.2022)
20. *Pulido-Gaytan B. et al.* Privacy-preserving neural networks with Homomorphic encryption: Challenges and opportunities // Peer-to-Peer Netw. Appl. 2021. V. 14. № 3. P. 1666–1691.
21. *Ribeiro M., Grolinger K., Capretz M.A.* Mlaas: Machine learning as a service // 2015 IEEE14th international conference on machine learning and applications (ICMLA). IEEE, 2015. P. 896–902.
22. *Manvi S.S., Shyam G.K.* Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey // Journal of network and computer applications. Elsevier, 2014. V. 41. P. 424–440.
23. *Rodero-Merino L. et al.* Building safe PaaS clouds: A survey on security in multitenant software platforms // computers & security. Elsevier, 2012. V. 31. № 1. P. 96–108.
24. *Cusumano M.* Cloud computing and SaaS as new computing platforms // Commun. ACM. 2010. V. 53. № 4. P. 27–29.
25. *Chen H., Chillotti I., Song Y.* Improved bootstrapping for approximate homomorphic encryption // Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part II. Springer, 2019. P. 34–54.
26. Microsoft SEAL: C++. Microsoft, 2023.
27. *OpenFHE.org* – OpenFHE – Open-Source Fully Homomorphic Encryption Library [Electronic resource]. https://www.openfhe.org/ (accessed: 01.04.2024)
28. *Dai W., Sunar B.* cuHE: A homomorphic encryption accelerator library // International Conference on Cryptography and Information Security in the Balkans. Springer, 2015. P. 169–186.
29. *Benaissa A. et al.* TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption: arXiv:2104.03152. arXiv, 2021.
30. *Lee J.-W. et al.* Privacy-preserving machine learning with fully homomorphic encryption for deep neural network // IEEE Access. IEEE, 2022. V. 10. P. 30039–30054.
31. *Halevi S., Shoup V.* Algorithms in helib // Annual Cryptology Conference. Springer, 2014. P. 554–571.
32. *Özerk Ö. et al.* Efficient number theoretic transform implementation on GPU for homomorphic encryption // The Journal of Supercomputing. Springer, 2022. V. 78. № 2. P. 2840–2872.

# HIGH-SPEED CONVOLUTION CORE ARCHITECTURE FOR PRIVACY-PRESERVING NEURAL NETWORKS

**© 2024 M. A. Lapina[a], E. M. Shiriaev[a], M. G. Babenko[a], I. Istamov[b]**

[a] *North Caucasian Center for Mathematical Research, North Caucasus Federal University*
*1, Pushkina st., Stavropol, 355017 Russia*
[b] *Samarkand State University named after Sharof Rashidov*
*15, University blv. Samarkand, 140104 Uzbekistan*

Due to legal restrictions or restrictions related to companies' internal information policies, businesses often do not trust sensitive information to public cloud providers. One of the mechanisms to ensure the security of sensitive data in clouds is homomorphic encryption. Privacy-preserving neural networks are used to design solutions that utilize neural networks under these conditions. They exploit the homomorphic encryption mechanism, thus enabling the security of commercial information in the cloud. The main deterrent to the use of privacy-preserving neural networks is the large computational and spatial complexity of the scalar multiplication algorithm, which is the basic algorithm for computing mathematical convolution. In this paper, we propose a scalar multiplication algorithm that reduces the spatial complexity from quadratic to linear, and reduces the computation time of scalar multiplication by a factor of 1.38.

*Keywords*: matrix operations; artificial neural networks; fully homomorphic encryption; CKKS; TenSEAL, privacy-preserving neural networks

*ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 12–23*

## ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

*УДК 004.056.5*

# АЛГОРИТМЫ РАЗМЕЩЕНИЯ И ЗАПРОСА К КОНФИДЕНЦИАЛЬНЫМ ДАННЫМ НА ОБЛАКЕ

**© 2024 г. Н. П. Варновский[a], *, С. А. Мартишин[b], **, М. В. Храпченко[b], ***, А. В. Шокуров[b], ****

[a] *Институт проблем информационной безопасности МГУ им. М.В. Ломоносова 119192 Москва, Мичуринский проспект, 1, офис 10, Россия*

[b] *Институт системного программирования имени В.П. Иванникова РАН 109004 Москва, ул. А. Солженицына, д. 25, Россия*

* *E-mail: otd13isp@gmail.com*

** *E-mail: mart@ispras.ru*

*** *E-mail: khrap@ispras.ru*

**** *E-mail: shok@ispras.ru*

Поступила в редакцию 10.07.2024 г.
После доработки 17.07.2024 г.
Принята к публикации 17.07.2024 г.

Авторами рассматривается задача PIR (Private Information Retrieval) обеспечения безопасных запросов к базе данных. Ранее авторы рассматривали задачу для базы данных, размещенной на облаке при наличии активного противника, который не вмешивается в выполнение протокола, но может производить атаку с известными открытыми запросами. В предложенных алгоритмах номер бита $i$ представляется в системе счисления по основанию $l$ с числом разрядов $d$. Предложен алгоритм размещения базы данных на облаке и алгоритм запроса требуемого бита с использованием перестановок в цифрах разряда номера бита, при задании номера бита $i$ в системе счисления по основанию $l$. Перестановки рассматриваются как секретные ключи шифрования. Приведены оценка коммуникационной сложности и оценки вероятности угадывания номера бита при однократной атаке с известным открытым запросом номера бита $i$ и при атаке с неограниченным числом известных открытых запросов.

*Ключевые слова*: база данных, облачные вычисления, PIR

**DOI:** 10.31857/S0132347424060027, **EDN:** DZDHBD

## 1. ВВЕДЕНИЕ

В работах [1, 2] авторы исследовали задачу организации конфиденциальных запросов к базе данных (PIR – Private Information Retrieval) в случае размещения реплицированной базы данных на облаке и наличия активного противника, работающего по протоколу, но имеющему возможность производить атаку с известными открытыми запросами.

Задача PIR в информационно-теоретической постановке была сформулирована в 1995 г. Шором, Голдрайхом, Кушелевицем и Суданом [3].

Классическая постановка задачи PIR:

• имеется база данных – бинарная строка $X = (x_1, ..., x_n)$ длины $n$, хранящаяся на сервере в облаке;

• клиент хочет получить один бит информации $x_i$ из базы данных $X$ так, чтобы никто, кроме клиента, обращающегося с запросом, не смог определить, с какой позиции $i$ был запрошен бит.

В работах [3, 4] было показано, что если база данных размещена на единственном сервере, который полностью контролируется противником, то теоретико-информационное условие конфиденциальности корректного протокола PIR может быть выполнено только том случае, когда клиент запрашивает базу данных целиком. Однако в этом случае коммуникационная сложность протокола, т. е. общее количество бит, которыми обмениваются участники протокола за время его работы, будет не меньше размера базы данных, что делает его практически не применимым.

Для уменьшения коммуникационной сложности в работах [3, 4] была предложена модель реплицированной базы данных, в которой несколько одинаковых копий строки $X = (x_1, ..., x_n)$

размещалось на разных серверах. Предполагалось, что клиент имеет связь с каждым из этих серверов, но сами серверы не имеют связи друг с другом. Также предполагалось, что противник может наблюдать любой из серверов, на которых размещена база данных, причем, возможно, разные серверы во время выполнения разных сеансов протокола.

Очевидно, что в облачных информационных системах нельзя обеспечить изолированность копий распределенной базы данных друг от друга. Поскольку ни пользователь, ни клиент не могут контролировать облако, то всегда предполагается наличие на облаке противника. С учетом этого выбирается модель для дальнейших исследований. Важное отличие от классической модели, в рассматриваемой модели противники на разных серверах могут общаться.

В работах [1, 2] предлагалась модель, включающая в себя облако, состоящее из нескольких серверов, на каждом из которых хранится копия одной и той же базы данных, дилера, центра аутентификации, пользователей, клиентов, противника.

Серверы в облаке соединены посредством незащищенных каналов связи друг с другом и дилером. По этим каналам серверы обмениваются информацией между собой и дилером. Облачные серверы и незащищенные каналы связи доступны для стороннего наблюдателя. Дилер находится вне облака, противнику недоступен. Основной функцией дилера являлось шифрование и дешифрование информации при работе с облаком. Предполагалось, что противник не только пассивно наблюдает, но производит атаку с известными открытыми запросами.

В работах [1, 2] была получена оценка коммуникационной сложности работы протокола, а также оценки вероятности угадывания противником, работающим по протоколу, номера бита i при однократной атаке и при атаке с неограниченным числом известных открытых запросов.

В отличие от работ [1, 2] в данной статье предлагается вместо шифрования использовать перестановки номеров битов.

Предложены алгоритмы, основанные на перестановках, для представления номера бита в новой системе счисления, формирования запроса к облаку, обработки ответа облака и получения значения искомого бита. Даны новые оценки коммуникационной сложности алгоритмов и вероятности угадывания противником номера запрошенного бита.

В отличие от рассматриваемых в [1, 2] алгоритмах предлагается: использовать перестановки цифр номера бита в выбранной системе счисления вместо шифрования. В новом алгоритме поиск противником рядом стоящих запрашиваемых битов затруднен, и замена сдвига в интервале множества битов на перестановку не позволяет противнику сделать вывод по одному запрашиваемому множеству об остальных не пересекающихся с этим множеством множествах.

Получены оценки вероятности угадывания номера бита при однократной атаке с известным открытым запросом номера бита $i$ и при атаке с неограниченным числом известных открытых запросов при наличии противника на облаке. Приведена оценка коммуникационной сложности алгоритма.

## 2. МОДЕЛЬ ВЫЧИСЛЕНИЙ, ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И ПОСТАНОВКА ЗАДАЧИ

### *2.1. Состав модели*

Состав модели вычислений, для которой в данной статье исследуется облачный вариант задачи PIR, аналогичен составу модели, рассмотренной в [2].

Модель включает в себя:

**Облако.** Состоит из нескольких серверов, хранящих копии одной и той же базы данных. Копии на серверах имеют разные перестановки номеров битов. Серверы соединены посредством незащищенных каналов связи друг с другом и дилером.

**Пользователь.** Хранит данные на облаке. Предполагается, что на облаке хранится $k$ копий баз данных. Для загрузки данных пользователь обращается к дилеру по каналу связи, использующему стандартный криптографический протокол для обмена зашифрованными данными.

**Клиенты.** Запрашивают некоторую информацию из базы данных. Обращаются для выполнения запроса к дилеру. С дилером соединены каналами связи, использующими стандартный криптографический протокол для обмена зашифрованными данными.

**Центр аутентификации.** Аутентифицирует пользователя и клиентов.

**Дилер.** Находится вне облака, противнику недоступен. Канал связи с облаком является незащищенным. Объем памяти дилера для постоянного хранения данных пренебрежимо мал по сравнению с $n$. Получив от клиента данные для размещения на облаке, дилер выполняет перестановку уникальную для каждой копии базы данных, размещает данные на облаке. При запросе клиента дилер возвращает клиенту значение.

В дальнейшем для простоты будем считать, что база данных загружается одним пользователем.

Предполагается, что после загрузки база данных не может быть изменена. Либо изменена полностью. Также будем рассматривать случай запроса одного бита.

### *2.2. Основные определения*

**Противник.** Не вмешивается в выполнение криптографического протокола. Имеет доступ к базе данных на облаке, к каждой ее копии, к каналам связи на облаке. Может создавать фальшивых клиентов, работающих по протоколу. То есть противник не только пассивно наблюдает, но и сам может производить атаку с известными открытыми запросами. Противник может заставить дилера отправить запрос на облако. При этом противник знает алгоритм формирования запроса к облаку, но не знает конкретных параметров. Противник также знает алгоритм формирования ответа клиенту дилером на основании ответа, полученного дилером от облака.

**Угроза:** угадывание исходного номера бита, запрашиваемого клиентом в базе данных.

**Атака:**

• Атака пассивного противника на облаке заключается в наблюдении за запросом от дилера и ответом облака дилеру. Эта информация используется для сбора статистики и анализа.

• Атака активного противника (атака с известными открытыми запросами) заключается в создании фальшивых клиентов и управление ими. При помощи фальшивых клиентов активный противник может сформировать произвольное число запросов, что позволит в коалиции с пассивным противником на облаке собрать и проанализировать информацию для всей базы данных.

### *2.3. Постановка задачи и основные обозначения*

В отличие от классической постановки задачи PIR будем рассматривать базу данных (бинарная строка) $X = (x_0, ..., x_{n-1})$ длины $n$, где элементы $X$ нумеруются с нуля. Это делает более удобным работу с различными системами счисления.

Клиент хочет получить один бит информации $x_i$ с номером $i$ из базы данных $X$ так, чтобы противник не узнал ничего о том, с какой позиции $i$ был запрошен бит.

Будем размещать на облаке $k$ копий баз данных. Пусть $k = 2^d$, где $d \geq 2$. На практике $4 \leq k \leq 32$. В дальнейшем будем предполагать, что на $k$ наложены эти ограничения.

Без потери общности предполагается, что $n = l^d$, т. е. $d = \log_2 k$ и $l = \sqrt[d]{n}$. Заметим, что поскольку $l$-основание системы счисления, то $l \geq 2$ и $d$ подбирается так, чтобы $l \ll n$. Пусть $L_p = \dfrac{n}{l}$.

Так как $l = \sqrt[d]{n}$, округлим $l$ до целого числа в большую сторону.

Пусть $x$ – элемент группы $Z_2$, обозначим через $h\left(h \in \{1, \ldots, k\}\right)$ – номер копии базы данных.

## 3. ПРЕДСТАВЛЕНИЕ НОМЕРА БИТА $l$-ИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

Представим номер бита как $d$ разрядов в $l$-ичной системе счисления. Выбор $l$ и $d$ основывается на известной модели [1, 2], где номер $i$ бита $x_i$ представлен в $l$-ичной записи: $i = a_0^{(i)} + a_1^{(i)} l + \ldots + a_{d-1}^{(i)} l^{d-1}$. Цифры $l$-ичной записи представляют собой элементы кортежа $\left(a_0^{(i)}, \ldots, a_{d-1}^{(i)}\right)$ длины $d$, т. е. на $j$-м месте кортежа стоит $l$-ичная цифра.

Эти $d$ элементов кортежа можно интерпретировать как точку с целочисленными координатами в гиперкубе размерности $d$ и длиной стороны $l$, где $l = \sqrt[d]{n}$. Такое множество точек куба размерности d можно рассматривать как множество слов длины $d$ с алфавитом $(0, ..., l-1)$.

## 4. ЗАГРУЗКА ДАННЫХ НА ОБЛАКО

### *4.1. Инициализация массива $X^c$*

Напомним, что $n = l^d$ и $L_p = \dfrac{n}{l}$. Выберем целые числа $d_u \geq 2$ и $l_u \geq 2$ такие, чтобы $l_u^{d_u} \approx L_p$ и обозначим $L_p^- = l_u^{d_u}$ $(L_p^- \leq L_p)$. Пусть $l^* = \left\lceil \dfrac{n}{L_p^-} \right\rceil$, где $l^* \geq l$ и $n_u = l^* L_p^-$. Ниже будет показано, что коммуникационная сложность зависит от $l^*$. Для того чтобы коммуникационная сложность не возрастала, нужно подобрать значения $d_u$ и $l_u$ так, чтобы минимизировать разность $L_p - L_p^-$. Конкретный выбор значений $d_u$ и $l_u$, который не только минимизирует разность $L_p - L_p^-$, но и уменьшает количество вычислительных операций, производимых дилером, будет описан ниже.

Поскольку $n_u \geq n$, добавим в базу данных $n_u - n$ битов. Добавленные биты могут быть заполнены произвольными значениями. Биты будут добавлены в конец базы данных.

Пусть $l_c$ и $d_c$ целые числа, такие, что $n_c = l_c^{d_c}$ и $n_c \geq n_u$, тогда $n_c \geq n$.

На облако будет загружаться массив $X^c = \left(x_0^c, \ldots, x_{n_c-1}^c\right)$, где $n_c \geq n$, т. е. на облако будет

загружено $n_c$ битов. Нумерация массива $X^c$ начинается с нуля. Увеличение числа элементов в массиве необходимо, чтобы вместо шифрования битов использовать их секретную перестановку. Это позволяет уменьшить объем памяти, необходимой дилеру для хранения данных и снизить коммуникационную сложность.

### *4.2. Построение матрицы перестановок*

Номер запрашиваемого бита в $l_c$-ичной системе счисления можно представить как целочисленную точку гиперкуба размерности $d_c$ с длиной стороны $l_c$. При этом $j$-я ($j \in \{1,\dots,d_c\}$) координата точки гиперкуба является $j$-м разрядом представления числа в $l_c$-ичной системе счисления.

Изменим номера битов путем изменения цифр в разрядах номеров в $l_c$-ичном представлении. Цифры в каждом разряде изменяются с помощью перестановок цифр от 0 до $l_c - 1$. Для каждого разряда вычисляется своя перестановка. Поскольку перестановки являются случайными, то для разных разрядов возможно, что перестановки совпадают.

Выберем копию базы данных $h \in \{1,\dots,k\}$. Пусть $\beta_{hj}$ – случайная перестановка из $l_c$ чисел от 0 до $l_c - 1$, где $j \in \{1,\dots,d_c\}$ – номер разряда числа в $l_c$-ичной системе счисления. Представим перестановку чисел в виде таблицы:

$$\begin{pmatrix} 0 & \dots & l_c - 1 \\ \beta_{hj}(0) & \dots & \beta_{hj}(l_c - 1) \end{pmatrix}.$$

Тогда $\beta_h = \{\beta_{h1},\dots,\beta_{hd_c}\}$ – множество "поразрядных" перестановок числа в $l_c$-ичной системе счисления.

Построим матрицу $M_\beta$ размера $k \times d_c$, где $k$ – число копий базы данных, а $d_c$ – целое число, такое, что $n_c = l_c^{d_c}$. Элементом $\beta_{ij}$ матрицы $M_\beta$ является случайная перестановка целых чисел от 0 до $l_c - 1$. Таким образом, для каждой копии базы данных будет построено $d_c$ случайных перестановок.

Для генерации перестановок цифр номера бита $l_c$-ичной системе счисления используется датчик псевдослучайных чисел $PRNG(db, m)$. На вход $PRNG(db, m)$ подается уникальное значение $db$ для конкретной базы данных и номер копии базы данных $m$. Датчик псевдослучайных чисел при одинаковой инициализации генерирует одинаковую перестановку.

Поскольку матрица $M_\beta$ требует для хранения значительного объема памяти, матрица $M_\beta$ при необходимости восстанавливается с помощью датчика псевдослучайных чисел.

### *4.3. Отображение $i \xrightarrow{\beta_h} i_c$*

Номер бита $i$ представляем в системе счисления по основанию $l_c$ с числом разрядов $d_c$:

$$i = \sum_{j=1}^{d_c} a_j \cdot l_c^{j-1}, \text{ где } a_j \in \{0,\dots,l_c - 1\}.$$

Для копии базы данных с номером $h$ выполним перестановки для разрядов числа $i$:

$$i_c = \sum_{j=1}^{d_c} \beta_{hj}(a_j) \cdot l_c^{j-1}.$$

Обозначим такое преобразование через $i \xrightarrow{\beta_h} i_c$, где $\beta_h = \{\beta_{h1},\dots,\beta_{hd_c}\}$ – строка матрицы $M_\beta$. Необходимые для работы алгоритма элементы матрицы $M_\beta$ генерируются с помощью датчика псевдослучайных чисел.

### *4.4. Вспомогательные построения, необходимые для работы алгоритма запроса бита – алгоритм построения матрицы G*

Построим матрицу $G$ размера $n_c \times 2$:

$$G = \begin{pmatrix} 0 & x_0^c \\ \vdots & \vdots \\ n_c - 1 & x_{n_c-1}^c \end{pmatrix}.$$

Первый столбец этой матрицы – последовательность $n_c$ целых чисел $[0, n_c - 1]$. Второй столбец матрицы $G$ – элементы $X^c = (x_0^c,\dots,x_{n_c-1}^c)$.

### *4.5. Формирование матриц $G_h$ для каждой копии базы данных из матрицы G*

**Алгоритм формирования матриц $G_h$**

Вход: матрица $G$.

Выход: матрицы $G_h$ для всех $h \in \{1,\dots,k\}$.

**Шаг 1.** Выберем копию базы данных $h$.

**Шаг 2.** Скопируем матрицу $G$ в матрицу $G_{aux}$.

**Шаг 3.** Для каждого элемента $i$, где $i = 0,\dots,n_c - 1$ первого столбца матрицы $G_{aux}$ выполним преобразование $i \xrightarrow{\beta_h} i_c$.

**Шаг 4.** Отсортируем строки матрицы $G_{aux}$ по первому столбцу. Получим матрицу $G_h$.

**Шаг 5.** Выполним Шаги 1–4 для всех $h \in \{1,\dots,k\}$ ■.

В результате работы алгоритма получим $G_h$ $(h \in \{1,\dots,k\})$ матриц для каждой копии базы данных.

### *4.6. Загрузка данных на облако*

Алгоритм загрузки данных на облако аналогичен алгоритму, приведенному в [1, 2].

На вход алгоритма подается бинарный массив $X = (x_1, ..., x_n)$. Для удобства будем предполагать, что $X = (x_0, ..., x_{n-1})$. В результате работы алгоритма на облако загружается $k$ преобразованных с помощью перестановки копий баз данных.

**Алгоритм загрузки данных на облако**

Вход: $X$.

Выход: загрузка на облако $G_h$, где $h \in \{1, \ldots, k\}$.

**Шаг 1.** Дилер аутентифицирует пользователя. Пользователь передает дилеру массив значений битов $X = (x_0, ..., x_{n-1})$. Дилер преобразует массив $X = (x_0, ..., x_{n-1})$ в массив $X^c = \left(x_0^c, \ldots, x_{n_c-1}^c\right)$.

**Шаг 2.** Дилер формирует матрицу $G = \|g_{i,j}\|$, где $i = 0, \ldots, n_c - 1,\ j = 1, 2$.

**Шаг 3.** Дилер генерирует элементы матрицы $M_\beta$ в процессе работы.

**Шаг 4.** Дилер создает матрицу $G_h$, где $h \in \{1, \ldots, k\}$.

**Шаг 5.** Дилер загружает матрицу $G_h$, где $h \in \{1, \ldots, k\}$ на облако.

**Шаг 6.** Шаги 4, 5 дилер выполняет в цикле для каждой копии базы данных. По окончании перебора $k$ копий базы данных происходит выход из алгоритма ■.

Число элементов, переданных каждой из $k$ копий базы данных равно $2n_c$.

Заметим, что поскольку в дальнейшем восстановление матрицы $G$ не предполагается, матрицу $G$ дилер не хранит.

Для выполнения перестановки дилеру требуется иметь объем памяти, сравнимый с объемом, необходимым для хранения одной копии базы данных. При этом предполагается, что дилер может работать с несколькими базами данных и нет необходимости их одновременного хранения. После загрузки базы данных на облако память дилера очищается для работы со следующей базой данных.

В приведенном ниже алгоритме дилер должен выполнять Шаги 4 и 5 для каждой копии базы данных последовательно. Иначе ему придется хранить в памяти все $k$ копий базы данных, что приводит к большим расходам памяти.

## 5. ВЫПОЛНЕНИЕ ДИЛЕРОМ ЗАПРОСА КЛИЕНТА

Для сокрытия при запросе искомого номера бита, вместо одного запрашиваемого номера бита дилер будет запрашивать у облака множество номеров битов мощностью $l^*$. Причем только один из этих номеров битов был запрошен клиентом. Каждый элемент из множества номеров битов лежит только в одном из интервалов мощности $L_p^-$ разбиения этого множества.

### *5.1. Использование множества номеров битов в запросе для сокрытия номера бита*

Пусть клиент интересуется элементом $x_i \in X$. Этому элементу соответствует $i$-я строка матрицы $G$, причем $0 \le i < n$. Отрезок целых чисел от 0 до $n_c - 1$ содержит $l^*$ интервалов по $L_p^-$ элементов в каждом из них.

Номер $i$ лежит ровно в одном из этих интервалов. Далее для этого элемента $i$ выберем по одному элементу в каждом из оставшихся $l^* - 1$ интервалов (алгоритм выбора элементов описан в п. 5.2). Таким образом, каждому номеру $i$ поставим в соответствие другие $l^* - 1$ чисел, лежащих в различных интервалах и не попадающих в интервал, где лежит само число $i$.

Как было сказано выше, каждый интервал можно интерпретировать как дискретный гиперкуб размерности $d_u$ с длиной стороны $l_u$.

Выберем интервал $m \in \{1, \ldots, l^*\}$. Пусть $\sigma_{mj}$ – случайная перестановка из $l_u$ чисел от 0 до $l_u - 1$, где $j \in \{1, \ldots, d_u\}$ – разряд числа в $l_u$-ичной системе счисления. Как и ранее, запишем перестановку чисел в виде

$$\begin{pmatrix} 0 & \ldots & l_u - 1 \\ \sigma_{mj}(0) & \ldots & \sigma_{mj}(l_u - 1) \end{pmatrix}.$$

Тогда $\sigma_m = \{\sigma_{m1}, \ldots, \sigma_{md_u}\}$ – множество “поразрядных” перестановок числа в $l_u$-ичной системе счисления $d_u$-разрядного числа.

Построим матрицу $M_\sigma$ (по аналогии с матрицей $M_\beta$) размера $l^* \times d_u$, где $l^*$ – число интервалов, а $d_u$ – целое число, такое, что $L_p^- = l_u^{d_u}$. Элементом $\sigma_{ij}$ матрицы $M_\sigma$ является случайная перестановка целых чисел от 0 до $l_u - 1$. Таким образом, для каждого интервала на которые разбивается база данных будет построено $d_u$ случайных перестановок.

Для генерации перестановок цифр номера бита матрицы $M_\sigma$ используется датчик псевдослучайных чисел $PRNG(db, m, j)$. На вход $PRNG(db, m, j)$ подается уникальное значение $db$ для конкретной базы данных, номер интервала $m$ (всего $l^*$ штук) и $j \in \{1, \ldots, d_u\}$ – разряд числа в $l_u$-ичной системе счисления. Датчик псевдослучайных чисел при одинаковой инициализации генерирует одинаковую перестановку.

Поскольку матрица $M_\sigma$ требует для хранения значительного объема памяти, элементы матрицы $M_\sigma$ восстанавливаются с помощью датчика псевдослучайных чисел.

Поскольку отрезок целых чисел от 0 до $n_c - 1$ разбивается на $l^*$ интервалов по $L_p^-$ элементов в каждом интервале, то любое число из этого множества попадет в один из таких интервалов.

Найдем номер интервала $w$, в который попало число $i$ при разбиении отрезка целых чисел от 0 до $n_c - 1$ на $l^*$ интервалов:

$$w = \left\lfloor \frac{i}{L_p^-} \right\rfloor + 1.$$

Заметим, что $w \in \{1, \ldots, l^*\}$.

Приведем $i$ по модулю $L_p^-$. Обозначим через $i' = i \bmod L_p^-$ результат приведения. Тогда $i'$ является порядковым номером в интервале с номером $w$.

Представим порядковый номер $i'$ в интервале с номером $w$ в $l_u$-ичной системе счисления:

$$i' = \sum_{j=1}^{d_u} a_j \cdot l_u^{j-1}, \text{ где } a_j \in \{0, \ldots, l_u - 1\}.$$

Рассмотрим сумму $\sum_{j=1}^{d_u} \left(\sigma_{wj}\right)^{-1}\left(a_j\right) \cdot l_u^{j-1}$, где $\left(\sigma_{wj}\right)^{-1}\left(a_j\right)$ – обратная перестановка, такая, что $\sigma_{wj}\left(\left(\sigma_{wj}\right)^{-1}\left(a_j\right)\right) = a_j$.

Обозначим через $D_j = \left(\sigma_{wj}\right)^{-1}\left(a_j\right)$, величина $D_j$ – секрет, известный только дилеру.

В каждом интервале $m\left(m \in \left[1, l^*\right]\right)$ найдем число, такое, что $i'(m) = \sum_{j=1}^{d_u} \sigma_{mj}\left(D_j\right) \cdot l_u^{j-1}$, где $j = 1, \ldots, d_u$. Для всех интервалов таких чисел $i'(m)$ будет $l^*$. Заметим, что для интервала $w$ это будет число $i'$.

Обозначим через $Set_i$ полученное по алгоритму 5.2 (Алгоритм построения множества $Set_i$ дилером) множество из $l^*$ чисел.

### *5.2. Построение множеств $Set_i$ дилером*

**Алгоритм построения множества $Set_i$ дилером**

Вход: $X^c$, $i$, $l^*$, $d_u$, $L_p^-$.
Выход: множество $Set_i$.
$Set_i = \varnothing$
// найдем номер интервала

$$w \leftarrow \left\lfloor \frac{i}{L_p^-} \right\rfloor + 1$$

// найдем порядковый номер числа $i$
// в соответствующем интервале разбиения
$i' = i \bmod L_p^-$
// найдем представление числа $i'$ в $l_u$-ичной записи

$$i' = \sum_{j=1}^{d_u} a_j \cdot l_u^{j-1},\ a_j \in \{0, \ldots, l_u - 1\}$$

// найдем координаты $D_j \in \{0, \ldots, l_u - 1\}$
for $j \leftarrow 1$ to $d_u$ do

$$D_j = \left(\sigma_{wj}\right)^{-1}\left(a_j\right)$$

for $m \leftarrow 1$ to $l^*$ do

$$i'(m) = \sum_{j=1}^{d_u} \sigma_{mj}\left(D_j\right) \cdot l_u^{j-1}$$

$$Set_i = Set_i \cup \left(L_p^- m + i'(m)\right)$$

Листинг 1. Алгоритм построения множества $Set_i$

### *5.3. Запрос клиентом бита с номером i. Предварительные замечания*

Пусть клиент запрашивает бит с номером $i$.

После подтверждения центром аутентификации полномочий клиента дилер разрешает клиенту отправить запрос.

Напомним, что физически в облаке хранятся матрицы $G_h$, $h \in \{1, \ldots, k\}$. Строка матрицы $G_h$ состоит из номера строки и значения бита. Без потери общности (как было сказано выше) будем индексировать элементы базы данных с 0.

Так же как и в [1, 2] для каждого элемента множества $Set_i$ случайно и равномерно выбирается номер копии базы данных. Все элементы множества $Set_i$ для которых выбрана копия базы данных $h$, обозначим через множества $Set_i^h$, где $h \in \{1, \ldots, k\}$. Очевидно, что множества $Set_i^h$ не пересекаются между собой, поскольку все элементы множества $Set_i$ различны и каждому элементу ставится в соответствие ровно один номер $h$ (номер копии базы данных). Объединение множеств $Set_i^h$ содержит все элементы множества $Set_i$, т. е. $Set_i = \cup Set_i^h$, где $h \in \{1, \ldots, k\}$.

Пусть для запрашиваемого номера $i$ в множестве $Set_i$ была выбрана копия базы данных $h$.

Для каждого элемента множества $Set_i^h$ выполняется перестановка $\beta^h$, пусть $y = \beta^h(i)$. Полученные после перестановки элементы сортируются. Это множество обозначим через $Set_i^{\beta^h}$. Позиция $i_{pos}$ элемента $y = \beta^h(i)$ в множестве $Set_i^{\beta^h}$ запоминается.

Дилер на время выполнения запроса формирует вектор-строку $\boldsymbol{s}_{num}$ из трех элементов $\boldsymbol{s}_{num} = (s_1, s_2, s_3)$:

• Первый элемент вектора-строки $\boldsymbol{s}_{num}$ содержит номер элемента $i$.

• Второй элемент вектора-строки $\boldsymbol{s}_{num}$ содержит номер копии $h$ базы данных, в которую попал номер $s_1$.

• Третий элемент вектора-строки $\boldsymbol{s}_{num}$ содержит позицию $i_{pos}$ номера $y$ в множестве $Set_i^{\beta^h}$.

Вектор-строка $\boldsymbol{s}_{num}$ позволяет дилеру после получения ответа от облака сразу отбросить данные, кроме данных, необходимых для выполнения запроса.

Вектор-строка $\boldsymbol{s}_{num}$ формируется у дилера и известна только дилеру. Вектор-строка $\boldsymbol{s}_{num}$ является секретной.

На $k$ копий базы данных дилер отправляет в общей сложности $l^*$ номеров.

После выполнения запроса дилер с помощью вектора-строки $\boldsymbol{s}_{num}$ получает значение искомого элемента и отправляет его клиенту.

**Алгоритм подготовки запроса к облаку**

Вход: $X^c$, $i$, $l^*$, $d_u$, $L_p^-$.

Выход: $Set_i^{\beta^h}$, $\boldsymbol{s}_{num}$.

**Шаг 1.** Клиент обращается к дилеру и запрашивает значение бита с номером $i$.

**Шаг 2.** Дилер генерирует множество $Set_i$.

**Шаг 3.** Дилер на время выполнения запроса $i$-го бита резервирует память для вектора-строки $\boldsymbol{s}_{num}$ трех элементов.

**Шаг 4.** Дилер заносит $i$ в первый элемент вектора-строки $\boldsymbol{s}_{num}$.

**Шаг 5.** Дилер выполняет разбиение множества $Set_i$ на непересекающиеся подмножества $Set_i^h$, где $h \in \{1,\ldots,k\}$, путем случайного и равномерного выбора номера копии базы данных для каждого элемента множества $Set_i$.

Дилер заносит во второй элемент вектора-строки $\boldsymbol{s}_{num}$ номер копии базы данных $s_2$, соответствующей элементу $s_1$.

**Шаг 6.** Дилер выполняет перестановку $\beta^h$ для элементов множества $Set_i^h$ после чего полученные после перестановки новые номера сортируются. Получаем множество $Set_i^{\beta^h}$. Этот шаг выполняется для всех $h \in \{1,\ldots,k\}$.

**Шаг 7.** Дилер заносит в третий элемент вектора-строки $\boldsymbol{s}_{num}$ позицию $i_{pos}$ номера $y = \beta^{s_2}(i)$ в множестве $Set_i^{\beta^h}$ ■.

### *5.4. Выполнение запроса к облаку и ответ облака*

**Алгоритм обмена информацией с облаком**

Вход: $Set_i^{\beta^h}\left(h \in \{1,\ldots,k\}\right)$.

Выход: значения битов в том же порядке, в котором передавались номера битов множества $Set_i^{\beta^h}$ для каждой копии базы данных $\left(h \in \{1,\ldots,k\}\right)$.

**Шаг 1.** Для каждой копии базы данных дилер отправляет на облако множества $Set_i^{\beta^h}$.

**Шаг 2.** Для запрошенных номеров битов облако возвращает дилеру значения битов из второго столбца матрицы $G_{\beta^h}$, где $h \in \{1,\ldots,k\}$). Порядок возвращаемых значений битов соответствует исходному порядку элементов множества $Set_i^{\beta^h}$ ■.

### *5.5. Обработка ответа облака*

**Алгоритм обработки ответа облака**

Вход: $\boldsymbol{s}_{num}$, значения битов в том же порядке, в котором передавались номера битов множества $Set_i^{\beta^h}$ для каждой копии базы данных $\left(h \in \{1,\ldots,k\}\right)$.

Выход: значение бита $i$.

**Шаг 1.** Дилер с помощью вектора-строки $\boldsymbol{s}_{num}$ выбирает нужное значения бита. Остальные значения битов отбрасываются.

**Шаг 2.** Дилер получает значение бита для первого элемента вектора-строки $\boldsymbol{s}_{num}$. По построению вектора-строки $\boldsymbol{s}_{num}$ запрашиваемый номер является элементом $s_1$.

**Шаг 3.** Дилер отправляет значение $i$ бита клиенту ■.

## 6. ОЦЕНКА ТРЕБУЕМОЙ ПАМЯТИ И СЛОЖНОСТИ АЛГОРИТМА

### *6.1. Объем информации, который необходимо хранить дилеру*

На протяжении всего времени существования базы данных дилер хранит информацию для базы данных объема $n_c$:

• значение $db$ для инициализации датчиков псевдослучайных чисел $PRNG(db, m)$ и $PRNG(db, m, j)$;

• значения $k$, $d_c$ и $l_c$ для датчика псевдослучайных чисел $PRNG(db, m)$, генерирующего матрицу $M_\beta$ размера $k \times d_c$. Элементом матрицы $M_\beta$ является случайная перестановка целых чисел от 0 до $l_c - 1$;

• значения $k$, $d_u$, $l_u$ и $l^*$ для датчика псевдослучайных чисел $PRNG(db, m, j)$, генерирующего матрицу $M_\sigma$ размера $l^* \times d_u$. Элементом матрицы $M_\sigma$ является случайная перестановка целых чисел от 0 до $l_u - 1$.

Объем хранимой дилером информации много меньше $n$.

*6.2. Характеристики предложенной схемы*

В данной схеме рассматривается активный противник, работающий по протоколу. Противник имеет доступ к базе данных на облаке, к каждой ее копии, к каналам связи на облаке, может создавать фальшивых клиентов, работающих по протоколу. То есть противник не только пассивно наблюдает, но и сам может производить запросы с помощью созданных фальшивых клиентов, т. е. производит атаку с известными открытыми запросами. Противник не имеет доступа к дилеру.

*6.3. Основные результаты*

Под коммуникационной сложностью для предложенной схемы будем понимать общее количество пересылаемых битов, необходимых для обмена информацией между дилером и облаком. То есть сумму числа битов, отправляемых дилером на облако и числа битов, получаемых от облака дилером, необходимых для нахождения дилером значения бита, запрашиваемого у дилера. Пусть $s$ – число битов для представления номера бита. Напомним, что $l^*$ является мощностью множества $Set_i$.

**Утверждение 1.** Коммуникационная сложность схемы получения значения номера бита дилером без раскрытия его номера равна $l^*(s+1)$ битов.

**Доказательство.** Для запроса значения бита дилер посылает копиям базы данных $l^*$ номеров битов длиной $s$ каждый. Облако отвечает $l^*$ битами ■.

Проанализируем вероятность угадывания противником номера запрашиваемого бита.

Если запросы выполнены к одному множеству, то они неразличимы. Таким образом, если противник совершает $n$ или более запросов, он не узнает номер конкретного бита. Максимум, какую информацию противник может получить – такое подмножество множества $Set_i$, что при запросе к каждому элементу из подмножества, на облаке осуществляется доступ ко всем битам из множества $Set_i$ и только к ним. Что будет означать, что противник не знает, а только угадывает, какой именно бит из $Set_i$ был выбран.

**Утверждение 2.** При однократной атаке с известным открытым запросом номера бита $i$ и предположении о наличии пассивного противника на облаке вероятность угадывания противником номера бита не более $\frac{1}{l^*}$.

**Доказательство.** Множество $Set_i$ состоит из $l^*$ элементов и для любого $j Set_i$ выполняется $Set_i = Set_j$. То есть существует одинаковое множество для $l^*$ различных номеров битов. Таким образом вероятность угадывания номера из множества $Set_i$ равна $\frac{1}{l^*}$, так как $Set_i$ имеет мощность $l^*$ ■.

Сделав $n$ или более запросов, противник получит информацию о числе реальных битов в каждом множестве $Set_i$. Число этих битов будет меньше или равно $l^*$. Таким образом, противник понимает, что запрошенный клиентом номер находится среди истинных номеров битов. И вероятность того, что клиент запрашивал конкретный номер реального бита для всех реальных номеров битов из одинаковая.

Для того чтобы оценить вероятность угадывания конкретного номера бита из множества $Set_i$, необходимо посчитать среднее число истинных битов, попадающих в множество $Set_i$.

Мощность множества $X^c$ равна $n_c$, где элементам множества $X$ соответствует $n$ ($n < n_c$) элементов. Из множества $X^c$ выбирается $l^*$ элементов. В этом случае возникает вопрос: сколько элементов $N$ из случайной выборки мощности $l^*$ в среднем соответствует элементам множества $X$?

Пусть $P_r$ – вероятность получения в выборке мощности $l^*$ ровно $r$ элементов, соответствующих элементам множества $X$. Тогда $P_r$ находится по формуле

$$P_r = \frac{C_n^r C_{n_c-n}^{l^*-r}}{C_{n_c}^{l^*}}.$$

Тогда среднее число истинных битов, попадающих в множество $Set_i$,

$$N = \sum_{r=1}^{l^*} r P_r.$$

**Утверждение 3.** При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником в среднем не более $\frac{1}{N}$.

**Доказательство.** Выполнив $n_c$ запросов, противник определяет элементы, входящие в каждое множество $Set_i$. Если фиктивные клиенты будут выполнять любое количество запросов большее $n_c$, противник все равно не получит никакой дополнительной информации.

Количество реальных бит, которые попали в каждую выборку мощности $l^*$ в среднем равно $N$.

Тогда вероятность угадывания номера $i$ в среднем будет не более $\frac{1}{N}$ ■.

**Утверждение 4.** При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$.

**Доказательство.** По построению гиперкубы, соответствующие интервалам, заполняются таким образом, что добавленные фиктивные номера битов добавляются только в интервал с номером $l^*$. Предположим, что $l^* - 1$ гиперкубов, соответствующих интервалам заполнены номерами битов, а интервал с номером $l^*$ заполнен только фиктивными битами.

В этом случае противник, выполнив любое количество запросов, определит номер фиктивного бита из интервала с номером $l^*$, поскольку из каждого интервала выбирается только один бит для построения множества $Set_i$. Следовательно, при любом количестве запросов вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$ ■.

Значение $k$ по сравнению с $n$ мало. Докажем, что предложенный протокол удовлетворяет условию лаконичности.

Как было сказано ранее, для того чтобы коммуникационная сложность не возрастала, нужно подобрать значения $d_u$ и $l_u$ так, чтобы минимизировать разность $L_p - L_p^-$. Это следует из того, что $l = \frac{n}{L_p}$ и $l^* = \left\lceil \frac{n}{L_p^-} \right\rceil$, тогда $l^* \approx l$, следовательно $l^* \ll n$.

**Теорема 1.** Для предложенной схемы организации базы данных размера $n$:

• Коммуникационная сложность запроса равна $l^*(s+1)$ битов.

• При однократной атаке с известным открытым запросом номера бита $i$ и предположении о наличии пассивного противника на облаке вероятность угадывания противником номера бита не более $\frac{1}{l^*}$.

• При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником в среднем не более $\frac{1}{N}$.

• При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$.

**Доказательство.** Из Утверждения 1 следует, что коммуникационная сложность запроса равна $l^*(s+1)$ битов.

Из Утверждения 2 следует, что при однократном запросе номера бита вероятность угадывания номера равна $\frac{1}{l^*}$.

Из Утверждения 3 следует, что при любом числе запросов номеров битов фиктивными клиентами и наличии противника на облаке вероятность угадывания номера бита в среднем не более $\frac{1}{N}$.

Из Утверждения 4 следует, что при любом числе запросов номеров битов фиктивными клиентами и наличии противника на облаке вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$ ■.

## 7. ОЦЕНКА ГЕНЕРИРУЕМОЙ ДИЛЕРОМ ИНФОРМАЦИИ

Напомним, что $n = l^d$, $l = \sqrt[d]{n}$. По приведенным выше построениям $d$ – размер гиперкуба, а $l$ – длина его стороны. Для минимизации вычислений дилером необходимо минимизировать размер вычисляемых матриц $M_\beta$ и $M_\sigma$. Для этого надо минимизировать размер строки каждой из матриц, т. е. величины $d_c l_c \left(l_c = \sqrt[d_c]{n_c}\right)$ и $d_u l_u \left(l_u = \sqrt[d_u]{n_u}\right)$.

Рассмотрим функцию $f(x) = \sqrt[x]{n} \cdot x$ описывающую соотношение между длиной стороны гиперкуба и его размерностью. Исследуем функцию $f(x)$ для того, чтобы минимизировать размер генерируемой дилером информации. Заметим, что $x \geq 1$.

По известной формуле $\left(a^{\frac{1}{x}}\right)' = -\frac{a^{\frac{1}{x}}}{x^2}\ln a$ найдем производную функции $f(x)$:

$$f'(x) = \left(n^{\frac{1}{x}} x\right)' = \left(n^{\frac{1}{x}}\right)' x + n^{\frac{1}{x}} x' =$$

$$= n^{\frac{1}{x}} \ln n \left(-\frac{1}{x^2}\right) x + n^{\frac{1}{x}} = n^{\frac{1}{x}} \left(1 - \frac{\ln n}{x}\right).$$

Производная равна нулю в точках экстремума. Найдем координату $x$ точек экстремума. Поскольку $n^{\frac{1}{x}} > 0$, приравняем к нулю второй сомножитель:

$$1 - \frac{\ln n}{x} = 0.$$

Тогда

$$\frac{\ln n}{x} = 1,$$

$$x = \ln n.$$

Теперь выясним, эта точка экстремума минимум или максимум:

$$f''(x) = \left( n^{\frac{1}{x}} \left( 1 - \frac{\ln n}{x} \right) \right)' =$$

$$= \left( n^{\frac{1}{x}} \right) \left( 1 - \frac{\ln n}{x} \right) + n^{\frac{1}{x}} \left( 1 - \frac{\ln n}{x} \right)' =$$

$$= -n^{\frac{1}{x}} \frac{\ln n}{x^2} \left( 1 - \frac{\ln n}{x} \right) + n^{\frac{1}{x}} \frac{\ln n}{x^2} = n^{\frac{1}{x}} \frac{\ln^2 n}{x^3} > 0.$$

Таким образом, минимум функции $f(x) = \sqrt[x]{n} \cdot x$ находится в точке $x = \ln n$. Следовательно, минимальное количество элементов матриц $M_\beta$ и $M_\sigma$, которое должен генерировать дилер, достигается при $x = \ln n$. Величина $x$ должна быть целым числом, поскольку число разрядов и основание системы счисления должны быть целыми числами.

Выбор $\lceil x \rceil$ или $\lfloor x \rfloor$ влияет на число интервалов $l^* = \left\lceil \frac{n}{L_p^-} \right\rceil \left( L_p^- = l_u^{d_u} \right)$. Если интервал содержит большее число элементов, то $l^*$ уменьшается, в противном случае $l^*$ увеличивается. Величина $l^*$ влияет на коммуникационную сложность.

## 8. СРАВНЕНИЕ С РАНЕЕ ПРЕДЛОЖЕННЫМИ АЛГОРИТМАМИ

Коммуникационная сложность предложенного алгоритма равна $l^*(s+1)$.

На практике для рассматриваемой базы данных размера $2^{40}$ число бит для представления номера $s$ – не более 40 битов $\le 2^6$ (биты нумеруются с нуля).

В [2] коммуникационная сложность равна $\hat{l}\, d!\, Len(K, K_{enc})$, где $Len(K, K_{enc})$ – сумма длин шифротекстов. На практике $d \le 4$.

По построению интервалов:

$l^* \le 2^{10}$;

$\hat{l} \le 2^{10}$.

Для широко используемых алгоритмов вероятностного шифрования (AES, RSA) $Len(K, K_{enc})$ не менее $128 + 128 = 256$ бит ($2^8$).

Сравним $l^*$ и $\hat{l} \cdot d!\cdot$, $l^* \le 2^{10}$, $\hat{l}\, d! \le 2^{10} \cdot 2^5$.

На практике:

• коммуникационная сложность предложенного алгоритма: $l^*(s+1) = 2^{10} \cdot 2^6$;

• коммуникационная сложность алгоритма из [2]: $2^{10} \cdot 2^4 \cdot 2^8 < \hat{l}\, d!\, Len(K, K_{enc}) < 2^{10} \cdot 2^5 \cdot 2^8$.

Напомним, что вероятность угадывания при однократном запросе с использованием перестановок:

$$\frac{1}{l^*} = \frac{1}{2^{10}}.$$

С использованием шифрования при $d = 4$:

$$\frac{1}{\hat{l}d!} \ge \frac{1}{2^{10} \cdot 2^5}.$$

Увеличим $l^*$ (пусть $l^* = 2^{15}$), чтобы вероятность угадывания при однократном запросе была одинаковая в ранее рассмотренном и предложенном алгоритме. В этом случае коммуникационная сложность предложенного алгоритма равна:

$$l^*(s+1) = 2^{15} \cdot 2^6.$$

Таким образом, при заданных параметрах на практике при аналогичной вероятности угадывания коммуникационная сложность предложенного алгоритма по крайней мере в 2 раза лучше.

Оценим объем памяти, которую необходимо генерировать дилеру на практике. Поскольку минимум функции $f(x) = \sqrt[x]{n} \cdot x$ находится в точке $x = \ln n$, определим целочисленное значение $x$ для $n = 2^{40}$. Если выбрать $x = 28$ то основание системы счисления $\sqrt[28]{n} \oplus 3$, если выбрать $x = 27$, то основание системы счисления $\sqrt[27]{n} \oplus 3$. В обоих случаях перестановка выполняется для последовательности чисел из 3 элементов и от выбора целой части $x$ не зависит. Но размерность гиперкуба зависит от выбора целой части $x$. Если размерность гиперкуба больше, то $l^*$ будет меньше, если размерность гиперкуба меньше, то $l^*$ будет больше. Как было показано выше, если $l^*$ увеличивается, то уменьшается вероятность угадывания, но увеличивается коммуникационная сложность и наоборот. Выбор целой части $x$ зависит от цели: уменьшить

вероятность угадывания или уменьшить коммуникационную сложность.

Таким образом, для рассматриваемых параметров число перестановок будет 28 и длина перестановки равна 3. Для $n = 2^{40}$ нужно генерировать $28 \cdot 3 = 84$ элемента для вычисления нового номера при запросе к облаку.

При генерации множества $Set_i$ после получения нового номера в интервале память дилера освобождается.

Генерацию перестановок как для всей базы данных и интервалов можно выполнять в разных потоках. Для каждого из $l^*$ интервалов перестановки цифр числа при формировании множества $Set_i$ могут выполняться параллельно, что увеличивает скорость работы алгоритма.

## 9. ЗАКЛЮЧЕНИЕ

Предложенный алгоритм обладает значительными преимуществами по сравнению с алгоритмами, описанными в [1, 2].

Алгоритм обладает следующими преимуществами:

• Вместо шифрования номера бита используются перестановки цифр в выбранной системе счисления. Шифрование и расшифрование номера бита на практике требует большего числа операций, чем перестановка разрядов в выбранной системе счисления.

• Так как при загрузке БД на облако выполняется перестановка, то на практике запрос рядом стоящих битов чаще всего выполняется для битов, которые не являются соседними на облаке. Это затрудняет противнику поиск стоящих рядом запрашиваемых битов.

• Все сдвиги в интервалах разные для любого множества $Set_i$. Это не позволяет сделать вывод о $Set_j$ по $Set_i$. Ранее, в статье [2], использовался вектор $\boldsymbol{b}$, который содержит величину сдвига в интервале для каждого элемента любого множества $Set_i$. Если противник угадает вектор $\boldsymbol{b}$, это позволит ему сделать вывод о $Set_j$ по $Set_i$.

Заметим также, что на практике при реализации вычислений легко организовать параллельные процессы, что позволяет увеличить скорость выполнения алгоритма.

## 10. БЛАГОДАРНОСТИ

Авторы выражают особую благодарность Лазареву Д.О. за ценные замечания в процессе работы над статьей.

## СПИСОК ЛИТЕРАТУРЫ

1. *Мартишин С.А., Храпченко М.В., Шокуров А.В.* Организация безопасного запроса к базе данных на облаке // Труды Института системного программирования РАН. 2022. Т. 34. № 3. С. 173–188. ISSN 2079-8156 (Print), ISSN 2220-6426 (Online).
2. *Варновский Н.П., Мартишин С.А., Храпченко М.В., Шокуров А.В.* Организация конфиденциальных запросов к облаку // Труды Института системного программирования РАН. 2023. Т. 35. № 5. С. 37–54. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).
3. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, in IEEE Annual Symposium on Foundations of Computer Science, 1995. P. 41–50.
4. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, Journal of the ACM, November 1998. V. 45. № 6. P. 965–982.

# CLOUD DATA PLACING AND PRIVATE INFORMATION RETRIEVAL ALGORITHMS

**© 2024 N. P. Varnovskiy[a], S. A. Martishin[b], M. V. Khrapchenko[b], A. V. Shokurov[b]**

[a] *Information Security Section of Information Security Institute of Moscow State Lomonosov University*
*Office 10, 1, Michurinskiy prospect, 119192, Moscow, Russia*
[b] *Ivannikov Institute for System Programming of the RAS*
*25, Alexander Solzhenitsyn st. 109004, Moscow, Russia*

The authors consider the problem of ensuring secure queries to the database PIR (Private Information Retrieval) problem. Previously, the authors considered the problem for a database hosted in the cloud in the presence of an active adversary who does not interfere with the execution of the protocol, but can carry out an attack with known open queries. In algorithms, bit number $i$ is represented as the $l$-ary number with a number of digits $d$. An algorithm for placing a database in the cloud and an algorithm for querying the required bit using permutations in the digits of the bit number, using the specification of the bit number $i$ in the base $l$ numerical system, were proposed. Permutations are treated as secret encryption keys. Communication complexity and probability of guessing the bit number for a one-time attack with a known open request for bit number $i$ and for an attack with unlimited number of known open requests were estimated.

*Keywords*: database, cloud computing, PIR

## REFERENCES

1. *Martishin S.A.., Khrapchenko M.V., Shokurov A.V.* Organization of a secure query to a database in the cloud. Trudy ISP RAN/Proc. ISP RAS. Vol. 34. Issue 3, 2022. P. 173–188 (in Russian). ISSN 2079-8156 (Print), ISSN 2220-6426 (Online).
2. *Varnovskiy N.P., Martishin S.A., Khrapchenko M.V., Shokurov A.V.* About cloud request protection. Trudy ISP RAN/Proc. ISP RAS. Vol. 35. Issue 5, 2023. P. 37–54 (in Russian). ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).
3. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, in IEEE Annual Symposium on Foundations of Computer Science, 1995. P. 41–50.
4. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, Journal of the ACM. Vol. 45. № 6. November 1998. P. 965–982.

*ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 24–34*

## АНАЛИЗ ДАННЫХ

*УДК 681.51+519.5*

# АДАПТИВНЫЙ БИХ-ФИЛЬТР НА БАЗЕ ШТРАФНОГО СПЛАЙНА

**© 2024 г. Е. А. Кочегурова[a],*, Ю. А. Мартынова[a],****

[a] *Национальный исследовательский Томский политехнический университет*

*634050 Томск, пр. Ленина, д. 30, Россия*

* *E-mail: kocheg@mail.ru*

** *E-mail: martynova@tpu.ru*

Поступила в редакцию 07.05.2024 г.
После доработки 30.06.2024 г.
Принята к публикации 30.06.2024 г.

Целью данной работы является развитие технологии сплайн-адаптивных фильтров (САФ) для реализации в реальном времени. Предложенный в работе Р-САФ на базе рекуррентного штрафного Р-сплайна по аналогии с классическим САФ состоит из линейной динамической и нелинейной статической компонентов. Для адаптации Р-САФ разработаны вычислительные схемы с различной топологией, что одновременно определяет способ адаптации узлов и вычисления коэффициентов сплайна. Это позволило повысить эффективность Р-САФ по сравнению с классическим САФ и сократить вычислительные затраты. Показатель эффективности *MSE* [*dB*] для Р-САФ при анализе модельных и реальных временных рядов оказался на уровне и выше классического САФ.

*Ключевые слова*: Р-сплайн, сплайн-адаптивный фильтр, аппаратная функция

**DOI:** 10.31857/S0132347424060036, **EDN:** DYUOIY

## 1. ВВЕДЕНИЕ

В последние годы возрос интерес научного и инженерного сообщества к нелинейным и адаптивным моделям, а также к прикладным задачам на их основе. Обоснован подобный интерес нелинейной природой многих процессов реальной жизни.

Нелинейные адаптивные модели и фильтры обладают хорошей гибкостью и высокой производительностью [1–3]. Нелинейность моделей отражает нестационарную природу процессов, а адаптивность моделей повышает эффективность их применения. Вычислительные затраты определяются, главным образом, принципами адаптации и обучения модели.

Одна из популярных идей при создании нелинейных фильтров основана на обновлении (адаптации) коэффициентов линейных фильтров. Наиболее популярные алгоритмы такой адаптации основаны на методе наименьших квадратов и его модификациях, а также методе аффинной проекции. Первая группа методов имеет небольшую вычислительную сложность, вторая – хорошую сходимость.

Другие идеи адаптивного обучения основаны на нейронных сетях [5], адаптивных фильтрах Вольтерра [4], ядра [7], функциональной связи [8], расширенном фильтре Калмана [6] и пр. Однако подобные фильтры эффективны только для объектов со слабой нелинейностью. А значительная нелинейность отрицательно сказывается на сходимости алгоритмов адаптации и усложняет вычисления, поскольку связана с увеличением порядка модели.

Системы реального времени наиболее требовательны к быстродействию алгоритмов обработки информации и, соответственно, к их вычислительной сложности. Поэтому в таких системах адаптивные модели с этапом обучения или с использованием численных методов мало пригодны.

В этой ситуации возможен подход с использованием адаптивных сплайнов, интерес к которым возрастает, как к инструменту нелинейного моделирования. Сама идея, а также термин “сплайн-адаптивный фильтр” (САФ) были введены в 2013 г. в работе [9] M. Scarpiniti. Концепция САФ включает последовательную комбинацию линейного фильтра и нелинейного алгоритма с функцией адаптации. Нелинейная часть в базовой конструкции представлена интерполяционным сплайном, структура которого остается неизменной. Для интерполяции используются базисные сплайны и сплайны Катмалла–Рома с фиксированной матрицей коэффициентов для локального звена сплайна.


**Рис. 1.** Структура сплайн-адаптивного фильтра.

Структурно модель САФ относится к блочно-ориентированному представлению и включает линейные и нелинейные блоки [10, 11]. Линейная часть модели является временно-инвариантной (динамической), а нелинейная модель – статической (рис. 1).

Топология блоков в САФ также может быть различной. Например, модель САФ [9], называемая моделью Винера, представляет собой линейно-нелинейную (ЛН) модель, включающую линейный фильтр и статическую нелинейную функцию адаптации. Другая популярная модель, известная как Хаммерштейн-модель, является нелинейно-линейной (НЛ) моделью, в которой динамический и нелинейный статический блок имеют обратный порядок [12]. Также существуют модели, которые комбинируют компоненты ЛН и НЛ, обеспечивая гибкость и разнообразие в работе с различными типами нелинейностей.

Концепция САФ оказалась довольно продуктивной и для теоретических исследований, и в прикладных задачах. Разными исследователями были разработаны и изучены различные варианты САФ, например, с использованием БИХ-фильтров [13, 14], для активного контроля и фильтрации разного типа помех [15, 16], для негауссовой среды [17, 18]. Ряд работ посвящен вопросам улучшения устойчивости, сходимости, анализу надежности и производительности [18–20].

Тем не менее, несмотря на большое количество работ по САФ, существуют определенные вопросы относительно его применения на практике.

САФ достаточно разработаны теоретически и опираются на априорную информацию о свойствах входных сигналов и помех. При адаптации параметров САФ используются градиентные методы оптимизации [11, 18]. Однако в реальной задаче желаемый сигнал часто неизвестен, а целевая функция обычно мультимодальна. Еще одна проблема САФ связана с длительностью машинного обучения, что затрудняет его использование в режиме реального времени.

Целью данной работы является развитие технологии САФ для реализации в реальном времени. Для этого использована модификация штрафного Р-сплайна, названная здесь Р-САФ. В отличие от традиционных Р-сплайнов, вместо одного параметра сглаживания, предлагаемый Р-САФ позволяет изменять этот параметр в пределах отдельного звена сплайна. А создание группы отсчетов ВР решает проблему выбора узлов сплайна.

Другая особенность Р-САФ состоит в экономичной вычислительной схеме Р-САФ в виде рекуррентных алгебраических выражений, что позволяет использовать его в реальном времени.

И наконец, модель Р-САФ представляет собой аналитическое выражение, что повышает интерпретируемость моделей на его основе.

## 2. ОПИСАНИЕ И ТОПОЛОГИЯ Р-САФ

Для реализации САФ в реальном времени предлагается математическая модель в форме рекуррентной сплайн-функции. Именно рекуррентное математическое описание делает возможным использование САФ в РРВ.

Большинство известных САФ основаны на сглаживающих сплайнах, однако также возможно использование штрафных Р-сплайнов и базисных В-сплайнов [21, 22]. При этом сглаживающие сплайны имеют большую вычислительную сложность, что становится проблемой при реализации в реальном времени [23]. А, например, впервые предложенная в [9] модель САФ использует базисные В-сплайны. Для базисных и штрафных сплайнов оптимальность выбора узлов оказывает значительное влияние на сложность их реализации [24]. И хотя адаптация в реальном времени является трудоемкой процедурой, эффективность САФ при этом существенно повышается.

В теории цифровой фильтрации можно выделить два подхода: цифровой фильтр (ЦФ) с конечной импульсной характеристикой (КИХ) и бесконечной импульсной характеристикой (БИХ). В САФ, в основном, применяются КИХ-фильтры. Их достоинствами являются линейная фазовая характеристика и устойчивость. Эти и другие достоинства КИХ объясняются отсутствием обратной связи по выходным параметрам фильтра.

Структура БИХ-фильтров содержит обратную связь, что является причиной неустойчивости и, нередко, низкой сходимости. Однако БИХ-фильтры, в отличие от КИХ-фильтров, могут обеспечить резкость переходной зоны пропуска и подавления сигнала при одинаковом с КИХ-фильтрами порядке ЦФ [25].

Одним из способов описания ЦФ является разностное уравнение. В данной работе используется рекуррентная форма штрафного Р-сплайна, полученная с применением вариационного подхода [26]. В классическом варианте сглаживающий кубический сплайн $S(t)$ может быть получен как решение задачи минимизации на всем интервале наблюдения $[a,b]$ по отсчетам $y_i$, $i=\overline{1,n}$:

$$S_\lambda = \underset{s\in W_2^2[a,b]}{\arg}\min_{\lambda}\left\{\lambda\left\|S''\right\|^2_{L_2[a,b]}+\sum_{i=1}^{n}\left|S(t_i)-y_i\right|^2\right\}, \quad (1)$$

где $\lambda$ – сглаживающий множитель, ассоциирован с параметром регуляризации Тихонова. А сами слагаемые в (1) определяют соответственно минимальную кривизну сплайна и минимум невязок [27]. Кубический сплайн $S(t)$ находится среди всех функций из пространства Соболева $s\in W_2^2[a,b]$. Степень гладкости или штраф за гладкость сплайна $S(t)$ определяется параметром $\lambda$. Отсюда и название штрафной Р-сплайн. Диапазон параметра $\lambda$ неизвестен и обычно велик $[10^{-9}, 10^9]$, но при $\lambda\to 0$ сплайн стремится к интерполяционному.

Для реализации Р-сплайна в РРВ критерий (1) модифицирован нами и адаптирован отдельно для каждого $i$-го звена сплайна:

$$S_{\rho,h} = \underset{s\in W_2^2[0,h]}{\arg}\min_{\rho,h}\left\{(1-\rho)\left\|S''\right\|^2_{L_2[0,h]}+\rho\sum_{j=0}^{h}\left|S\left(t_j^i\right)-y_j^i\right|^2\right\}. \quad (2)$$

Входные данные в (2) собраны в группы по $h$ значений $\left\{y_0^i, y_1^i, \ldots, y_h^i\right\}$ между крайними отсчетами группы $t_0^i$, $t_h^i$ для каждого $i$ -го звена сплайна.

Предлагаемый Р-САФ позволяет изменять гладкость в пределах звена с помощью переменного параметра гладкости $\rho$, тем самым повышая гибкость нелинейных моделей отдельных звеньев. В отличие от традиционных Р-сплайнов (1) с единым параметром сглаживания $\rho$.

Вид критерия (2) известен как блочная регуляризация Тихонова [28], которая в данном случае определяет блок, как группу из $h$ отсчетов. Минимальный размер группы $h=3$ (т. е. 4 отсчета ВР) определен порядком кубического сплайна. Но обычно $h>3$ и это означает, что выборка избыточна и это положительно сказывается на равновесии данных для описания нелинейности [10].

Другой особенностью критерия (2) является адаптация штрафного параметра $\rho$ в интервале $\rho\in[0,1]$. Нормирование сглаживающего сомножителя $\rho$ уменьшает сложность его выбора в соответствии с физическим смыслом: от максимальной гладкости при $\rho=0$ до интерполирующего сплайна при $\rho=1$.

Переход от критерия (2) к функционалу $J(S)$ упрощает получение неизвестных коэффициентов сплайна:

$$J(S) = (1-\rho)(h\Delta t)^2\int_{t_0^i}^{t_h^i}\left[S''(t)\right]^2 dt + \rho\sum_{k=0}^{h}\left[S\left(t_k^i\right)-y\left(t_k^i\right)\right]^2. \quad (3)$$

Шаг дискретизации $\Delta t$ уравновешивает размерности слагаемых, а сам функционал (3) становится безразмерным. Далее $\Delta t = 1$.

Для получения коэффициентов $a_0^i$, $a_1^i$, $a_2^i$, $a_3^i$ рекуррентного сплайна $S_i(\tau)$ на $i$-м звене

$$S_i(\tau) = a_0^i + a_1^i\tau + a_2^i\tau^2 + a_3^i\tau^3,\ -q\le\tau\le h-q \quad (4)$$

использованы два типа условий:

1) условия равенства непрерывных производных для смежных звеньев сплайна $S^{(k)}(t_{q+1}^{i-1})_+ = S^{(k)}(t_q^i)_-$ позволяют найти рекуррентные соотношения для непрерывных коэффициентов $a_0^i$, $a_1^i$ ($k=0, 1$) смежных ($i-1$)-х и $i$-х звеньев;

2) из условия $\frac{\partial J(S)}{\partial a_2^i}=0, \frac{\partial J(S)}{\partial a_3^i}=0$ найдены разрывные коэффициенты $a_2^i$, $a_3^i$.

Математические соотношения для коэффициентов Р-сплайна $a_0^i$, $a_1^i$, $a_2^i$, $a_3^i$ представляют собой алгебраические выражения [26] и не требуют дополнительных методов решения (аналитических или численных).

Отличительная особенность предлагаемого Р-сплайна (4) – это возможность сопряжения смежных звеньев в любой точке $q=t_k^i$ внутри $i$-го звена ($k=\overline{0,h}$) (рис. 2а). Это особенность уникальна и для теории сплайнов с последовательным сопряжением звеньев, и для реализации в реальном времени со скользящим окном.

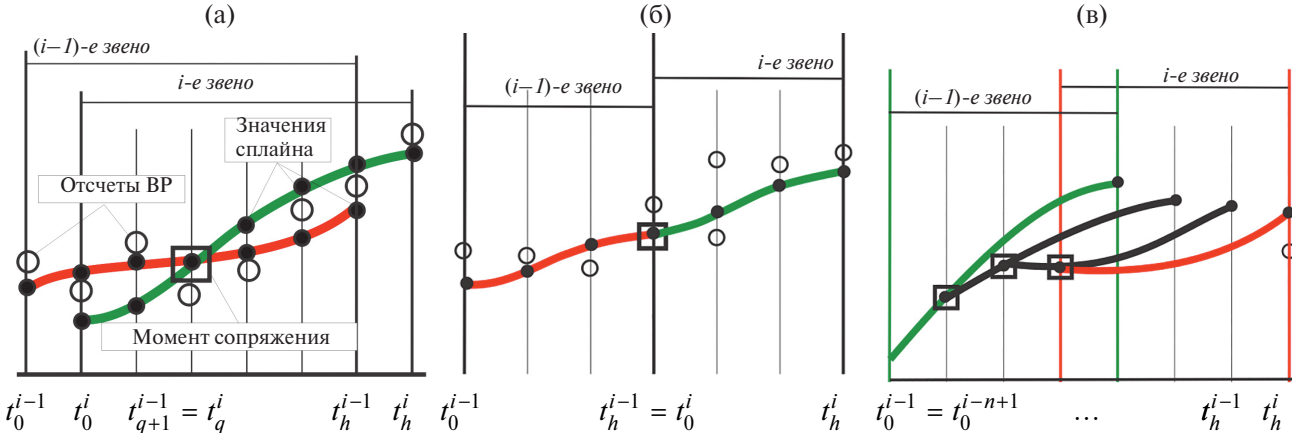



**Рис. 2.** Топология вычислительных схем Р-САФ.

Временные моменты сопряжения сплайна $q$ и вычисления $\tau$ являются параметрами вычислительных схем Р-САФ. И на основе их взаимного расположения можно разработать несколько топологий для вычислительных схем сплайна [29], в том числе последовательной рис. 2б и многократной фильтрации рис. 2в. И все три схемы сочетают рекуррентность коэффициентов сплайна, что соответствует адаптации линейного КИХ-фильтра в традиционном САФ, и локальность к группе отсчетов внутри звена.

Наибольший интерес с позиции РРВ имеет универсальная вычислительная схема, показанная на рис. 2а. Для произвольных значений $q$ и $\tau$ разностное уравнение ЦФ, соответствующего такой схеме, представляет собой уравнение с переменными параметрами, порядок уравнения соответствует значению $\tau$ [30]. И именно параметры $q$ и $\tau$, как параметры топологии, определяют структурную адаптацию Р-САФ.

Рассмотрим частный случай данной вычислительной схемы для $\tau = q+1$, $q = \overline{0, h-1}$. Для получения разностного уравнения требуется задание единого временного отсчета, определенного для каждого $i$-го звена сплайна. Принципиально таким моментом может быть любой отсчет $i$-го звена $t_j^i$, $j = \overline{0,h}$. Здесь выбран момент времени $t_\tau^i$ и далее для этого момента введено обозначение $i = t_\tau^i$.

Проанализируем компоненты функционала (3) относительно выбранного момента. Если введены обозначения $y_i = y(t_\tau^i)$, $S_i = S(t_\tau^i)$, то последовательности $\{S_i\}$, $\{y_i\}$, $i = 1, 2, 3...$ можно рассматривать, как решетчатые функции с интервалом квантования $\Delta t$. И для введенных обозначений разностное уравнение Р-САФ имеет следующий вид:

$$\gamma_0 S_i - \gamma_1 S_{i-1} - \gamma_2 S'_{i-1} = \gamma_3 \sum_{k=i}^{i+h} y_{k-1}(k-1)^2 + \gamma_4 \sum_{k=i}^{i+h} y_{k-1}(k-1)^3, \quad (5)$$

где

$$\gamma_0 = BC - A^2;$$
$$\gamma_1 = \alpha_0 + \rho\left(AH_3 - CH_2\right) + \rho\left(AH_2 - BH_3\right);$$
$$\gamma_2 = \alpha_0 + \rho\left(AH_4 - CH_3\right) + \rho\left(AH_3 - BH_4\right);$$
$$\gamma_3 = \rho\left(C - A\right);\ \gamma_4 = \rho\left(B - A\right);$$
$$A = 6(1-\rho)h^4 + \rho H_5;$$
$$B = 4(1-\rho)h^3 + \rho H_4;$$
$$C = 12(1-\rho)h^5 + \rho H_6;$$
$$H_n = \sum_{k=0}^{h} k^n.$$

Разностное уравнение (5) – наиболее простой случай разностного уравнения для Р-САФ и соответствует значениям $\tau = q+1$, $q = 0$. Это уравнение первого порядка с постоянными коэффициентами $\gamma_j$, $j = \overline{0,4}$. Если $\tau > q+1$ и для всех звеньев сплайна $\tau$=const, то разностное уравнение остается с постоянными параметрами, но порядок уравнения равен $(\tau - h)$. В данном случае коэффициенты $\gamma_j$, $j = \overline{0,4}$ разностного уравнения (5) не зависят от параметров вычислительной схемы $q$ и $\tau$, а зависят только от параметров самого сплайна $h$ и $\tau$. Эти параметры и определяют адаптивные свойства Р-САФ, т.е. участвуют в процессе параметрической адаптации.

## 3. УСТОЙЧИВОСТЬ И ПЕРЕХОДНЫЕ ПРОЦЕССЫ Р-САФ

Концепция САФ оказалась довольно популярной и это можно объяснить способностью цифровых фильтров моделировать нелинейные

системы при невысокой сложности самих САФ. Однако скорость сходимости САФ остается по-прежнему недостаточно высокой [31].

Рекуррентный Р-САФ как математический инструмент обработки информации в РРВ должен соответствовать требованиям сходимости, устойчивости и точности [32]. Для оценки эффективности Р-САФ в установившихся и переходных режимах целесообразно использовать методы линейных динамических систем. Как и в случае любого ЦФ для описания Р-САФ используется математический аппарат, включающий аппаратную и системную функции фильтра.

В отличие от КИХ-фильтров, обладающих линейной фазой и устойчивостью, БИХ-фильтры могут оказаться неустойчивыми. Поэтому в обязательном порядке следует анализировать устойчивость рекуррентного сплайн-фильтра в области изменения его параметров.

На основе $z$-преобразования правой и левой частей разностного уравнения (5) аналитически получена системная функция сплайн-фильтра $W(z) = S(z)/Y(z)$ [31]:

$$W(z) = \frac{\sum_{k=0}^{h} z^{k-1} \cdot \left((k-1)^2\gamma_3 + (k-1)^3\gamma_4\right)}{\gamma_0 - \gamma_1 z^{-1} - \gamma_2 \ln(z) z^{-1}}, \quad (6)$$

которая является аналогом частотной передаточной функции (ПФ) непрерывных систем.

И несмотря на несложный вид системной функции, синтез Р-САФ непосредственно на его основе довольно проблематичен. Альтернативным способом синтеза Р-САФ является представление ПФ в виде соединения элементарных звеньев САУ с прямыми, параллельными или каскадными связями [33]. Для этого представим системную функцию (6) в традиционном виде отношения полиномов:

$$W(z) = \frac{\beta_0 z^{-1} + \beta_1 + \beta_2 z + \beta_3 z^2 + \ldots + \beta_h z^{h-1}}{1 + \alpha_0 z^{-1} + \alpha_1 \ln(z) z^{-1}} \quad (7)$$

с коэффициентами

$$\alpha_0 = -\frac{\gamma_1}{\gamma_0};\ \alpha_1 = -\frac{\gamma_2}{\gamma_0};\ \beta_k = \gamma_3(k-1)^2 + \gamma_4(k-1)^3.$$

Тогда структурная схема прямой реализации Р-САФ будет выглядеть следующим образом (рис. 3).

Обратная связь в структуре Р-САФ, являясь достоинством рекурсивных ЦФ, может привести к его неустойчивости, т. е. наличию корней характеристического уравнения за пределами единичного круга $|z| < 1$. Для оценки областей устойчивости запишем характеристический полином ПФ (6) с учетом билинейного $w$-преобразования. И, заменив $z = \frac{1+w}{1-w}$, получим

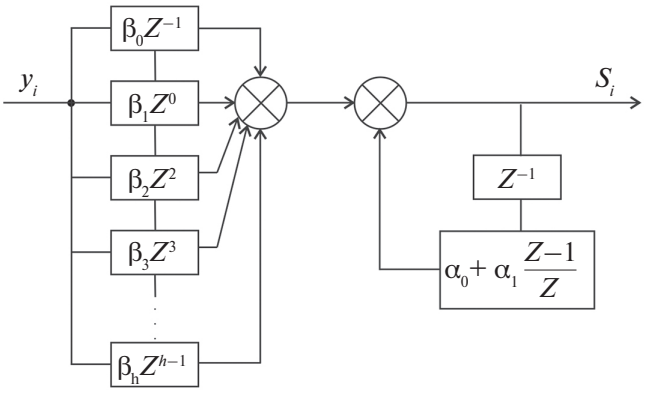



**Рис. 3.** Структурная схема рекуррентного Р-САФ.

$$u_0 w^2 + u_1 w + u_2 = 0, \quad (8)$$

где

$$u_0 = 4\gamma_1;$$
$$u_1 = \gamma_0 + \gamma_1 - 2\gamma_2;$$
$$u_2 = 2\gamma_0 - \gamma_1 - \gamma_2.$$

Для уравнения второго порядка (8) критерий Гурвица–Мизеса определяет условия устойчивости дискретных систем:

$$4\gamma_1 > 0;$$
$$\gamma_0 + \gamma_1 - 2\gamma_2 > 0;$$
$$2\gamma_0 - \gamma_1 - \gamma_2 > 0.$$

С учетом введенных в (5) обозначений для коэффициентов $\gamma_j, j = \overline{0,2}$ разностного уравнения устойчивость Р-САФ полностью определяется параметрами сплайна $h$ и $\tau$. И для выбранной топологии вычислительной схемы $q = 0, \tau = 1$ Р-САФ устойчив при любых значениях сглаживающего параметра $\rho \in [0,1]$ при любом размере группы $h$.

На рис. 4 области неустойчивости не заштрихованы, области абсолютной устойчивости (т. е. при всех значениях $\rho \in [0,1]$) заштрихованы полностью, а на областях частичной устойчивости приведена нижняя граница диапазона $\rho$ при некоторых соотношениях $h$ и $q$. Изменение параметра топологии $q > 0$ заметно сужает области устойчивости Р-САФ. Устойчивость наблюдается только при стремлении $\rho \to 1$, и сами диапазоны $\rho$ весьма малы: [0.99–1], [0.84–1]. И наконец при $q \to h$ Р-САФ всегда неустойчив.

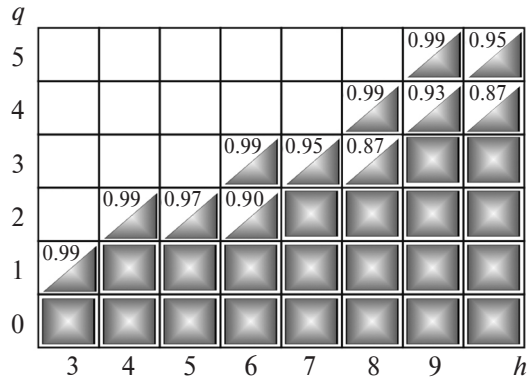



**Рис. 4.** Области устойчивости Р-САФ.

При сопряжении смежных звеньев в начале текущего звена ($q=0$) Р-САФ всегда остается устойчив и при $\tau > 1$. В практических приложениях сопряжении в начале звена и используется чаще всего.

Также целесообразно оценить сходимость Р-САФ на основе косвенных критериев, т. е. исследовать аппаратную функцию $g(t)$. Известно, что аппаратная функция ЦФ является функцией веса компонент фильтра во времени. И по мере увеличения длины веса ЦФ резко возрастает вычислительная сложность процессов фильтрации и адаптации.

Аппаратная функция $g(t)$ Р-САФ была получена аналитически на основе системной функции Р-САФ (6) с использованием обратного преобразования Фурье.

Являясь аналогом импульсной весовой функции непрерывной системы, аппаратная функция выражает аналитическую зависимость сигналов между входными и выходными сигналами ЦФ на основе уравнения дискретной свертки. Визуально аппаратная функция Р-САФ несимметрична, что типично для БИХ-фильтров (рис. 5а). А затухающий характер подтверждает сходимость Р-САФ при изменении параметров сплайна ρ и $h$.

Однако условие каузальности ($g(t) = 0$, $t < 0$) соблюдается только в случае, если Р-САФ работает в режиме без задержки, т. е. не имеет запаздывания по параметрам топологии вычислительной схемы. В других случаях аппаратная функция отлична от нуля, что характерно для систем с запаздыванием, например, при обработке данных группами.

Системную ошибку ЦФ определяет ширина аппаратной функции Δ (рис. 5в). Количественно ширина аппаратной функции может быть оценена следующим соотношением [34]:

$$\Delta = \frac{\int_{-\infty}^{\infty} |g(t)|\,dt}{g(0)}. \qquad (9)$$

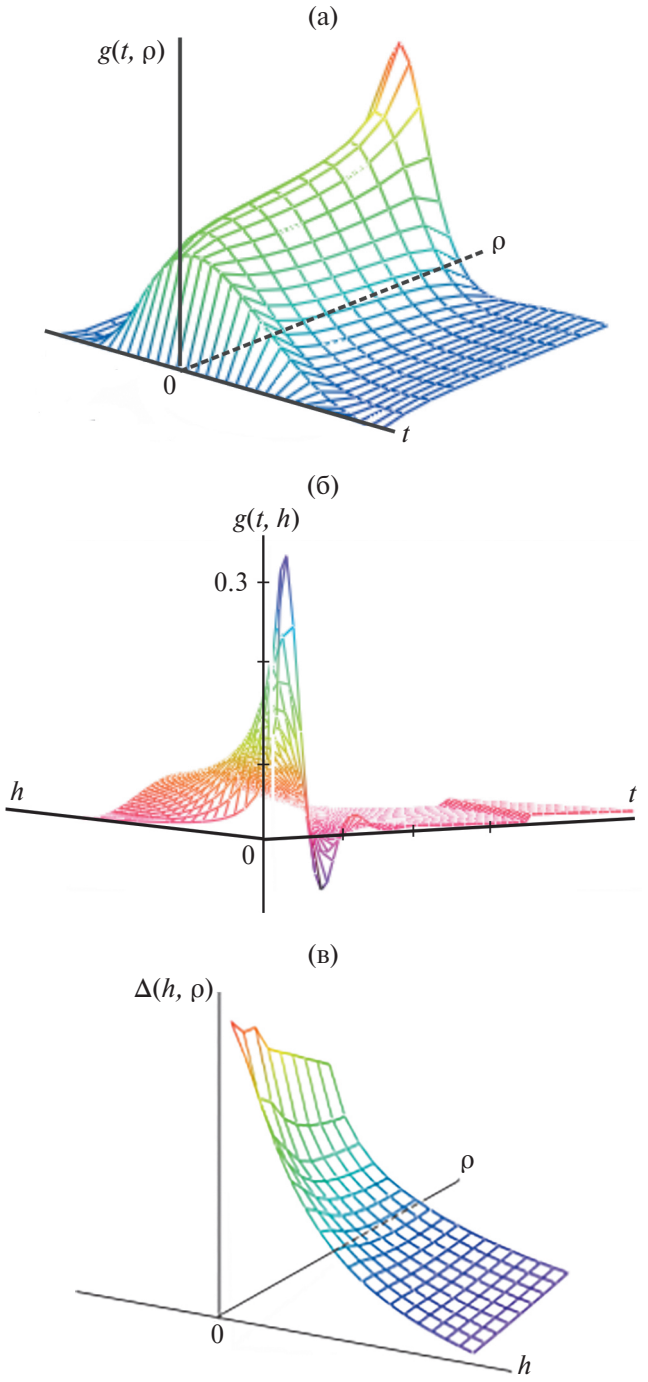



**Рис. 5.** Аппаратная функция Р-САФ.

Как видно на геометрической иллюстрации ширина аппаратной функции зависит от параметров сплайна ρ и $h$. Причем ширина уменьшается и с ростом сглаживающего множителя ρ, и с ростом длины сплайна $h$. С увеличением числа отсчетов звена $h$ полоса пропускания фильтра уменьшается. Соответственно уменьшается и ширина аппаратной функции, что видно из рисунка. Гладкость сплайна на выходе фильтра в таких случаях увеличивается, однако он

становится значительно отдален от линии регрессии, что приводит к увеличению систематической ошибки. Таким образом, ширина амплитудной функции физически интерпретируется как фактор, влияющий на точность измерений.

Влияние сглаживающего множителя ρ много слабее, чем параметра $h$. И при $h > 15$ влияние параметра сглаживания практически нет.

## 5. РЕЗУЛЬТАТЫ ЧИСЛЕННЫХ ЭКСПЕРИМЕНТОВ

Для оценки эффективности предлагаемого Р-САФ выполнена серия вычислительных экспериментов с реализацией в режиме реального времени. Численные эксперименты предназначены для демонстрации сглаживающих и фильтрующих возможностей Р-САФ, а также для сопоставления результатов с другими САФ.

Результаты предложенного Р-САФ, основанного на штрафном сплайне реального времени, сравниваются с классическими вариантами САФ на основе SAF-LMS [9].

Для проведения исследований в качестве модельных и реальных данных целесообразно выбрать широко известные функции с выраженными нелинейностями, которые часто используются в подобных исследованиях алгоритмов САФ.

Для оценки эффективности САФ [18] используется показатель точности, основанный на среднеквадратическом отклонении и выраженный в децибелах:

$$MSE\left[dB\right] = 10\log_{10} E\left[e^2(n)\right], \qquad (10)$$

где $E[*]$ – среднее значение; $e(n)$ – разность полезного и восстановленного сигналов.

**Модельные входные сигналы**

Все результаты получены путем усреднения 20 испытаний Монте-Карло. Максимальный объем выборки для всех модельных сигналов – 30 000 отсчетов.

На рис. 6 представлены результаты эффективности предлагаемого Р-САФ для двух наиболее популярных примеров в теории САФ. Для рис. 6а входной полезный сигнал $x_n$ представляет собой гауссовский случайный процесс и генерируется соотношением [9]

$$x_n = rx_{n-1} + \sqrt{1 - r^2} \cdot \nu_n,$$

где $\nu_n$ – белый гауссовский шум с нулевым средним с единичной дисперсией; $r \in [0, 1)$ – коэффициент, определяющий корреляцию между соседними входными отсчетами $x_n$.

Кроме того, ко входным данным добавляется независимый белый гауссов шум $\xi_n$ с различными соотношениями сигнал/шум ($SNR = 10, 20, 30, 40$ $dB$).

Рис. 6б отражает эффективность фильтрации процесса, порожденного альфа-стабильным распределением, и для $\alpha \neq 1$ имеет следующий вид [17, 18]:

$$f(t) = \exp\left\{j\rho t - \gamma\left|t\right|^{\alpha}\left[1 + j\beta\,\mathrm{sign}(t)\tan\left(\frac{\alpha\pi}{2}\right)\right]\right\},$$

где $\alpha \in (0, 2]$ – индекс стабильности, определяющий выраженность импульса; $-1 \leqslant \beta \leqslant 1$ – индекс симметрии; ρ – параметр положения; $\gamma > 0$ – параметр дисперсии. Очевидно, что при $\alpha = 2$ имеет место гуссовский сигнал. И по аналогии для остальных значений $\alpha \in (0, 2]$ сигнал называют

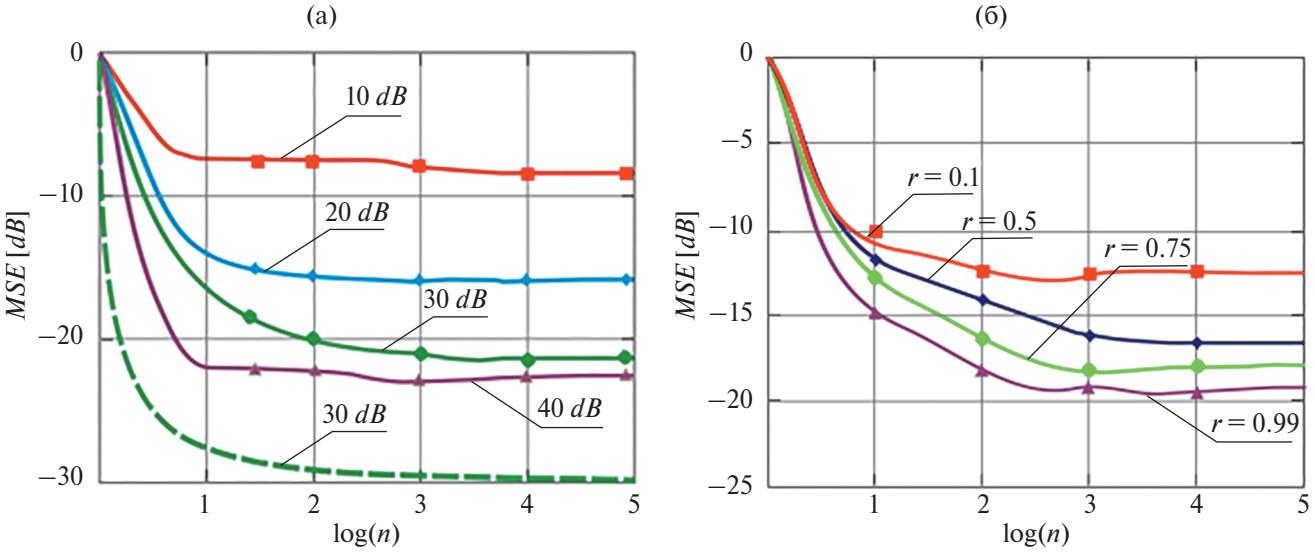



**Рис. 6.** Эффективность алгоритма Р-САФ в условиях гауссовского случайного процесса.

негауссовским. В эксперименте данный сигнал является полезным и для него заданы значения параметров $\alpha = 1.6$, $\beta = 0$, $\rho = 0$, $\gamma = 0.05$. Аддитивная помеха – белый гауссов шум $\xi_n$ с различными соотношениями сигнал/шум ($SNR = 10, 20, 30, 40\ dB$).

Кривые $MSE$ [$dB$] позволяют оценить общую эффективность ($MSE$ [$dB$] приближение к установившемуся значению $MSE$) и скорость сходимости алгоритмов.

Рис. 6б демонстрирует хорошую эффективность Р-САФ в условиях негауссовского входного сигнала. Графики $MSE$ [$dB$] подтверждают сходимость алгоритма к значению установившейся ошибки при различных соотношениях сигнал/шум. И в установившемся режиме графики $MSE$ [$dB$] асимптотически стремятся к значению мощности шума [9]. При заданных уровнях сигнал/шум ($SNR = 10, 20, 30, 40\ dB$) они теоретически равны ($-10, -20, -30, -40\ dB$) соответственно.

Однако в случае гауссовского полезного сигнала (рис. 6а) графики $MSE$ [$dB$] довольно далеки от теоретических значений поэтому и алгоритм Р-САФ имеет значительную установившуюся ошибку. Для сравнения на рис. 6а пунктирной линией приведена кривая $MSE$ [$dB$] для классического САФ [13] при $SNR = 30\ dB$. Установившаяся ошибка согласуется с мощностью сигнала, но при довольно большом числе отсчетов $n > 2000$ (рис. 5 в [13]).

Следующим модельным сигналом, часто используемым при анализе ВР является Доплеровская функция, определенная на интервале [0, 1]:

$$f(t) = 5\sqrt{t(t-1)}\sin\left(\frac{2\pi(1+0{,}05)}{t+0{,}05}\right).$$

Аддитивная помеха в данном случае также представлена белым гауссовским шумом $\xi_n$ с различными соотношениями сигнал/шум ($SNR = 10, 20, 30, 40\ dB$). На графике (рис. 7а) показан полезный сигнал (серая линия), смесь сигнала и шума (черные точки) и результат обработки алгоритмом Р-САФ (синяя линия) на интервале [0, 1] при $SNR = 10\ dB$ для 1000 отсчетов ВР. Параметры алгоритма Р-САФ приведены на рисунке. В зумированной области представлен результат работы алгоритма (красная линия) на интервале [0, 0.3] при значениях параметров Р-САФ $h = 3$, $\rho = 0.5$. Для подобных сигналов со значительным изменением и частоты, и амплитуды параметры Р-САФ оказывают существенное влияние на эффективность его работы. На графике (рис. 7б) отображены кривые $MSE$ [$dB$] для разных значений соотношения сигнал/шум при $h = 3$, $\rho = 0.5$. Сходимость алгоритма подтверждается приближением графиков $MSE$ [$dB$] к установившемуся значению. Причем скорость сходимости для слабого шума с $SNR = 40\ dB$ на порядок ниже, чем с низким значением $SNR = 10\ dB$.

**Реальные временные ряды**

В качестве входных данных использованы два реальных набора данных из репозитория *DaISy* [11]. Эта БД содержит большое количество реальных статистических данных из разных отраслей: механические системы, биомедицинские, промышленные процессы, экологические и др.

Набор данных № 96-008 “Данные о флаттере крыла” содержит 1024 значения. На рис. 8а серым цветом показана информативная часть ВР – 512 значений. Флаттер крыла – это колебания крыла самолета во время полета. Характеризуется флаттер высокочастотными колебаниями и влияет на безопасность полета. На рисунке красной линией отображена работа алгоритма Р-САФ при параметрах алгоритма $h = 3$, $\rho = 0.8$. При обработке подобных высокочастотных сигналов

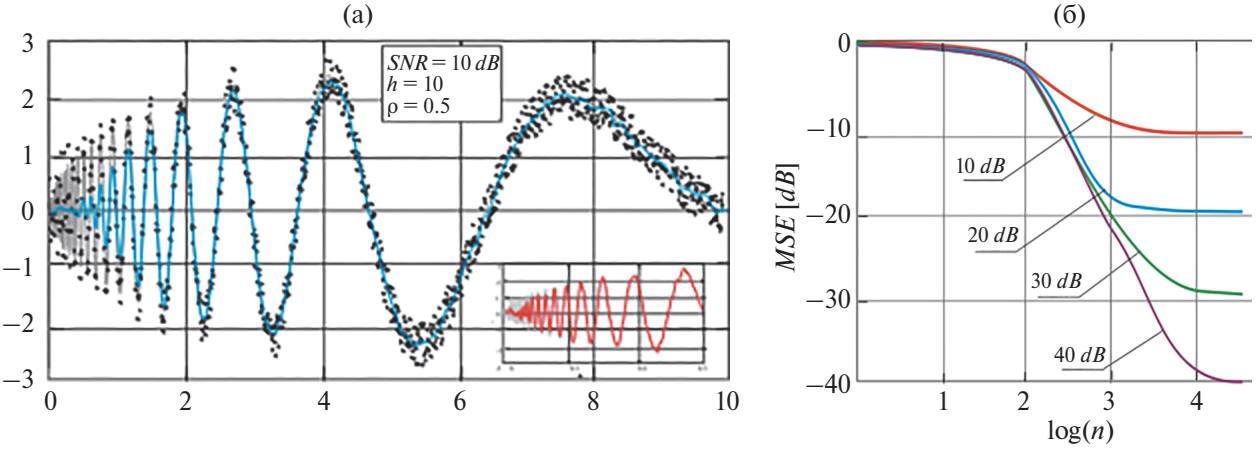



**Рис. 7.** Эффективность алгоритма Р-САФ для Доплеровской функции.

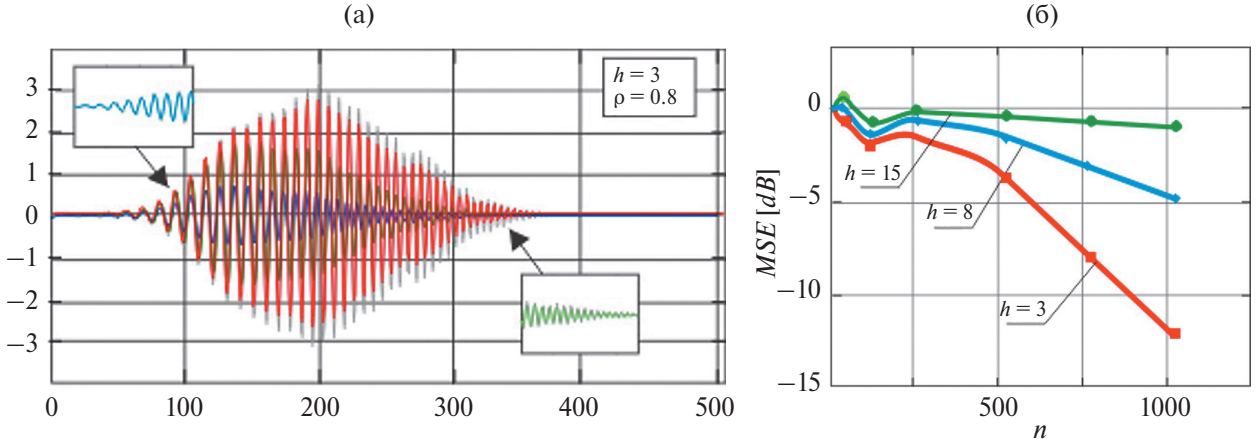



**Рис. 8.** Эффективность алгоритма Р-САФ для набора данных № 96-008 DaISy.

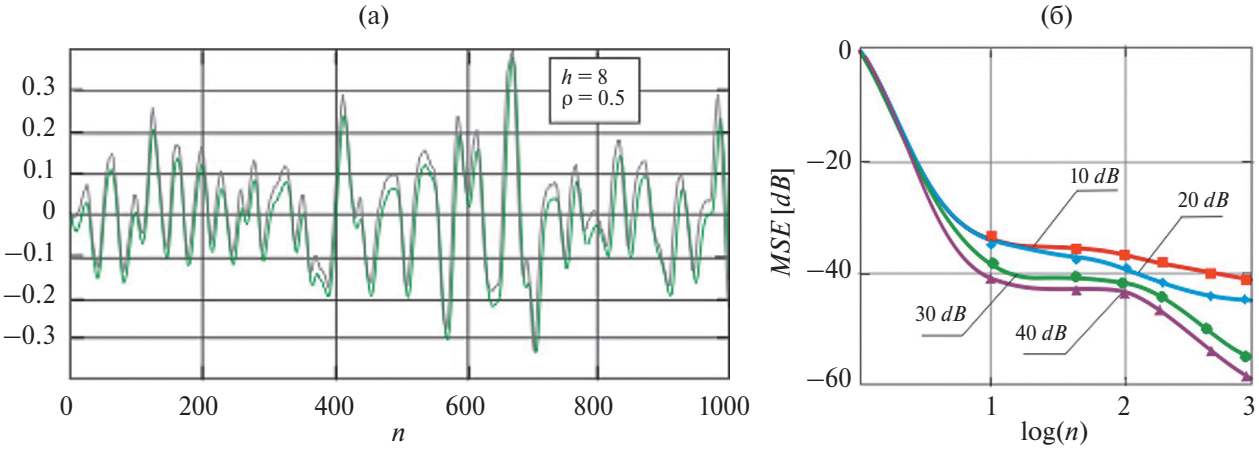



**Рис. 9.** Эффективность алгоритма Р-САФ для набора данных № 96-004 DaISy.

существенную роль имеют параметры настройки Р-САФ. Влияние параметров сплайна взаимное и сочетанное. В предложенном алгоритме Р-САФ параметр ρ в большей степени определяет амплитуду восстановленного сигнала, в то время как параметр $h$ влияет на частоту. При значении $h=3$ наблюдается полное совпадение сигналов по частоте во всем временном диапазоне: синяя линия в диапазоне отсчетов [60÷160], зеленая – [250÷350]. Кроме того, на рис. 8б отображено влияние размера группы отсчетов $h$ на точность восстановления сигнала. Очевидно, что точность повышается при уменьшении параметра $h$.

Еще один набор данных из репозитория *DaISy* № 96-004 “Данные шаровой установки *SISTA* (система оценки информационной безопасности)” содержит 1000 значений ВР. На рис. 9а исходный набор данных показан серой линией, а результат работы алгоритма – зеленой. Расширяя исследования [11], ВР здесь был дополнительно исследован в условиях аддитивной помехи в виде белого гауссовского шума с различными соотношениями сигнал/шум ($SNR$ = 10, 20, 30, 40 $dB$). Рис. 9б показывает хорошую сходимость алгоритма Р-САФ для всех заданных соотношений сигнал/шум.

При отсутствии шума значение погрешности $MSE$ [$dB$] варьируется в диапазоне (–55, –60) $dB$ при различных соотношениях параметров $h$ и ρ. И эти значения можно рассматривать как систематическую погрешность предложенного алгоритма Р-САФ для заданного ВР.

## 6. ЗАКЛЮЧЕНИЕ

БИХ-фильтры привлекают внимание исследователей благодаря широкому спектру прикладных задач при обработке данных в реальном времени. Особую актуальность БИХ-фильтры на базе сплайнов приобрели, как инструмент нелинейной обработки, известный, как САФ.

Предложенный в работе Р-САФ на основе рекуррентного Р-сплайна по аналогии с классическим САФ M. Scarpiniti [9] состоит из линейной динамической и нелинейной статической компонент. Для адаптации Р-САФ разработаны

вычислительные схемы с различной топологией, что одновременно определяет способ адаптации узлов и вычисления коэффициентов сплайна. Это повышает эффективность Р-САФ по сравнению с классическим САФ и сокращает вычислительные затраты.

Был проведен анализ частотных и временных характеристик рекурсивного Р-САФ, а также изучены условия его сходимости. Установлено, что при изменении параметров Р-САФ остается низкочастотным.

Сравнительный анализ предложенного Р-САФ с другими САФ выполнен с использованием модельных и реальных данных из репозитария DAISY. Значения показателя *MSE* [*dB*] для Р-САФ оказались на уровне и выше классического САФ в случае высокочастотных детерминированных или реальных сигналов. В этом и проявляется преимущество Р-САФ: короткая рекурсивная часть и наличие аналитической модели.

## ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке Российского научного фонда (проект № 23-21-00259).

## СПИСОК ЛИТЕРАТУРЫ

1. *Haykin S.S.* Adaptive Filter Theory. 5th ed., Pearson Education, 2014. 912 p. ISBN: 9780132671453.
2. *Comminiello D., Príncipe J.C.* Adaptive learning methods for nonlinear system modeling. Oxford: Butterworth-Heinemann, 2018. 388 p. ISBN: 9780128129777.
3. *Shcherbakov M.A., Panov A.P.* Nonlinear filtering with adaptation to local properties of the image. Computer Optics. 2014. V. 38. № 4. P. 818–824. https://doi.org/10.18287/0134-2452-2014-38-4-818-824
4. *Wang Y.R., Cheng C.M.* Volterra series identification and its applications in structural identification of nonlinear block-oriented systems. International Journal of Systems Science. 2020. V. 51. № 11. P. 1959–1968.
5. *Solovyeva E.B.* Behavioural nonlinear system models specified by various types of neural networks. Journal of Physics: Conference Series. IOP Publishing. 2018. V. 1015(3). P. 032139.
6. *Fallahi K., Raoufi R., Khoshbin H.* An application of chen system for secure chaotic communication based on extended Kalman filter and multi-shift cipher algorithm. Commun. Nonlinear Sci. Numer. Simul. 2008. V. 13. № 4. P. 763–781.
7. *Liu W., Principe J.C., Haykin S.* Kernel Adaptive Filtering: A Comprehensive Introduction. Wiley, 2011. ISBN: 9780470447536.
8. *Comminiello D., Scarpiniti M., Azpicueta-Ruiz L.A., Arenas-Garcia J., Uncini A.* Functional link adaptive filters for nonlinear acoustic echo cancellation. EEE Trans. Audio Speech Lang. Process. 2013. V. 21. № 7. P. 1502–1512.
9. *Scarpiniti M., Comminiello D., Parisi F., Uncini A.* Nonlinear spline adaptive filtering. Signal Processing. 2013. V. 93. № 4. P. 772–783.
10. *Yang L., Liu J., Sun R. et al.* Spline adaptive filters based on real-time over-sampling strategy for nonlinear system identification. Nonlinear Dyn. 2021. V. 103. P. 657–675. https://doi.org/10.1007/s11071-020-05899-7
11. *Guan S., Biswal B.* Spline adaptive filtering algorithm based on different iterative gradients: Performance analysis and comparison. Journal of Automation and Intelligence. 2023. V. 2. № 1. P. 1–13.
12. *Cheng S., Wei Y., Sheng D., Wang Y.* Identification for Hammerstein nonlinear systems based on universal spline fractional order LMS algorithm. Commun. Nonlinear Sci. Numer. Simul. 2019. V. 79. P. 104901.
13. *Scarpiniti M., Comminiello D., Parisi R., Uncini A.* Nonlinear system identification using IIR Spline Adaptive Filters. Signal Processing. 2015. V. 108. P. 30–35.
14. *Wang Y., Ding F., Xu L.* Some new results of designing an IIR filter with colored noise for signal processing. Digital Signal Processing. 2018. V. 72. P. 44–58.
15. *Patel V., George N.V.* Multi-channel spline adaptive filters for non-linear active noise control. Applied Acoustics. 2020. V. 161. P. 107142.
16. *Liu C., Zhang Z., Tang X.* Sign-normalized IIR spline adaptive filtering algorithms for impulsive noise environments. Circu. Syst. Signal Process. 2019. V. 38. № 2. P. 891–903.
17. *Guo W., Zhi Y.* Nonlinear Spline Adaptive Filtering Against Non-Gaussian Noise. Circuits, Systems, and Signal Processing. 2022. V. 41. P. 579–596. https://doi.org/10.1007/s00034-021-01798-3
18. *Tao Y., Wenqi L., Yi Y., Rodrigo C.L.* Robust spline adaptive filtering based on accelerated gradient learning: Design and performance analysis. Signal Processing. 2021. V. 183. P. 107965.
19. *Gao Y., Zhao H., Zhu Y., Lou J.* The q-gradient LMS spline adaptive filtering algorithm and its variable step-size variant. Information Sciences. 2023. V. 658. P. 119983. https://doi.org/10.1016/j.ins.2023.119983
20. *Zhao H., Chen B.* Spline Adaptive Filter. In Book: Efficient Nonlinear Adaptive Filters. Chap. Springer: Cham. 2023. P. 163–208. https://doi.org/10.1007/978-3-031-20818-8_4
21. *Sharif S., Kamal S.* Comparison of Significant Approaches of Penalized Spline Regression (P-splines). Pakistan Journal of Statistics and Operation Research. 2018. V. XIV (2). P. 289–303.
22. *Lenz D., Yeh R., Mahadevan V., Grindeanu I., Peterka T.* Customizable adaptive regularization techniques for B-spline modeling. Journal of Computational Science. 2023. V. 71. P. 102037. https://doi.org/10.1016/j.jocs.2023.102037

23. *Budakçı G., Dişibüyük Ç., Goldman R., Oruç H.* Extending fundamental formulas from classical B-splines to quantum B-splines. Journal of Computational and Applied Mathematics. 2015. V. 282. P. 17–33.
24. *Goepp V., Bouaziz O., Nuel G.* Spline Regression with Automatic Knot Selection. arXiv: Applications. 2018. hal-01853459. P. 26.
25. *Tan L., Jiang J.* Digital Signal Processing. 3rd ed. Academic Press, 2019. 816 p. ISBN: 978-0-12-374090-8.
26. *Kochegurova E.A, Gorokhova E.S.* Current Derivative Estimation of Non-stationary Processes Based on Metrical Information. Lecture Notes in Computer Science. 2015. V. 9330. P. 512–519.
27. *Rozhenko A.I.* A new method for finding an optimal smoothing parameter of the abstract smoothing spline. J. Approx. Theory. 2010. V. 162. P. 1117–1127. https://doi.org/10.1016/j.jat.2009.08.002
28. *Sameni R.* Online filtering using piecewise smoothness priors: Application to normal and abnormal electrocardiogram denoising. Signal Processing. 2017. V. 133. P. 52–63.
29. *Kochegurova E.A., Kochegurov A.I., Rozhkova N.E.* Frequency Analysis of Recurrence Variational P-Splines. Optoelectronics, Instrumentation and Data Processing. 2017. V. 53. № 6. P. 591–598.
30. *Kochegurova E.A., Wu D.* Realization of a recursive digital filter based on penalized splines. Computer Optics 2018. V. 42. № 6. P. 1083–1092.
31. *Bhattacharjee S.S., Patel V., George N.V.* Nonlinear Spline Adaptive Filters based on a Low Rank Approximation. Signal Processing. 2022. V. 201. P. 108726.
32. *Agrawal N., Kumar A., Bajaj V., Singh G.K.* Design of digital IIR filter: A research survey. Applied Acoustics 2021. V. 172. P. 107669. https://doi.org/10.1016/j.apacoust.2020.107669
33. *Rathod M., Patel V., George N.V.* Generalized spline nonlinear adaptive filters. Expert Systems with Applications. 2017. V. 83. № 15. P. 122–130.
34. *Воскобойников Ю.Е., Колкер А.Б.* Аппроксимация изолиний изображений сглаживающими сплайнами. Автометрия. 2003. Т. 39. № 4. С. 3–12.

# ADAPTIVE IIR FILTER BASED ON PENALIZED SPLINE

**© 2024 E. A. Kochegurova[a], I. A. Martynova[a]**

[a] *National Research Tomsk Polytechnic University*
*Lenina Prospekt 30, Tomsk, 634050 Russia*

The purpose of this research is to develop the technique of spline adaptive filters (SAF) for real-time implementation. The P-SAF proposed in the article based on the recurrent penalty P-spline, by analogy with the classical SAF, consists of linear dynamic and nonlinear static components. To adapt P-SAF, computing circuits with different topologies have been developed. This approach specifies a way to adapt the knots and calculate the spline coefficients simultaneously. This made it possible to increase the efficiency of P-SAF compared to the classical SAF and reduce computational costs. The efficiency indicator *MSE* [*dB*] for P-SAF is equal to and higher than for classical SAF when analyzing model and real time series.

*Keywords*: P-spline, spline adaptive filter, instrumental function

*ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 35–47*

## АНАЛИЗ ДАННЫХ

*УДК 004.023*

# ДИСКРЕТНЫЙ АЛГОРИТМ ОПТИМИЗАЦИИ НА ОСНОВЕ РАСПРЕДЕЛЕНИЯ ВЕРОЯТНОСТЕЙ С ТРАНСФОРМАЦИЕЙ ЦЕЛЕВЫХ ЗНАЧЕНИЙ

**© 2024 г. К. С. Сарин[a, *]**

[a] *Томский государственный университет систем управления и радиоэлектроники*
*634510 Томск, ул. Ленина, д. 40, Россия*

** E-mail: sarin.konstantin@mail.ru*

Поступила в редакцию 18.03.2024 г.
После доработки 06.06.2024 г.
Принята к публикации 17.06.2024 г.

Оптимизационные задачи поиска в дискретном пространстве и, в частности, бинарном, где переменная может принимать только два значения, имеют большое прикладное значение. В статье предлагается новый популяционный алгоритм дискретной оптимизации, основанный на распределениях вероятностей переменных. Распределения определяют вероятность выбора дискретных значений переменных при поиске и формируются с помощью трансформации целевых значений решений в их весовые коэффициенты. Работоспособность алгоритма оценивалась на унимодальных и мультимодальных тестовых функциях с бинарными переменными. Результаты эксперимента показали высокую эффективность предлагаемого алгоритма на оценках сходимости и стабильности.

*Ключевые слова*: методы оптимизации, дискретная оптимизация, бинарная оптимизация, метаэвристики, стохастические алгоритмы, распределение вероятностей

**DOI:** 10.31857/S0132347424060049, **EDN:** DYRZOO

## 1. ВВЕДЕНИЕ

Решение задач оптимизации является необходимостью практически во всех сферах жизнедеятельности человека. Настоящая работа сосредоточена на комбинаторной оптимизации, которая занимается проблемами нахождения оптимума с дискретными значениями возможных решений. Одним из частных случаев этой проблемы является бинарная оптимизация, в которой элементы вектора решения могут принимать только два значения. Практическое применение таких задач весьма обширно. В области медицины бинарная оптимизация применялась для диагностики опухолей головного мозга [1], нахождения подмножеств согласованных признаков при прогнозировании эффективности реабилитации пациентов после коронавирусной инфекции [2], классификации сложных заболеваний [3], классификации аритмии по электрокардиограмме [4]. В экономической сфере для выбора издателей журналов при размещении рекламы [5], планировании рабочего процесса [6], планировании выпуска новой версии программного обеспечения [7], проектировании производственных ячеек [8]. В науке и технике бинарная оптимизация использовалась для нахождения подмножества информативных признаков при построении прогностических систем [9–11], восстановлении нагрузки в первичных распределительных сетях [12], диагностики неисправности энергосистем [13], решении проблемы позиционирования антенны [14], проектировании сварных балок [15], разделение аппаратного и программного обеспечения во встроенных системах [16]. Так же в [17] отмечается, что помимо чисто комбинаторных задач, задачи с вещественными числами могут быть представлены в двоичном виде и решены в дискретном числовом пространстве.

Для решения задач бинарной оптимизации применяют два типа методов. Первый тип – это традиционные детерминированные методы оптимизации, а второй тип основан на стохастических, недетерминированных алгоритмах. Традиционными являются методы релаксации, Лагранжа, ветвей и границ, целочисленное программирование [18, 19]. Эти методы являются трудозатратными и предназначены для решения задач небольших размерностей, что на практике встречается очень редко. Кроме того, большинство традиционных методов требуют

аналитическое задание целевой функции, а также ее дифференцируемость и непрерывность. Задачи большой размерности с множеством локальных оптимумов значительно ухудшают поиск традиционных методов.

Недетерминированные методы, представляемые метаэвристическими алгоритмами [20–23], устраняют вышеперечисленные проблемы. В отличие от традиционных методов данные алгоритмы не подвергнуты "застреванию" в локальных оптимумах, в меньшей степени зависят от исходных отправных точек, не ограничены видом целевой функции и способны решать оптимизационную проблему "черного ящика" [24].

В [25] доказано, что не существует эвристического алгоритма, который мог бы работать достаточно эффективно для решения всех задач оптимизации. Разработанные в настоящее время алгоритмы дают удовлетворительные результаты при решении некоторых задач, но не всех. Поэтому в этой области ведутся активные исследования, в результате чего предлагаются новые эвристические алгоритмы.

Цель настоящей работы заключается в разработке эффективного алгоритма дискретной оптимизации, конкурирующего с популярными алгоритмами в бинарном пространстве поиска.

Основной научный вклад работы представлен следующими пунктами.

1. Разработан новый популяционный метаэвристический алгоритм оптимизации для поиска в дискретном пространстве. Алгоритм использует распределения вероятностей для выбора значений переменных. Распределения формируются с помощью трансформации целевых значений решений в весовые коэффициенты.

2. Эмпирически доказана эффективность предложенного алгоритма для поиска в бинарном пространстве с помощью критериев сходимости и стабильности. Статистический тест Уилкоксона показал значимое преимущество предлагаемого алгоритма по сравнению с генетическим алгоритмом и алгоритмом роящихся частиц для оптимизации унимодальных и мультимодальных тестовых функций.

Остальная часть статьи организована следующим образом. В п. 2 рассмотрены подходы и методы решения задач бинарной оптимизации с помощью метаэвристических алгоритмов; в п. 3 представлен новый алгоритм и детали его работы; в п. 4 описана экспериментальная часть исследования; в п. 5 обсуждены полученные результаты; в заключении сделаны выводы о проделанной работе.

## 2. БЛИЗКИЕ РАБОТЫ ПО ТЕМЕ ИССЛЕДОВАНИЯ

Наиболее популярные алгоритмы бинарной оптимизации относятся к алгоритмам роевого интеллекта. Подобно эволюционным они основаны на механизмах природы и представляют собой модель скоординированного поведения объектов, которые могут быть представителями флоры, фауны или физическими объектами. Эволюционные вычисления основаны на конкуренции и естественном отборе, тогда как роевой интеллект опирается главным образом на сотрудничество агентов [26].

Большинство алгоритмов роевого интеллекта разработаны для непрерывной оптимизации и для того чтобы осуществлять поиск в бинарном пространстве применяются механизмы адаптации, называемые бинаризацией [27]. Самым популярным методом бинаризации является использование трансформационных функций, которые переводят непрерывные значения элементов векторов решений в значения из диапазона [0, 1]. Затем применяется правило бинаризации, при котором решение преобразуется в бинарное значение из множества {0, 1}. С помощью функций трансформации были адаптированы алгоритмы роящихся частиц [17, 28], искусственных водорослей [29], шимпанзе [30], роя сальпов [31], стаи китов [32]. В [28] были исследованы различные варианты функций трансформации для алгоритма роящихся частиц. Лучшая сходимость была достигнута алгоритмом с V-образной функцией трансформации.

Метод бинаризации на основе модификации алгебраических операций преобразует вещественные операторы, используемых в формулах перемещения частиц, в их логические аналоги, что позволяет оперировать бинарными решениями. Например, вместо сложения используется операция дизъюнкции, а вместо умножения – конъюнкция. С помощью данного метода были адаптированы алгоритмы роящихся частиц [33], мозгового штурма [13], роста деревьев [34], летучих мышей [33], непрерывной муравьиной колонии [5], кукушкин поиск [35], черной дыры [36].

Квантовый метод бинаризации тоже преобразует операторы непрерывного алгоритма. В этом методе каждое допустимое решение имеет позицию и вектор квантования, который содержит вероятности принять значение 1 для соответствующего элемента решения. Вектор квантования обновляются с учетом положений глобальных и локальных лидеров. Используя данный метод, были адаптированы алгоритмы

роящихся частиц [37], искусственных водорослей [7], гравитационный поиск [38], муравьиной колонии [39].

Среди алгоритмов эволюционного интеллекта для бинарной оптимизации широко применялся генетический алгоритм [40–42]. Также были предложены его модификации, так, например, в работе [43] представлен гибрид на основе генетического алгоритма и алгоритма роящихся частиц. Сначала оба алгоритма находят решения независимо друг от друга, а затем результаты объединяются с помощью метода средневзвешенной комбинации. После этого применяется локальный поиск для нахождения окончательного решения.

Оценка эффективности алгоритмов в большинстве исследований проводилась при решении определенных прикладных задач. Для объективной оценки работы алгоритмов применяют тестовые функции, которые позволяют определить эффективность при нахождении оптимума различных целевых функций, например, унимодальных, мультимодальных, овражных, разрывных, выпуклых, вогнутых. При бинарной оптимизации применяют тестовые функции для поиска в непрерывном пространстве. Бинарное пространство поиска образуют путем дискретизации непрерывного и последующим бинарным кодировании дискретных значений [28, 44–46].

## 3. НОВЫЙ ДИСКРЕТНЫЙ АЛГОРИТМ ОПТИМИЗАЦИИ

В настоящей работе рассматривается проблема оптимизации, в которой минимизируется критерий эффективности. В данном разделе представлен оригинальный дискретный метаэвристический алгоритм оптимизации на основе распределения вероятностей с трансформацией целевых значений (Probability Distributions with Transformation of target values, PDT). Алгоритм является итерационным, где на каждой итерации формируется вероятностная модель. Модель определяет вероятность появления конкретного дискретного значения переменной. Вероятности формируются на основе частоты появления дискретного значения каждой переменной среди решений популяции, причем меньшее значение целевой функции должно увеличивать вклад решения в повышение вероятности. Для этого вводятся трансформационные функции, которые переводят значение целевой функции решения популяции в весовой коэффициент. На рис. 1 представлена блок-схема алгоритма.


**Рис. 1.** Блок-схема алгоритма.

На этапе инициализации определяется начальная популяция решений *Pop* случайным или иным образом. Далее рассчитываются весовые коэффициенты решений популяции *w*. Весовой коэффициент принимает значение из диапазона [0, 1], чем меньше целевое значение решения, тем больше значение *w*. Для того чтобы сформировать весовые коэффициенты из целевых значений предлагается использовать функции трансформации. В качестве таких функций, например, могут быть использованы следующие: $T_L$ – линейная функция, $T_S$ – сигмоида, $T_Q$ – квадратичная функция, $T_{Th}$ – гиперболический тангенс. Графики функций представлены на рис. 2. Область определения функций ограничена отрезком $[f_{\min}, f_{\max}]$, где $f_{\min}$

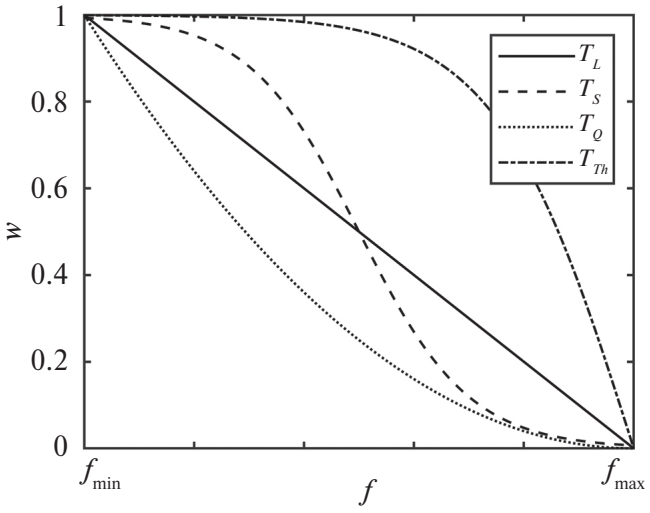



**Рис. 2.** Графики функций трансформаций для перевода целевых значений в веса решений.

и $f_{\max}$ – минимальное и максимальное значение целевой функции в популяции соответственно, а $f$ – текущее значение. Аналитические выражения функций трансформации показаны ниже:

$$T_L(f) = \frac{f_{\max} - f}{f_{\max} - f_{\min}},$$

$$T_s(f) = \frac{1}{1 + e^{\left(10\left(f - f_{\min}\right)/\left(f_{\max} - f_{\min}\right) - 5\right)}},$$

$$T_Q(f) = \left(\frac{f - f_{\min}}{f_{\max} - f_{\min}}\right)^2,$$

$$T_{Th}(f) = \left|\text{th}\left(4\left(\frac{f - f_{\min}}{f_{\max} - f_{\min}} - 1\right)\right)\right|.$$

После получения весовых коэффициентов решений рассчитываются распределения вероятностей. Каждое распределение состоит из вероятностей получить переменной определенное дискретное значение. Вероятности рассчитываются на основе значений переменных и весовых коэффициентов решений.

С помощью полученных распределений генерируется новая популяция и подвергается мутации, чтобы предотвратить преждевременную сходимость. Далее популяция обновляется лучшими решениями текущей и новой популяции. После этого снова рассчитываются весовые коэффициенты решений, и продолжается новый цикл. По завершении заданного количества итераций из популяции выбирается лучшее решение.

Ниже представлено пошаговое описание алгоритма.

*Вход*: Установить размер популяции $N$, число итераций *MaxIter*, вероятность мутации $p_a$ и функцию трансформации $T$. Обозначим $l_{jk}$ $k$-е дискретное значение $j$-й переменной.

*Выход*: $R$ – найденное решение.

*Начало*

*Шаг 1.* Инициализация.

Случайным или иным образом сгенерировать популяцию решений $Pop = [Pop_1, Pop_2, ..., Pop_N]$ и вычислить соответствующее целевое значение $f = [f_1, f_2, ..., f_N]$.

*Шаг 2.* Инициализировать счетчик итераций $t = 1$. Начало итерационного процесса.

*Шаг 3.* Вычислить весовые коэффициенты решений популяции *Pop* с помощью функции трансформации $T$:

$$w_i = T\left(f_i\right),$$

где $i = 1, ..., N$.

*Шаг 4.* Вычислить распределения вероятностей.

Для каждого дискретного значения $k$ переменной $j$ определить сумму весовых коэффициентов решений *Pop*, которые принимают данное дискретное значение $k$. Обозначим такую сумму $S_{jk}$, где $j = 1, ..., n$, $k = 1, ..., m$:

$$S_{jk} = \sum_{i=1}^{N} w_i \cdot \begin{cases} 1, \text{ если } Pop_{ij} = l_{jk} \\ 0, \text{ иначе} \end{cases}.$$

Вычислить эмпирическую вероятность появления $k$-го значения переменной $j$:

$$P_{jk} = \frac{S_{jk}}{\sum_{k=1}^{m} S_{jk}}.$$

*Шаг 5.* Сгенерировать новые решения.

Формируется популяция новых решений $Pop^{\text{new}}$ на основе вероятностей $P$ каждой переменной:

$$Pop_{ij}^{\text{new}} = l_{jk},$$

где $k$ удовлетворяет условию $p_{k-1} < rand\,(0,1) \leqslant p_k$,

$$p = \left[0, P_{j1}, P_{j1} + P_{j2}, \ldots, \sum\nolimits_{k=1}^{m} P_{jm}\right],$$

$j = 1, ..., n$, $k = 1, ..., m$.

*Шаг 6.* Изменить новые решения (Мутация).

Изменить значения элементов векторов решений $Pop^{\text{new}}$ с вероятностью $p_a$. Новые значения выбираются случайным образом из области значений элемента вектора решений.

*Шаг 7.* Вычислить значение целевых функций $f_i^{\text{new}}$, где $i = 1, ..., N$, для каждого решения $Pop^{\text{new}}$.

*Шаг 8.* Обновить популяцию *Pop* путем выбора *N* лучших решений из множества решений $Pop \cup Pop^{new}$. Обновить значения целевых функций $f_i$ согласно решениям *Pop*.

*Шаг 9.* Проверка остановки алгоритма.

Если $t < MaxIter$, то $t = t + 1$ и перейти на *Шаг 3*, иначе перейти на *Шаг 10*.

*Шаг 10.* Извлечь в *R* лучшее решение из популяции *Pop*.

*Конец*

## 4. ЭКСПЕРИМЕНТ

В настоящем разделе представлены эксперименты с разработанным дискретным алгоритмом оптимизации на основе распределения вероятностей с трансформацией целевых значений. Алгоритм тестировался для бинарной проблемы оптимизации, т. е. когда переменные принимают только два значения. В экспериментальном исследовании использовались восемнадцать различных унимодальных и мультимодальных эталонных функции, широко применяемых для тестирования алгоритмов оптимизации [28, 44–46]. В табл. 1 представлены их характеристики, а на рис. 3 графики в двумерном пространстве поиска. Функции $f_1$–$f_{11}$ являются унимодальными, т. е. содержат только один глобальный оптимум. Функции $f_{12}$–$f_{18}$ являются мультимодальными и содержат один глобальный и множество локальных оптимумов, число которых экспоненциально растет с увеличением размерности задачи. Эксперимент проводился согласно методике работы [44].

Реализация алгоритма осуществлялась на языке MATLAB в среде программирования MATLAB R2022b. Программа доступна по ссылке https://cloud.tusur.ru/index.php/s/395znYyx87rRoDP. Эксперимент проводился на персональном компьютере под управлением операционной системы Windows 10 с 8 Гб оперативной памяти и процессором Intel Core i7-12700.

### *4.1. Дискретизация непрерывных значений*

Поскольку алгоритм является дискретным и оперирует в эксперименте бинарными векторами решений, элементы которых принимают значение 0 или 1, проводится кодировка вещественных значений бинарным вектором. Процедура перевода бинарного вектора решения в значения вещественных переменных представлена на рис. 4. Данная процедура выполняется всякий раз, когда алгоритму необходимо рассчитать значение целевой функции. Количество переменных в эксперименте 5, количество битов для кодирования значения каждой переменной – 15. Таким образом, величина бинарного вектора решений составляет $n = 5 \times 15 = 75$ элементов. Количество дискретных значений, которое может иметь каждая переменная, соответствует $2^{15}$. Эти значения определяются с помощью равномерного квантования на диапазоне поиска переменной. Шаг дискретизации определяется следующим образом:

$$\Delta h = \frac{R_{\max} - R_{\min}}{2^{15} - 1},$$

где $R_{\min}$ и $R_{\max}$ – нижняя и верхняя граница диапазона значений переменной соответственно. Фактически, бинарное значение переменной – это бинарное представление порядкового номера дискретного значения на диапазоне $[R_{\min}, R_{\max}]$ с шагом дискретизации $\Delta h$.

Если переменная $x$ кодируется бинарным вектором $[b_1, ..., b_{15}]$, то вещественное значение этой переменной определяется следующим образом:

$$x = R_{\min} + \Delta h \sum_{i=1}^{15} b_i \cdot 2^{i-1}.$$

### *4.2. Критерии эффективности*

Для оценки эффективности работы алгоритма применялись два критерия [47]. Первый оценивает сходимость алгоритма и определяется средним отклонением найденного целевого значения от фактического:

$$E = \frac{1}{n_{run}} \sum_{i=1}^{n_{run}} \left| f_i - f' \right|,$$

где $n_{run}$ – количество запусков алгоритма; $f_i$ – найденное алгоритмом значение целевой функции в $i$-м запуске; $f'$ – фактическое значения оптимума целевой функции. Второй критерий оценивает стабильность работы недетерминированного алгоритма и определяется среднеквадратичным отклонением найденного оптимума целевой функции:

$$STD = \sqrt{\frac{1}{n_{run}} \sum_{i=1}^{n_{run}} \left( f_i - M \right)^2},$$

где $M$ – среднее значение целевой функции;

$$M = \frac{\sum_{i=1}^{n_{run}} f_i}{n_{run}}.$$

Меньшее значение обоих критериев соответствует лучшему значению эффективности.

Кроме вышеприведенных критериев в работе представлены графики сходимости алгоритмов, позволяющие оценить скорость сходимости стохастических алгоритмов и показывающие

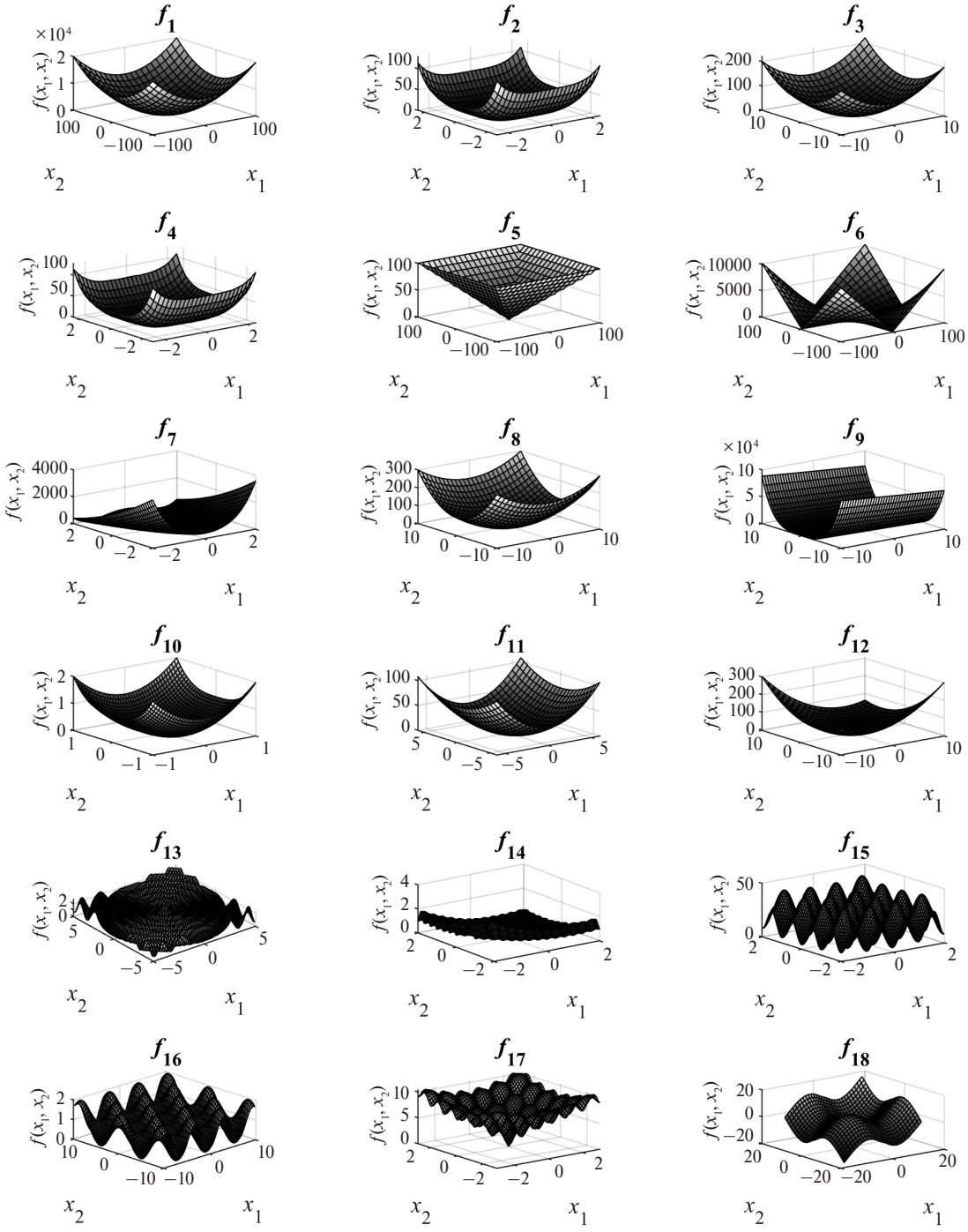



**Рис. 3.** Графики тестовых функций в двумерном пространстве поиска.

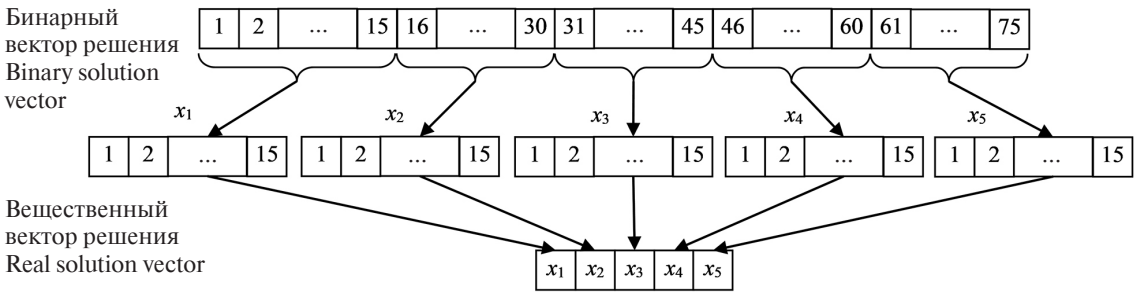



**Рис. 4.** Перевод бинарного вектора решения в непрерывный вектор для вычисления значения целевой функции.

**Таблица 1.** Тестовые функции эксперимента

| № | Целевая функция | Диапазон поиска | Значение оптимума функции |
|---|---|---|---|
| 1 | $f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$ | [−100; 100] | 0 |
| 2 | $f_2(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4$ | [−2.56; 2.56] | 0 |
| 3 | $f_3(\mathbf{x}) = \sum_{i=1}^{n} \left(\lvert x_i + 0.5\rvert\right)^2$ | [−10; 10] | 0 |
| 4 | $f_4(\mathbf{x}) = \sum_{i=1}^{n} i x_i^4 + rand(0, 1)$ | [−2.56; 2.56] | 0 |
| 5 | $f_5(\mathbf{x}) = \max_{i=1}^{n}(x_i)$ | [−100; 100] | 0 |
| 6 | $f_6(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$ | [−100; 100] | 0 |
| 7 | $f_7(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right)$ | [−2; 2] | 0 |
| 8 | $f_8(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$ | [−10; 10] | 0 |
| 9 | $f_9(\mathbf{x}) = \left(x_1 - 1\right)^2 + \sum_{i=2}^{n} i\left(2x_i^2 - x_{i-1}\right)^2$ | [−10; 10] | 0 |
| 10 | $f_{10}(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i\rvert^{i+1}$ | [−1; 1] | 0 |
| 11 | $f_{11}(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$ | [−6; 6] | 0 |
| 12 | $f_{12}(\mathbf{x}) = \sum_{i=1}^{n} \left(x_i - 1\right)^2 - \sum_{i=2}^{n} x_i x_{i-1}$ | [−10; 10] | −30 (при $n = 5$) |
| 13 | $f_{13}(\mathbf{x}) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$ | [−25; 25] | 0 |
| 14 | $f_{14}(\mathbf{x}) = 0.1\left(\sin^2\left(3\pi x_1\right) + \sum_{i=1}^{n-1} \left(x_i - 1\right)^2 \left(1 + \sin^2\left(3\pi x_{i+1}\right)\right) + \ldots\right.$ $\left.\ldots\left(x_n - 1\right)\left(1 + \sin^2\left(2\pi x_n\right)\right)\right) + \sum_{i=1}^{n} u\left(x_i, 5, 100, 4\right)$ $u\left(x_i, a, k, m\right) = \begin{cases} k\left(x_i - a\right)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k\left(-x_i - a\right)^m, & x_i < -a \end{cases}$ | [−5; 5] | 0 |

**Таблица 1.** Окончание

| № | Целевая функция | Диапазон поиска | Значение оптимума функции |
|---|---|---|---|
| 15 | $f_{15}(\mathbf{x}) = \sum_{i=1}^{n}\left(x_i - 10\cos\left(2\pi x_i\right) + 10\right)$ | [−2; 2] | 0 |
| 16 | $f_{16}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ | [−10; 10] | 0 |
| 17 | $f_{17}(\mathbf{x}) = -20e^{\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)} - e^{\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)} + 20 + e$ | [−3; 3] | 0 |
| 18 | $f_{18}(\mathbf{x}) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$ | [−15; 15] | −50.0929 (при $n = 5$) |

зависимость критерия $E$ от итерации [28, 44, 45]. Значение $E$, приводимое на графиках, является средним значением по запускам алгоритма.

### *4.3. Выбор функции трансформации целевых значений*

Для выбора функции трансформации целевых значений в весовые коэффициенты решений были использованы следующее функции: $T_L$ – линейная функция, $T_S$ – сигмоида, $T_Q$ – квадратичная функция, $T_S$ – гиперболический тангенс.

Алгоритм PDT с разными функциями трансформации был использован для поиска оптимума тестовых функций. Было осуществлено 30 запусков на каждой тестовой функции. Полученные значения критериев эффективности приведены в табл. 2.

Для улучшения оценки эффективности эволюционных алгоритмов в [47] отмечается, что необходимо проводить статистические тесты. Недостаточно сравнивать алгоритмы по значениям $E$ и $STD$ [48], необходимо провести статистический тест, чтобы доказать, что предлагаемый новый

**Таблица 2.** Оценка эффективности алгоритма PDT с различными функциями трансформации

| $f$ | $T_L$ | | $T_S$ | | $T_Q$ | | $T_{Th}$ | |
|---|---|---|---|---|---|---|---|---|
| | $E$ | $STD$ | $E$ | $STD$ | $E$ | $STD$ | $E$ | $STD$ |
| $f_1$ | 0.000047 | 0.000000 | 0.000049 | 0.000014 | 0.000047 | 0.000000 | 0.000059 | 0.000034 |
| $f_2$ | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| $f_3$ | 0.402924 | 0.231313 | 0.454002 | 0.280296 | 0.478528 | 0.240684 | 0.498614 | 0.281871 |
| $f_4$ | 0.005653 | 0.004306 | 0.006070 | 0.004650 | 0.005624 | 0.003659 | 0.005170 | 0.003590 |
| $f_5$ | 0.041098 | 0.096504 | 0.030112 | 0.026176 | 0.042929 | 0.140588 | 0.037639 | 0.042328 |
| $f_6$ | 0.015259 | 0.000000 | 0.015259 | 0.000000 | 0.015666 | 0.002229 | 0.015463 | 0.001114 |
| $f_7$ | 2.629031 | 1.500038 | 2.511857 | 1.363161 | 3.214144 | 1.576402 | 3.036793 | 1.573228 |
| $f_8$ | 0.000001 | 0.000000 | 0.000001 | 0.000000 | 0.000001 | 0.000000 | 0.000003 | 0.000003 |
| $f_9$ | 0.885491 | 0.863594 | 0.850277 | 0.472212 | 0.803704 | 0.485206 | 0.835841 | 0.739384 |
| $f_{10}$ | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| $f_{11}$ | 0.000001 | 0.000000 | 0.000001 | 0.000000 | 0.000001 | 0.000000 | 0.000001 | 0.000002 |
| $f_{12}$ | 0.239046 | 0.613565 | 0.870265 | 1.432694 | 0.885326 | 2.910895 | 0.362761 | 0.898002 |
| $f_{13}$ | 0.119874 | 0.040684 | 0.149874 | 0.062972 | 0.163207 | 0.071839 | 0.123821 | 0.056141 |
| $f_{14}$ | 0.057948 | 0.077372 | 0.058916 | 0.059387 | 0.053028 | 0.053763 | 0.033397 | 0.046236 |
| $f_{15}$ | 0.331668 | 0.603435 | 0.199166 | 0.404835 | 0.398790 | 0.618058 | 0.099557 | 0.303747 |
| $f_{16}$ | 0.036349 | 0.016301 | 0.028747 | 0.014743 | 0.029202 | 0.015521 | 0.034756 | 0.019396 |
| $f_{17}$ | 0.000374 | 0.000041 | 0.000374 | 0.000041 | 0.000367 | 0.000000 | 0.000389 | 0.000069 |
| $f_{18}$ | 0.407336 | 1.540216 | 0.405125 | 1.540966 | 0.610862 | 1.858916 | 0.000837 | 0.003885 |

алгоритм представляет собой значительное улучшение по сравнению с другими существующими методами.

Чтобы судить о том отличаются ли статистически значимо результаты работы алгоритма с различными функциями трансформации друг от друга, был проведен непараметрический статистический тест Фридмана на уровне значимости $\alpha = 0.05$. Значения асимптотической значимости *p-value*, которые меньше 0.05, можно рассматривать как убедительное свидетельство против нулевой гипотезы $H_0$ [47].

Тест Фридмана множественных сравнений не выявил отклонение гипотезы $H_0$ для обоих критериев. Гипотеза $H_0$ здесь утверждение об отсутствии значимых различий между вариантами алгоритма с различными функциями трансформации. Асимптотическая значимость для критерия *E* соответствует значению *p-value* = 0.757, а для критерия *STD* значению *p-value* = 0.590. Таким образом, выбор рассмотренных функций трансформации существенно не повлияет на эффективность работы алгоритма. В дальнейшем будет использоваться линейная функция.

### *4.4. Параметры эксперимента*

Эффективность предлагаемого алгоритма PDT оценивалась в сравнении с такими популярными алгоритмами оптимизации как генетический алгоритм (GA) и бинарный алгоритм роящихся частиц (BPSO). Алгоритмы выполнялись в одинаковых условиях. Общие настройки имели следующие значения. Размер популяции – 30, количество итераций – 100, количество переменных – 5, число бит на одну переменную – 15, количество запусков алгоритма на каждую тестовую функцию – 30. Специфичные параметры алгоритмов GA и BPSO были установлены в значения, рекомендованные в [28, 45]. Значения специфичных параметров приведены в табл. 3.

**Таблица 3.** Значения параметров алгоритмов

| Алгоритм | Параметр | Значение |
|---|---|---|
| PDT | Функция трансформации $T$ | Линейная $T_L$ |
| | Вероятность мутации $p_a$ | 0.05 |
| GA | Вид селекции | Рулеточная |
| | Вид скрещивания (вероятность) | Одноточечный (0.9) |
| | Вид мутации (вероятность) | Равномерный (0.005) |
| BPSO | Коэффициенты $C_1$, $C_2$ | 2, 2 |
| | Вес инерции $W$ | Линейно уменьшается с 0.9 до 0.4 |
| | Максимальная скорость | 6 |
| | Функция трансформации | V-образная |

### *4.5. Результаты эксперимента*

В результате выполнения эксперимента были получены значения критериев эффективности каждого алгоритма. Данные значения приведены в табл. 4. Последняя строка таблицы содержит средние значения показателей. На рис. 5 показаны

**Таблица 4.** Оценки эффективности алгоритмов

| $f$ | *GA* | | *BPSO* | | *PDT* | |
|---|---|---|---|---|---|---|
| | *E* | *STD* | *E* | *STD* | *E* | *STD* |
| $f_1$ | 0.005349 | 0.028272 | 27.845964 | 37.522025 | 0.000047 | 0.000000 |
| $f_2$ | 0.000000 | 0.000000 | 0.008293 | 0.018114 | 0.000000 | 0.000000 |
| $f_3$ | 0.440548 | 0.254233 | 0.799580 | 0.513039 | 0.402924 | 0.231313 |
| $f_4$ | 0.021143 | 0.046160 | 0.067826 | 0.054019 | 0.005653 | 0.004306 |
| $f_5$ | 0.649841 | 1.354239 | 5.634938 | 3.111477 | 0.041098 | 0.096504 |
| $f_6$ | 0.016683 | 0.004988 | 4.112874 | 3.333243 | 0.015259 | 0.000000 |
| $f_7$ | 3.113280 | 2.491653 | 4.288055 | 2.188550 | 2.629031 | 1.500038 |
| $f_8$ | 0.000009 | 0.000025 | 0.804053 | 0.875332 | 0.000001 | 0.000000 |
| $f_9$ | 10.522179 | 24.073438 | 1.827357 | 1.320775 | 0.885491 | 0.863594 |
| $f_{10}$ | 0.000000 | 0.000000 | 0.000166 | 0.000383 | 0.000000 | 0.000000 |
| $f_{11}$ | 0.000004 | 0.000008 | 0.303480 | 0.574945 | 0.000001 | 0.000000 |
| $f_{12}$ | 0.760543 | 1.068967 | 0.571626 | 0.780258 | 0.239046 | 0.613565 |
| $f_{13}$ | 0.457784 | 0.259707 | 0.418610 | 0.159854 | 0.119874 | 0.040684 |
| $f_{14}$ | 0.072918 | 0.062301 | 0.054331 | 0.053543 | 0.057948 | 0.077372 |

**Таблица 4.** Окончание

| $f$ | *GA* | | *BPSO* | | *PDT* | |
|---|---|---|---|---|---|---|
| | *E* | *STD* | *E* | *STD* | *E* | *STD* |
| $f_{15}$ | 1.227600 | 1.001766 | 1.408564 | 0.734589 | 0.331668 | 0.603435 |
| $f_{16}$ | 0.042970 | 0.016959 | 0.061572 | 0.021475 | 0.036349 | 0.016301 |
| $f_{17}$ | 0.000583 | 0.000525 | 0.357644 | 0.145604 | 0.000374 | 0.000041 |
| $f_{18}$ | 4.935197 | 6.147513 | 3.989981 | 3.238601 | 0.407336 | 1.540216 |
| | **1.24** | **2.05** | **2.92** | **3.04** | **0.29** | **0.31** |

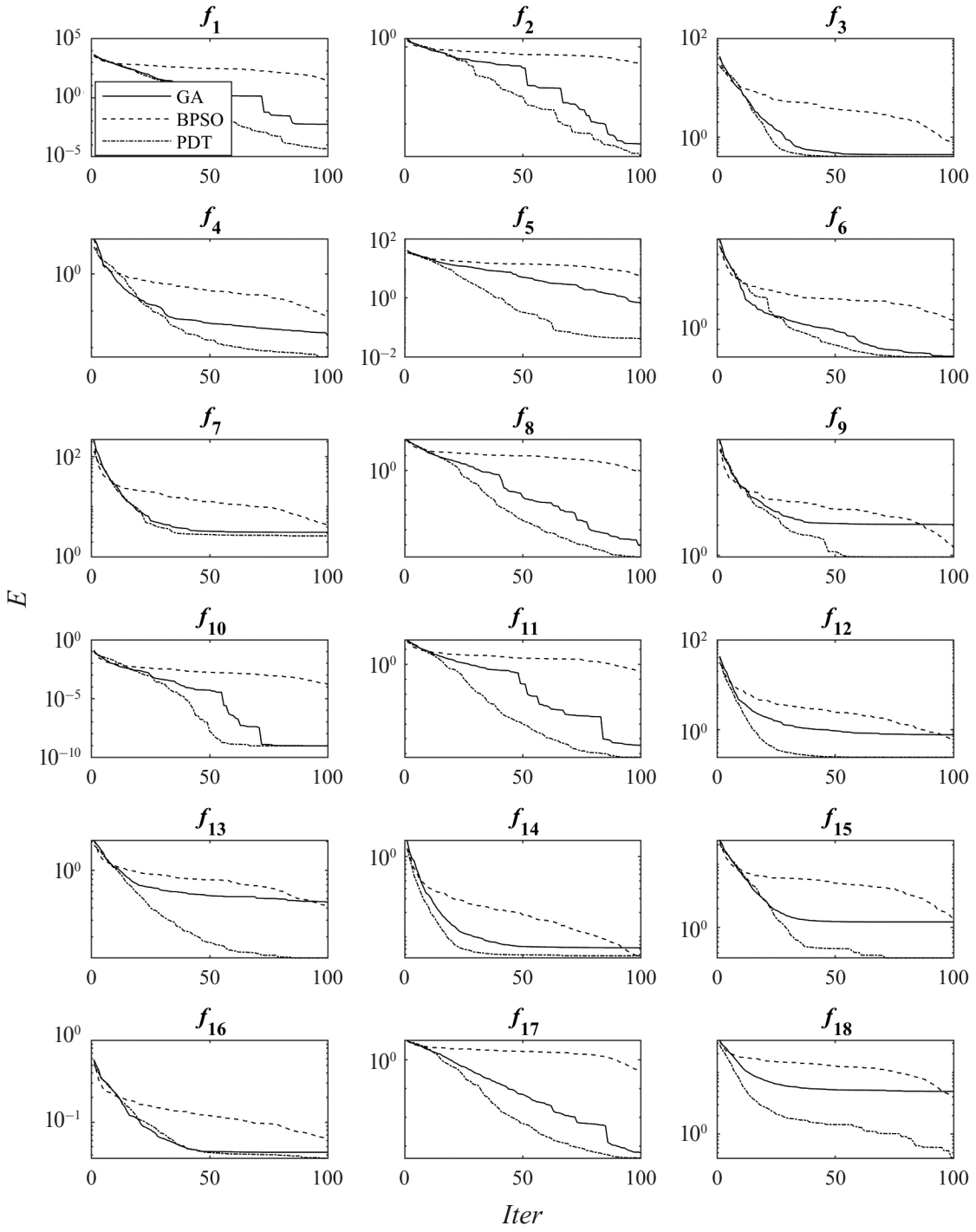



**Рис. 5.** Графики сходимости алгоритмов.

**Таблица 5.** Статистическое сравнение алгоритмов

| | *PDT-GA* | | *PDT-BPSO* | |
|---|---|---|---|---|
| | *E* | *STD* | *E* | *STD* |
| *P-value* | <0.01 | <0.01 | <0.01 | <0.01 |
| Гипотеза $H_0$ | Отвергается | Отвергается | Отвергается | Отвергается |

графики сходимости алгоритмов, позволяющие оценить скорость сходимости. Графики представлены в логарифмической шкале по оси критерия сходимости, что позволяет более четко отследить скорость сходимости алгоритмов на протяжении всей их работы.

## 5. ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

Чтобы определить, значительно ли отличаются результаты эффективности предлагаемого алгоритма от аналогов, воспользуемся парным статистическим тестом Уилкоксона [47]. Нулевая гипотеза $H_0$ теста утверждает отсутствие значимых различий в оценках эффективности сравниваемых алгоритмов. В табл. 5 представлены результаты сравнения. Асимптотическая значимость для критериев *E* и *STD* при сравнении с генетическим алгоритмом и алгоритмом роящихся частиц соответствует значению *p-value* < 0.01. Сумма отрицательных рангов теста превалирует над положительными. Это говорит о том, что значения критериев алгоритма PDT статистически значимо меньше алгоритмов GA и BPSO на уровне значимости $\alpha = 0.01$.

Анализ рис. 5 показывает, что на начальных итерациях скорость алгоритма роящихся частиц для тестов $f_6, f_7, f_9, f_{15}$ и $f_{16}$ оказывается выше остальных алгоритмов, но начиная, примерно, с пятнадцатой итерации она спадает. В целом же алгоритм на основе распределения вероятностей с трансформацией целевых значений опережает по скорости своих конкурентов.

## 6. ЗАКЛЮЧЕНИЕ

Предложенный дискретный алгоритм на основе распределения вероятностей с трансформацией целевых значений показал статистически значимое улучшение показателей сходимости и стабильности, таких как отклонение от оптимума и среднеквадратичное отклонение целевых значений. Сравнения проводились с генетическим алгоритмом и бинарным алгоритмом роящихся частиц. Для эксперимента использовались восемнадцать тестовых унимодальных и мультимодальных функций. В среднем отклонение от оптимума уменьшилось в 4.3 раза по сравнению с генетическим алгоритмом и в 10.1 раза по сравнению с бинарным алгоритмом роящихся частиц. Полученные результаты говорят об эффективности предложенного алгоритма для оптимизации в бинарном пространстве.

## 7. БЛАГОДАРНОСТИ

Автор выражает глубокую признательность Илье Александровичу Ходашинскому, профессору Томского государственного университета систем управления и радиоэлектроники, за обсуждение настоящего исследования и ряда ценных советов и замечаний.

## ИСТОЧНИК ФИНАНСИРОВАНИЯ

Исследование выполнено за счет гранта Российского научного фонда № 24-21-00168, https://rscf.ru/project/24-21-00168/.

## СПИСОК ЛИТЕРАТУРЫ

1. *Aly R.H.M., Rahouma K.H., Hamed H.F.* Brain Tumors Diagnosis and Prediction Based on Applying the Learning Metaheuristic Optimization Techniques of Particle Swarm, Ant Colony and Bee Colony. Procedia Computer Science. 2019. V. 163. P. 165–179.
2. *Ходашинский И.А., Смирнова И.Н., Бардамова М.Б., Сарин К.С., Светлаков М.О., Зайцев А.А., Тицкая Е.В., Тонкошкурова А.В., Антипова И.И., Ходашинская А.И., Зарипова Т.Н.* Метод нахождения подмножеств согласованных признаков при прогнозировании эффективности реабилитации пациентов после перенесенной коронавирусной инфекции // Сибирский журнал клинической и экспериментальной медицины. Т. 38. № 4. С. 270–279.
3. *Phogat M. Kumar D.* Classification of complex diseases using an improved binary cuckoo search and conditional mutual information maximization. Computacion y Sistemas. 2020. V. 24. P. 1121–1129.
4. *Houssein E.H., Ibrahim I.E., Neggaz N., Hassaballah M., Wazery Y.M.* An efficient ECG arrhythmia classification method based on Manta ray foraging optimization. Expert Systems with Applications. 2021. V. 181. P. 115131.
5. *Aytimur A., Babayigit B.* Binary Artificial Bee Colony Algorithms for {0–1} Advertisement Problem. Proceedings of the 2019 6th International Conference on

Electrical and Electronics Engineering (ICEEE), Istanbul, Turkey, 16–17 April 2019. P. 91–95.

6. *Mohammadzadeh A., Masdari M., Gharehchopogh F.S., Jafarian A.* Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing. Evolutionary Intelligence. 2021. V. 14. P. 1997–2025.
7. *Pirozmand P., Ebrahimnejad A., Alrezaamiri H., Motameni H.* A novel approach for the next software release using a binary artificial algae algorithm. Journal of Intelligent and Fuzzy Systems. 2021. V. 40. P. 5027–5041.
8. *Almonacid B., Aspee F., Soto, R., Crawford B., Lama J.* Solving the manufacturing cell design problem using the modified binary firefly algorithm and the egyptian vulture optimisation algorithm. IET Software. 2017. V. 11. P. 105–115.
9. *Ходашинский И.А., Сарин К.С.* Отбор классифицирующих признаков с помощью популяционного случайного поиска с памятью. Автоматика и телемеханика. 2019. № 2. С. 161–172.
10. *Ходашинский И.А., Сарин К.С.* Отбор классифицирующих признаков: сравнительный анализ бинарных метаэвристик и популяционного алгоритма с адаптивной памятью // Программирование. 2019. № 5. С. 3–9.
11. *Sarin K., Hodashinsky I., Slezkin A.* Feature selection and identification of fuzzy classifiers based on the cuckoo search algorithm. Communications in Computer and Information Science. 2018. V. 934. P. 22–34.
12. *El-Dakroury H.E.D.M., Gad A., Abdelaziz A.Y.* Load Restoration in Primary Distribution Networks Using the Binary Particle Swarm Optimization. Proceedings of the 2019 IEEE Electrical Power and Energy Conference (EPEC), Ottawa, ON, Canada, 12–14 October 2016. P. 1–6.
13. *Xiong G., Shi D., Zhang J., Zhang Y.* A binary coded brain storm optimization for fault section diagnosis of power systems. Electric Power Systems Research. 2018. V. 163. P. 441–451.
14. *Dahi Z.A.E.M., Mezioud C., Draa A.* A 0–1 bat algorithm for cellular network optimisation: A systematic study on mapping techniques. International Journal of Reasoning-based Intelligent Systems. 2017. V. 9. P. 22–42.
15. *Hussien A.G., Hassanien A.E., Houssein E.H., Amin M., Azar A.T.* New binary whale optimization algorithm for discrete optimization problems. Engineering Optimization. 2020. V. 52. P. 945–959.
16. *Mourad K., Boudour R.* A modified binary firefly algorithm to solve hardware/software partitioning problem. Informatica. 2021. V. 45. P. 1–12.
17. *Kennedy J., Eberhart R.C.* A discrete binary version of the particle swarm algorithm. Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997. 1997. V. 5. P. 4104–4108.
18. *Serigne G., Philippe M.* A linearization framework for unconstrained quadratic (0–1) problems. Discrete Applied Mathematics. 2009. V. 157. P. 1255–1266.
19. *Sherali H.D., Driscoll P.J.* Evolution, and state-of-the-art in integer programming. Journal of Computational and Applied Mathematics. 2000. V. 124. P. 319–340.
20. *Ходашинский И.А.* Методы повышения эффективности роевых алгоритмов оптимизации // Автоматика и телемеханика. 2021. № 6. С. 3–45.
21. *Gendreau M., Potvin J.-Yv. (Eds.)* Handbook of methaheuristics. Springer, 2019, 604 p.
22. *Карпенко А.П.* Методы повышения эффективности метаэвристических алгоритмов глобальной оптимизации // Математические методы в технике и технологиях – ММТТ. 2017. Т. 1. С. 77–83.
23. *Курейчик В.В., Родзин С.И.* Биоэвристики, инспирированные фауной (обзор) // Информационные технологии. 2023. Т. 29. № 11. С. 559–573.
24. *Pardalos P.M., Rasskazova V., Vrahatis M.N. (Eds.)* Black box optimization, machine learning, and no-free lunch theorems. Springer, 2021. 388 p.
25. *Wolpert D.H., Macready W.G.* No free lunch theorems for optimization. IEEE. Transactions on Evolutionary Computation. 1997. V. 1. P. 67–82.
26. *Becerra-Rozas M., Lemus-Romani J., Cisternas-Caneo F., Crawford B., Soto R., Astorga G., Castro C., Garcia J.* Continuous Metaheuristics for Binary Optimization Problems: An Updated Systematic Literature Review. Mathematics. 2022. V. 11. № 1. P. 129.
27. *Бардамова М.Б., Буймов А.Г., Тарасенко В.Ф.* Способы адаптации алгоритма прыгающих лягушек к бинарному пространству поиска при решении задачи отбора признаков // Доклады Томского государственного университета систем управления и радиоэлектроники. 2020. Т. 23. № 4. С. 57–62.
28. *Mirjalili S., Lewis A.* S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm and Evolutionary Computation. 2013. V. 9. P. 1–14.
29. *Turkoglu B., Uymaz S.A., Kaya E.* Binary Artificial Algae Algorithm for feature selection. Applied Soft Computing. 2022. V. 120. P. 108630.
30. *Pashaei E., Pashaei E.* An efficient binary chimp optimization algorithm for feature selection in biomedical data. Neural Computing and Applications. 2022. V. 34. P. 6427–6451.
31. *Jain S., Dharavath R.* Memetic salp swarm optimization algorithm based feature selection approach for crop disease detection. Journal of Ambient Intelligence and Humanized Computing. 2023. V. 14. – P. 1817–1835.
32. *Mohd Yusof N., Muda A.K., Pratama S.F., Abraham A.* A novel nonlinear time-varying sigmoid transfer function in binary whale optimization algorithm for descriptors selection in drug classification. Molecular Diversity. 2023. V. 27. № 1. P. 71–80.

33. *Merikhi B., Soleymani M.* Automatic data clustering framework using nature-inspired binary optimization algorithms. IEEE Access. 2021. V. 9. P. 93703–93722.
34. *Zhong C., Chen Y., Peng J.* Feature selection based on a novel improved tree growth algorithm. International Journal of Computational Intelligence Systems. 2020. V. 13. P. 247–258.
35. *Pandey A.C., Rajpoot D.S., Saraswat M.* Feature selection method based on hybrid data transformation and binary binomial cuckoo search. Journal of Ambient Intelligence and Humanized Computing. 2020. V. 11. P. 719–738.
36. *Yepes V., Marti J.V., Garcia J.* Black hole algorithm for sustainable design of counterfort retaining walls. Sustainability. 2020. V. 12. P. 2767.
37. *Lai X., Hao J.K., Fu Z.H., Yue D.* Diversity-preserving quantum particle swarm optimization for the multidimensional knapsack problem. Expert Systems with Applications. 2020. V. 149. P. 113310.
38. *Barani F., Mirhosseini M., Nezamabadi-Pour H.* Application of binary quantum-inspired gravitational search algorithm in feature subset selection. Applied Intelligence. 2017. V. 47. P. 304–318.
39. *Ross O.H.M.* A review of quantum-inspired metaheuristics: Going from classical computers to real quantum computers. IEEE Access. 2019. V. 8. P. 814–838.
40. *Shreem S.S., Turabieh H., Al Azwari S., Baothman F.* Enhanced binary genetic algorithm as a feature selection to predict student performance. Soft Computing. 2022. V. 26. P. 1811–1823.
41. *Nicolau M.* Application of a simple binary genetic algorithm to a noiseless testbed benchmark. Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. 2009. P. 2473–2478.
42. *Haupt R.L., Haupt S.E.* Practical genetic algorithms. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004. 253 p.
43. *Ghosh M., Guha R., Alam I., Lohariwal P., Jalan D., Sarkar R.* Binary Genetic Swarm Optimization: A Combination of GA and PSO for Feature Selection. Journal of Intelligent Systems. 2019. V. 29. P. 1598–1610.
44. *Bas E., Ulker E.* A binary social spider algorithm for continuous optimization task. Soft Computing. 2020. V. 24. P. 12953–12979.
45. *Mirjalili S., Mirjalili S.M., Yang X.-S.* Binary bat algorithm. Neural Computing and Applications. 2014. V. 25. P. 663–681.
46. *Pan J.-S., Hu P., Chu S.-C.* Binary fish migration optimization for solving unit commitment. Energy. 2021. V. 226. P. 120329.
47. *Derrac J., Garcia S., Molina D., Herrera F.* A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation. 2011. V. 1. P. 3–18.
48. *Garcia S., Molina D., Lozano M., Herrera F.* A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. Journal of Heuristics. 2009. V. 15. № 6. P. 617–644.

# DISCRETE OPTIMIZATION ALGORITHM BASED ON PROBABILITY DISTRIBUTION WITH TRANSFORMATION OF TARGET VALUES

**© 2024 K. S. Sarin[a]**

[a] *Tomsk State University of Control Systems and Radioelectronics*
*Prospect Lenina 40, Tomsk, 634050 Russia*

Optimization problems of searching in discrete space and, in particular, binary space, where a variable can take only two values, are of great practical importance. This paper proposes a new population discrete optimization algorithm based on probability distributions of variables. Distributions determine the probability of accepting one or another discrete value and are formed by transforming the target values of decisions into their weighting coefficients. The performance of the algorithm was assessed using unimodal and multimodal test functions with binary variables. The experimental results showed the high efficiency of the proposed algorithm in terms of convergence and stability estimates.

*Keywords*: optimization methods, discrete optimization, binary optimization, metaheuristics, stochastic algorithms, probability distribution

*ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 48–58*

## АНАЛИЗ ДАННЫХ

*УДК: 004.93*

# ОПТИМИЗАЦИЯ БЫСТРОДЕЙСТВИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РЕАЛИЗАЦИИ АЛГОРИТМОВ КЛАССИФИКАЦИИ И ПРИВЯЗКИ ДЕЛОВЫХ ДОКУМЕНТОВ

**© 2024 г. О. А. Славин**[a, b, *]

[a] *Федеральный исследовательский центр "Информатика и управление" РАН*
*119333 Москва, ул. Вавилова, 44/2, Россия*
[b] *ООО "Смарт Энджинс Сервис"*
*117312 Москва, проспект 60-летия Октября, 9, Россия*
** E-mail: oslavin@isa.ru*

Поступила в редакцию 13.07.2024 г.
После доработки 15.07.2024 г.
Принята к публикации 15.07.2024 г.

В работе рассматриваются технологии оптимизации быстродействия программного обеспечения. Методы оптимизации подразделяются на высокоуровневые и низкоуровневые, а также на распараллеливание. Описываемые методы оптимизации применяются к программам и программным системам, реализующим разнообразную обработку информации, в которых неэффективность использования аппаратных ресурсов может присутствовать в большом числе горячих точек. Как пример приведен алгоритм классификации и привязки полей в распознанном образе делового документа. Перечисляются особенности реализации задач классификации и привязки, состоящие в применении созвездий особых текстовых точек и модифицированного расстояния Левенштейна, В качестве OCR была использована система SDK Smart Document Engine и Tesseract. Описано несколько способов оптимизации быстродействия функций классификации и привязки содержимого документа. Также описана оптимизация быстродействия системы сортировки потока изображений деловых документов. Предлагаемые методы оптимизации быстродействия программного обеспечения пригодны не только для реализации алгоритмов обработки изображений, но и для вычислительных алгоритмов, в которых проводится циклическая обработка информации большого объема.

*Ключевые слова*: анализ текста, распознавание документа, классификация документа, текстовая особая точка, ускорение

**DOI:** 10.31857/S0132347424060057, **EDN:** DYKMMM

## 1. ВВЕДЕНИЕ

Необходимость оптимизации быстродействия программ объясняется различными потребностями. Требования к быстродействию может сформулировать Заказчик информационной системы в техническом задании. Требования к быстродействию указываются для конкретных видов компьютеров и конкретных операционных систем. Аналогично возникают требования к быстродействию для приложений, предназначенных для мобильных устройств. Ограничения по времени выполнения мобильных приложений связаны не только с ограничением времени реакции, но и с ограничениями энергопотребления и нагрева мобильного устройства. Существует корреляция между быстродействием приложения и энергопотреблением.

Оптимизация ПО преследует одну или несколько целей:

- уменьшение среднего времени исполнения программного приложения на некотором тестовом наборе;
- уменьшение среднего времени исполнения реализованной функции.

## 2. ПРИНЦИПЫ ОПТИМИЗАЦИИ БЫСТРОДЕЙСТВИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При разработке программного обеспечения (далее – ПО) возможна оптимизация нескольких типов:

- высокоуровневая оптимизация (оптимизация алгоритмов);
- низкоуровневая оптимизация с использованием особенностей вычислительной техники;
- параллельное программирование.

Высокоуровневая оптимизация базируется на выборе метода решения задачи, которая должна войти в состав разрабатываемой программы. Алгоритм может существенно зависеть от области определения, например, уменьшение объема данных очевидным образом уменьшает время перебора. Выбор параметров алгоритма также может существенно влиять на сложность. Высокоуровневая оптимизация проводится для упрощенной архитектуры компьютера, такой как архитектура фон Неймана или Гарвардская архитектура. Под архитектурой понимается совокупность пользовательских характеристик, к которым относят основные устройства и блоки упрощенного компьютера, а также структуру связей между ними. С точки зрения разработчика программного обеспечения архитектура компьютера – набор описаний используемых данных, операций (инструкций) и их характеристик.

В высокоуровневой оптимизации возможно применение следующих способов:

• анализ алгоритма (реализация, выбор, область определения, параметризация, правило остановки);

• использование промежуточных данных (мемоизация [1]);

• представление исходных данных;

• уменьшение сложности алгоритма (lookup table [2], интерполяция);

• оптимизация циклов (вынос вычислений из тела цикла, слияние циклов, loop unrolling).

Низкоуровневая оптимизация, ориентированная на аппаратную платформу, – это совокупность технических средств, определяющих среду функционирования конкретных программ. Основой аппаратной платформы является совокупность системной (материнской) платы, центрального процессора (далее – ЦП) и запоминающих устройств. Выполняемая на компьютере программа состоит из команд конкретного процессора.

Низкоуровневая оптимизация проводится для конкретных процессорных микроархитектур [3], таких как:

• CISC (различная длина машинной инструкции);

• RISC (одинаковая длина машинной инструкции);

• VLIW (параллельное выполнение нескольких операций в одной инструкции);

• суперскалярная архитектура, в которой решение о параллельном исполнении двух или более команд между несколькими устройствами исполнения принимается аппаратурой процессора на этапе исполнения.

Вообще говоря, низкоуровневая оптимизация для различных процессорных микроархитектур будет различной. Например, различаются множества команд SIMD для ЦП ARM и ЦП Intel. Директивы программиста для улучшения параллельного исполнения инструкций для ЦП архитектуры VLIW невозможны для ЦП Intel.

Использование параллельного исполнения (параллелизации) является эффективным способом повышения быстродействия ПО. Параллельному исполнению способствует такая организация вычислительного процесса, при которой на одном процессоре попеременно выполняются сразу несколько программ, совместно использующих один или несколько процессоров и другие ресурсы компьютера. Такой способ называется многозадачностью. Многозадачность призвана повысить эффективность использования вычислительной системы, при этом могут использоваться различные критерии эффективности вычислительных систем, например:

• пропускная способность – количество задач, выполняемых вычислительной системой в единицу времени в операционных системах (далее – ОС) пакетной обработки;

• удобство работы пользователей, заключающееся, в частности, в том, что они имеют возможность интерактивно работать одновременно с несколькими приложениями на одной машине в ОС разделения времени;

• реактивность системы – способность системы выдерживать заранее заданные (возможно, очень короткие) интервалы времени между запуском программы и получением результата в ОС реального времени.

Параллелизацию имеет смысл проводить после завершения высокоуровневой и низкоуровневой оптимизации последовательной программы или алгоритма. Основой параллелизации является выполнение частей программы на нескольких исполнительных устройствах, в качестве которых мы будем рассматривать несколько центральных процессоров. Формально различают параллельное выполнение задач приложения на одном компьютере и распределенное выполнение нескольких приложений на нескольких компьютерах в локальной сети. Основными подходами к разработке параллельных программ являются:

• последовательное программирование с дальнейшим автоматическим распараллеливанием;

• непосредственное формирование потоков параллельного управления с учетом особенностей архитектур параллельных вычислительных систем или операционных систем;

• описание параллелизма без использования явного управления.

## 3. МЕТОДЫ ОПТИМИЗАЦИИ БЫСТРОДЕЙСТВИЯ ПО

Описываемые далее методы оптимизации применяются, прежде всего, к программам и программным системам, реализующим разнообразную обработку информации. То есть мы рассматриваем не программы, в которых реализуется один или несколько математических алгоритмов, таких как, например, функции из библиотек Eigen или MKL. Иначе класс рассматриваемых программ можно охарактеризовать как программы, в которых неэффективность использования аппаратных ресурсов может присутствовать в большом числе горячих точек.

### *3.1. Особенности высокоуровневой оптимизации быстродействия для реализации проектов и разработки продуктов*

Описанные в разделе 2 принципы оптимизации ПО применяются в различных условиях по-разному.

Так, формирование требований к быстродействию существенно различается при разработке продуктов (API, библиотек, приложений) и реализации проектов (в том числе систем, подсистем, функциональных модулей). Оба вида разработки имеются сходство, для них возможны:

• использование готовых программных модулей;

• высокоуровневая оптимизация;

• применение экспертных оценок ускорения.

Однако имеются и существенные различия. Например, план выпуска продукта допускает увеличение времени на разработку, связанную с созданием новых алгоритмов, которые априори должны обладать меньшей сложностью, нежели уже реализованные алгоритмы. При разработке продуктов необходимо предусмотреть возможность постоянного профилирования и другого анализа быстродействия. В разработке продуктов необходим анализ быстродействия конкурирующих продуктов, на которые следует ориентироваться. В реализации проектов требования к быстродействию могут быть сформированы заказчиком проекта, но срок реализации и ресурсы проекта могут ограничить время, затрачиваемое на оптимизацию. В обоих случаях предусматривается оптимизация покупных и собственных модулей. Оптимизация покупных модулей в части замены или модификации имплементированных алгоритмов чаще всего невозможна, но возможно управление быстродействием посредством представления данных и выбора параметров вызываемых методов.

### *3.2. Профилирование ПО*

Во всех случаях необходимо *профилирование*, т. е. измерение быстродействия пользовательской программы или частей программы. Целью профилирования является исследование быстродействия приложения как в целом, так и составляющих его объектах (точках): функциях, циклах, строках, инструкциях.

Замеры позволяют анализировать:

• общее время исполнения приложения;

• набор горячих точек (hot spots);

• удельное время каждой точки;

• количество вызовов точки;

• степень покрытия программы (доля кода, которая была использована при исполнении);

• загрузку CPU и захват памяти;

• низкоуровневую статистику о загрузке CPU и шины доступа к памяти, кэш-промахах и кэш-попадания;

• степень параллелизма приложения.

Профилированию может подвергаться как отладочная версия приложения, так и оптимизированная отладочная версия или релизная версия с отладочной информацией. Существенным условием профилирования является проведение замеров на одном и том же тестовом наборе (датасете), который может быть как опубликованным в открытом доступе, так и собственным. Профилирование необходимо проводить в одних и тех же условиях на выбранных платформах, включающих аппаратную часть и системное ПО.

Известны несколько методов инструментального профилирования:

• ручной (проведение замеров времени с помощью вручную добавленных в код приложения вызовов системных функций, таких как std:: clock, или использования методов библиотеки std:: chrono);

• семплирование (time-based profiling) – сбор статистики о работе приложения во время профилирования;

• инструментирование (instrumentation) – сбор детализированной информации о времени работы каждой вызванной функции во время профилирования.

Достоинством ручного подхода является простота замеров небольшого числа заранее известных точек, недостатки состоят в следующем:

• необходимость вставки дополнительного кода в программу, что может приводить

к появлению не только ошибок, но и наведенных эффектов;

• возможность анализа результата в текстовом виде;

• отсутствие дерева вызовов функций, средств автоматического анализа.

В профайлере, основанном на семплировании, периодически собирается информация о состоянии программы, например, значения счетчиков производительности процессора, значение счетчика команд. На основании этих замеров проводится подсчет производительности. Метод семплирования менее всего влияет на работу анализируемого ПО и полученная информация обладает малой погрешностью. Основными недостатками таких профайлеров являются:

• получение неполной информации о коде;

• длительное время для сбора реальной статистики.

Метод инструментирования модифицирует исполняемый файл ПО для сбора информации о выполнении каждой функции, исключая время, потраченное на вызываемые функции и обращения к операционной системе. Метод инструментирования предоставляет больше информации, чем метод семплирования, но может существенно замедлить работу профилируемого ПО.

Известно несколько универсальных профайлеров, к ним относятся следующие программы:

• CodeAnalyst;

• Valgrind;

• Performance Profiler из среды Visual Studio;

• Intel VTune.

Все перечисленные профайлеры позволяют определять горячие точки программы, локализовать участки кода, в которых неэффективно используются аппаратные ресурсы, неэффективность использования процессора, выявлять объекты синхронизации, которые негативно влияют на производительность программы. Продукт Intel VTune предоставляет возможность оценки большого числа аппаратных счетчиков и метрик для определения критических объектов. Intel VTune позволяет в режиме эмуляции моделировать работу определенного процессора, включая кэш-память различного уровня с механизмами замещения, декодеры и буферы инструкций, конвейеры инструкций и другие компоненты ЦП и его взаимодействия с памятью.

### *3.3. Методы низкоуровневой оптимизации быстродействия ПО*

Основным способом низкоуровневой оптимизации при разработке ПО является выбор параметров компилятора. Например, для среды Microsoft Visual Studio и других компиляторов важнейшим параметром является вид оптимизации: без оптимизации или с оптимизацией быстродействия, или с оптимизацией объема программы. Другим способом оптимизации быстродействия является явное указание архитектуры, позволяющее компилятору применять при трансляции инструкции выбранного набора команд.

Использование математических библиотек также является эффективным средством ускорения вычислений. Компилятор Intel Compiler Enable Matrix содержит параметр Multiply Library Call, который включает или отключает вызов библиотеки Matrix Multiply. Широко известна библиотека Intel MKL, содержащая многочисленные реализованные и оптимизированные для современных ЦП вычислительные методы.

Разумеется, остается востребованным и программирование на ассемблере или применение интринсиков, что, собственно, и позволяет создавать библиотеки для быстрых вычислений.

### *3.4. Подходы к распараллеливанию программ*

Основными подходами к разработке параллельных программ являются:

• последовательное программирование с дальнейшим автоматическим распараллеливанием;

• непосредственное формирование потоков параллельного управления с учетом особенностей архитектур параллельных вычислительных систем или операционных систем;

• описание параллелизма без использования явного управления.

Последовательное программирование с дальнейшим автоматическим распараллеливанием упрощает разработку, позволяя полностью абстрагироваться от возможностей параллелизации и возложить распараллеливание на инструментальные средства. Очевидный недостаток этого подхода – невозможность достичь максимального ускорения.

Перечислим некоторые средства автоматического распараллеливания с помощью кроссплатформенных многопоточных библиотек для языка C++:

• Qt4 Threads;

• Intel Threading Building Blocks (TBB).

Достоинством подхода, состоящего в непосредственном формировании потоков параллельного управления, является возможность получения значительно большего ускорения для конкретной вычислительной системы по

сравнению с предыдущим подходом. Недостатки подхода состоят в следующем:

• усложнение разработки из-за ручного управления собственными процессами и потоками, в том числе, анализ возможных конфликтов;

• зависимость от конкретной архитектуры многопроцессорного комплекса, что затрудняет переносимость на другие платформы.

Упрощение разработки возможно при распараллеливании с указанием директив, которые указывают компилятору на необходимость распараллеливании фрагмента исходного кода с предполагаемой возможной параллельной реализацией.

При распараллеливании также необходимо проводить профилирование для оценки потерь времени из-за конфликтов потоков и неоптимальности использования данных несколькими потоками.

## 4. ЗАДАЧИ РАСПОЗНАВАНИЯ И КЛАССИФИКАЦИИ ДЕЛОВЫХ ДОКУМЕНТОВ

Распознавание изображений документов с известным описанием включает в себя несколько задач, таких как:

• поиск границ документа;

• нормализация размера и границ документа;

• извлечение графических примитивов;

• распознавание символов и слов, анализ структуры документа;

• поиск границ и распознавание полей документа;

• постобработка результатов распознавания.

Важнейшими задачами являются классификация типа документа (фрагмента документа) и привязки полей (поиск областей документа для извлечения заполнения). При анализе распознанных изображений необходимо учитывать ошибки OCR, появляющиеся в зашумленных, осветленных или искаженных образах документов. В данной работе рассматриваются деловые документы, предназначенные для обмена данными с организациями и физическими личностями [4]. Деловые документы характеризуются относительно простой структурой и ограниченным словарем статических текстов.

Мы будем определять документ как совокупность полей и статической информации. Структура текста делового документа может быть описана с помощью трех объектов: слово, строка текста и фрагмент текста. Для классификации распознанного документа и привязки полей могут быть применены текстовые особые точки и созвездия текстовых точек, определенные в [5, 6]. Текстовые особые точки являются аналогами геометрических особых точек [7, 8]. Слово модели представляется последовательностью символов. Распознанное слово представляется матрицей альтернатив соответствия знакомест символов символам алфавита распознавания и рамкой слова.

Для пары текстовых особых точек ($\omega_1$, $\omega_2$) могут быть заданы следующие отношения:

• $\omega_1 \in S$, $\omega_2 \in S$ ($\omega_1 \in F$, $\omega_2 \in F$) – обе текстовые точки принадлежат одной строке S или одному фрагменту текста F;

• $\omega_1 \in S_1$, $\omega_2 \in S_1$ ($\omega_1 \in F_1$, $\omega_2 \in F_2$) – обе точки $\omega_1$ и $\omega_2$ принадлежат различным строкам $S_1$ и $S_2$ или различным фрагментам текста $F_1$ и $F_2$;

• $\omega_1 < \omega_2$ – точка $\omega_1$ размещена “перед” точкой $\omega_2$;

• $\omega_1 \vee \omega_2$ – точка $\omega_1$ размещена “выше” точки $\omega_2$.

Строка текста является созвездием нескольких близких друг к другу текстовых особых точек. Строки текста могут быть найдены с помощью алгоритмов кластеризации рамок распознанных слов. Строка описывается множеством упорядоченных текстовых особых точек. Для двух строк ($S_1$, $S_2$) могут быть заданы следующие отношения:

• $S_1 \in F$, $S_2 \in F$ – обе строки текста принадлежат одному фрагменту текста F;

• $S_1 \in F_1$, $S_2 \in F_2$ – обе строки текста принадлежат различным фрагментам текста $F_1$ и $F_2$;

• $S_1 \vee S_2$ – строка $S_1$ размещена “выше” строки $S_2$.

Под привязкой строки мы понимаем установление соответствия слов распознанной строки с одной из описанных возможных строк. Некоторые точки являются обязательными для привязки. При привязке с каждой обязательной точкой должно быть отождествлено некоторое распознанное слово. Также в описании строки могут присутствовать запрещенные текстовые точки. При привязке строки ни одна из запрещенных точек не может быть отождествлена с некоторым распознанным словом. Остальные текстовые точки могут быть отождествлены с распознанными словами при привязке строки, или не быть отождествленными.

В описании строки для пары текстовых точек могут быть заданы ограничения с помощью следующих метрик:

• количество точек в промежутке между двумя точками $\omega_1$ и $\omega_2$;

• сумма ширин текстовых точек, размещенных между точками $\omega_1$ и $\omega_2$;

• количество строк в промежутке между строками, содержащими точки $\omega_1$ и $\omega_2$;

• евклидово расстояние между двумя проекциями рамок точек $\omega_1$ и $\omega_2$.

Фрагмент текста является совокупностью нескольких текстовых строк. В нашей модели предполагается, что в одном фрагменте строки группируются только в одну колонку. Для двух фрагментов ($F_1$, $F_2$) могут быть заданы следующие отношения:

• $F_1 \in F$, $F_2 \in F$ – два фрагмента принадлежат текстовому фрагменту F;

• $F_1 < F_2$ – фрагмент $F_1$ размещен "перед" фрагментом $F_2$;

• $F_1 \vee F_2$ – фрагмент $F_1$ размещен "выше" фрагмента $F_2$.

Фрагменты текста могут быть созданы с помощью алгоритмов анализа структуры текста. Разбиение документа на части осуществляется на основе его графического строения (разделяющих прямых, колонок, абзацев и т. п.) под управлением некоторого описания (шаблона) документа [9, 10]. Для разбиения на фрагменты могут быть использованы как отрезки, разделяющие фрагменты, так и промежутки между фрагментами. Под привязкой фрагмента мы понимаем установление соответствия слов и строк фрагмента с одним из описанных возможных фрагментов. Аналогично описаниям строки в описании фрагмента содержатся обязательные, запрещенные и обычные строки и слова.

Созвездие задается в виде последовательности упорядоченных точек $\omega_1, \ldots, \omega_n$. Простым случаем созвездия является последовательность точек, принадлежащих одной текстовой строке, в самом простом случае это – шингл (последовательность слов заголовка документа). Другим случаем созвездия являются цепи – последовательность точек, пары которых упорядочены отношениями $\omega_1 < \omega_2$ (простая цепь) или $\omega_1 \vee \omega_2$ (вертикальная цепь). Использование цепей и созвездий позволяет не только находить тип документа, но и детектировать фрагменты и строки текста. Последнее позволяет сократить объем текста, используемого в анализе текстового объекта, например, в привязке поля.

В работах [5, 6] описан способ привязки полей гибкого документа. Привязка строк, параграфов и фрагментов документа проводится с помощью следующего алгоритма классификации. Задаются модели допустимых строк $M_1, M_2, \ldots, M_q$, каждая модель M определена набором текстовых особых точек:

$$M = \begin{Bmatrix} W^+_1, W^+_2, \ldots, W^+_{k+}, W_1, W_2, \ldots, \\ W_k, W^-_1, W^-_2, \ldots, W^-_{k-} \end{Bmatrix} \quad (1)$$

и параметр $d_{\mathrm{LINK}}(M)$ – пороговое значение числа привязанных точек для надежной привязки. В наборе (1) используются три мешка слов:

• запрещенные слова $W^- = \{W^-_1, W^-_2, \ldots, W_{k-}\}$;

• обязательные слова $W^+ = \{W^+_1, W^+_2, \ldots, W^+_{k+}\}$;

• необязательные слова $W = \{W_1, W_2, \ldots, W_k\}$.

Вычисляются оценки $\Delta(S, M_i)$ соответствия моделям каждой из строк S. Оценка $\Delta(S, M_i)$ равняется 0, если:

• была привязана хотя бы одна точка из множества $W^-(S)$;

• не было привязано ни одной точки из множества $W^+(S)$.

Оценка $\Delta(S, M_i)$ равняется 1, если не было привязано ни одной точки $W^-(S)$ и число привязанных точек W(S) и $W^+(S)$ превосходит $d_{\mathrm{LINK}}(M_i)$. Если $\Delta(S, M_i)$ равняется 1, то строка S считается привязанной к модели $M_i$. Точность привязки строк зависит от предварительной привязки окрестности допустимого размещения строк. После привязки строк проводится поиск (прогноз) границ полей для последующего извлечения информации. Для области каждого поля задаются опорные элементы, определяющие прямоугольник или многоугольник. Привязка поля проводится с помощью привязанных опорных элементов.

Описанный алгоритм классификации строк применяется для классификации документа. Классификация образа страницы документа проводится с помощью привязки точек созвездия с учетом заданных отношений между некоторыми точками.

Отождествление текстовой точки и распознанного слова проводится с помощью предложенного в [6] модифицированного расстояния Левенштейна (далее – МРЛ). Механизм отождествления слов применяется во многих задачах, основанных на сравнении слов с алфавитом в базе данных [11, 12]. Оригинальное расстояние Левенштейна [13] между двумя текстовыми строками V и W определяется как минимальное число редакционных операций для трансформации V в W и вычисляется следующим образом:

$$d_{\mathrm{LEV}}(V, W) = D_{\mathrm{LEV}}(|V|, |W|),$$

$$\forall_j D_{\mathrm{LEV}}(0, j) = 0, \forall_i D_{\mathrm{LEV}}(i, 0) = 0, \quad (2)$$

$$D_{\mathrm{LEV}}(i, j) = \min\big(D_{\mathrm{LEV}}(i, j-1) + 1, D_{\mathrm{LEV}}(i-1, j) + 1, D_{\mathrm{LEV}}(i-1, j-1) + \mathrm{substCost}(v_i, w_i)\big),$$

где $\mathrm{substCost}(v_i, w_j)$ – цена операции замены символа $v_i$ на символ $w_j$, |V| и |W| – длины слов V

и W. По умолчанию цена любой из редакционных операций равняется 1. В работе алгоритм вычисления расстояния Левенштейна между двумя текстовыми строками реализован в полном соответствии с рекуррентной формулой (2). В реализации не применялись методы экономии памяти, уменьшающие производительность.

Мы будем считать тождественными слова V и W, если $d_{\mathrm{LEV}}(\mathrm{V}, \mathrm{W}) < d(\mathrm{V})$, где $d(\mathrm{V})$ – известный порог для слова модели. При распознавании программами OCR появляются неединичные ошибки распознавания. Поэтому порог $d(\mathrm{V})$ не может быть нулевым. Очевидно, что порог $d(\mathrm{V})$ должен быть различным для слов различной длины. Для учета этого обстоятельства можно использовать нормализованное расстояние Левенштейна [14]:

$$\rho_{\mathrm{LEV}}\left(\mathrm{V}, \mathrm{W}\right) = \frac{2d_{\mathrm{LEV}}\left(\mathrm{V}, \mathrm{W}\right)}{\left|\mathrm{V}\right| + \left|\mathrm{W}\right| + d_{\mathrm{LEV}}\left(\mathrm{V}, \mathrm{W}\right)}.$$

При распознавании зашумленных и искаженных изображений документов возможно появление многочисленных ошибок распознавания. Некоторые ошибки OCR не являются случайными. Ошибочное распознавание образа символа “X” как символа “O” маловероятно. В то же время образ символа “Ъ” может быть ошибочно распознан как символ “Ь” из-за сходства образов “Ъ” и “Ь”. Примерами сходных образов для латинского алфавита являются пары символов “B8”, “DO”, “1I”. Другими словами, некоторые ошибки распознавания символов случаются чаще, чем другие. Для учета этого нужно построить $\mathrm{substCost}(v_i, w_j)$ так, чтобы при вычислении расстояния Левенштейна штраф за сходные символы был меньше, чем за символы несходные:

- для одинаковых символов $\mathrm{substCost}(v_i, v_i) = 0$;
- для различных несходных символов $\mathrm{substCost}(v_i, w_j) = 1$;
- для сходных же символов $\mathrm{substCost}(v_i, w_j) = 0$, либо $0 < \mathrm{substCost}(v_i, w_j) < 1$.

Описанная модификация позволяет уменьшить расстояние, вычисляемое для слов с ошибками в виде сходных символов.

Для некоторых далеких по смыслу слов, например, идентификаторов, расстояние Левенштейна между ними является небольшим. Для исключения рассмотренных случаев ложного отождествления предлагается применять шаблоны слов модели следующего вида:

$$G\left(V\right) = b_1 b_2 \ldots b_k \cdot m_1 m_2 \ldots m_p \cdot e_1 e_2 \ldots e_q.$$

В этих шаблонах заданы обязательные символы в начале, в середине или в конце слова. Если при сравнении символы распознанного слова не удовлетворяют шаблону, то расстояние Левенштейна увеличивается на заданный заранее штраф. Эта модификация позволяет увеличить расстояние между словами, различающимися незначительным числом символов, которые являются признаками для различия слов.

Штраф при отождествлении может быть назначен за несоответствие длин слов V и W:

$$G_2\left(V,\ W\right) = \left\|V\right| - \left|W\right\| > \delta\left(V\right).$$

Сходство между словом модели V и распознанным словом W устанавливают по формуле

$$\mathrm{Sim}\left(\mathrm{V}, \mathrm{W}\right) = d_{\mathrm{LEV}}\left(\mathrm{V}, \mathrm{W}\right) - f_1\left(\mathrm{G}\left(\mathrm{V}\right), \mathrm{W}\right) - f_2\left(\mathrm{G}_2\left(\mathrm{V}, \mathrm{W}\right)\right),$$

где $f_1(\mathrm{G}(\mathrm{V}), \mathrm{W})$ – штраф за несоответствие слова модели V и распознанного слова W, вычисленный с помощью шаблона $\mathrm{G}(\mathrm{V})$; $f_2(\mathrm{G}_2(\mathrm{V}, \mathrm{W}))$ – штраф за несоответствие длин слова модели V и распознанного слова W.

При этом может применяться функция $\mathrm{substCost}(v_i, w_j)$, учитывающая ошибки распознавания для сходных символов. Если штраф отсутствует ($f(\mathrm{G}(\mathrm{V}), \mathrm{W}) = 0$) и сходных символов нет ($\mathrm{substCost}(v_i, w_j) = 0$ или $\mathrm{substCost}(v_i, w_j) = 1$), то $\mathrm{Sim}(\mathrm{V}, \mathrm{W})$ совпадает с расстоянием Левенштейна $d_{\mathrm{LEV}}(\mathrm{V}, \mathrm{W})$. Сходство $\mathrm{Sim}(\mathrm{V}, \mathrm{W})$ также может быть нормализовано аналогично $\rho_{\mathrm{LEV}}(\mathrm{V}, \mathrm{W})$.

Предложенные модификации расстояния Левенштейна позволяют уменьшить число совпадений слов, которые нельзя отождествлять, и одновременно увеличить число совпадений слов, в которых имеются несущественные ошибки распознавания.

## 5. ОПТИМИЗАЦИЯ РЕАЛИЗОВАННЫХ АЛГОРИТМОВ КЛАССИФИКАЦИИ И ПРИВЯЗКИ ДЕЛОВЫХ ДОКУМЕНТОВ

Реализацию описанных алгоритмов классификации и привязки, основанных на отождествлении слов, мы рассмотрим в качестве объекта оптимизации быстродействия.

Высокоуровневая оптимизация быстродействия алгоритмов классификации и привязки основана на создании описания структуры страницы документа и соответствующих фрагментам документа созвездий. При профилировании реализации алгоритмов на языке C++ вычисление расстояния между словами является “горячей точкой” и занимает 20–50% от общих затрат времени на работу алгоритма (5–15 миллисекунд на один документ различного типа). Другими словами, основное время занимает отождествление слов.

Цель оптимизации функций привязки и классификации обусловлена необходимостью применять эти функции не один раз для изображения документа, а столько раз, сколько имеется описаний типов различных возможных документов.

Очевидно, что при анализе фрагмента документа количество кандидатов на отождествление с текстовыми точками модели может быть существенно меньше, чем при анализе всех распознанных слов страницы. Это обеспечивается штрафными функциями $f_1$ и $f_2$, отношениями между точками в созвездии и порогами, применяемыми при вычислении расстояний между V и W с помощью метрик. При обучении моделей классификации (1) на документах более чем на 10 типах деловых документов объем модели составляет более 100 точек, то при задании созвездий в виде простой или вертикальной цепей требуется не более 10 точек. Затраты на работу реализации уменьшаются существенно.

Эффективной оптимизацией вычисления расстояния между словами V и W является вычисление на первом этапе штрафов $\mathrm{Pen}(V, W) = f_1(G(V), W) + f_2(G_2(V, W))$. В случае превышения Pen(V, W) порога $d(V)$. Вычисление по рекуррентной формуле (2), имеющей квадратичную сложность, проводится только в случае, когда $\mathrm{Pen}(V, W) < d(V)$.

Предложенная оптимизация является высокоуровневой и будет давать эффект независимо от архитектурной платформы, на которой исполняется реализация алгоритма. Также была предпринята низкоуровневая оптимизация. Целью этого была реальная потребность. Затраты времени на классификацию и привязку на наборе распознанных слов являются незначительными в схеме обработки, в которой классификация и привязка полей документа проводится один раз для каждого образа документа. Если же классификация и привязка проводятся многократно для нескольких типов документа и многократно применяются к одному набору распознанных слов, то затраты времени увеличиваются вместе с числом применяемых типов. Опишем две оптимизации, направленные на ускорение вычислений для реальных типов центральных процессоров.

Первая оптимизация была связана с вычислением функции substCost(s, c). В реализации алгоритма создавались глобальные (соответствующие документу в целом) и локальные (соответствующие одному слову) таблицы эквивалентности символов, $s \neq c$ для которых $\mathrm{substCost}(s, c) = 0$. При сравнении двух символов проводился поиск в таблице эквивалентности m_nEquChars, имеющий целый 32-разрядный тип, этих символов с учетом перестановки. При профилировании на компьютере Intel(R) Core(TM) i7–4790 CPU3.60 GHz, 16,0 GB, Windows 7 prof 64-bit с помощью ПО MVS Analizer и Intel VTune [15] функция substCost определилась как “горячая точка”. Исходный и ассемблерный коды представлены на рис. 1.

Функция может быть ускорена за счет использования 64-разрядного целого типа (рис. 2). При использовании типа __int64 при компиляции с помощью MSV Compiler для режима x64 количество инструкций для реализации тела цикла уменьшается с 13 до 8. Ускорение на некоторых типах документов составляет 10%.

Другая оптимизация была предназначена для платформы ARM в смартфонах iPhone. Оказалось, что в сложных сценариях многократного применения классификации и привязки реализация доступа к объектам (текстовым точкам, строкам, отношениям между точками) типа get_object(int id, void *pObject) приводят к появлению горячей точки на архитектуре RISC. Это объясняется высокой латентностью операций копирования данных из оперативной памяти. Ускорение достигается при отмене создания новой копии объекта и предоставления непосредственного доступа к объекту.

Описанные алгоритмы были внедрены в SDK Smart Document Engine [16], предназначенный для распознавания гибких деловых документов.

```
int get_substitutionCost_EnableEquChars(uint16_t c1, uint16_t c2) {
      for (int i = 0; i < m_nCountEquChars; i += 2) {
            if ((m_nEquChars[i] == (int)c1 && m_nEquChars[i + 1] == (int)c2) ||
                (m_nEquChars[i] == (int)c2 && m_nEquChars[i + 1] == (int)c1))
                  return 0;
      }
      return c1 != c2;
}
```

```
mov         ecx,dword ptr [rax-4]
cmp         ecx,edx
jne         …
cmp         dword ptr [rax],r8d
je          …
cmp         ecx,r8d
jne         …
cmp         dword ptr [rax],edx
je          …
add         r9d,2
add         rax,8
cmp         r9d,r10d
jl          …
```

**Рис. 1.** Реализация исходного варианта функции substCost.

```
int get_substitutionCost64(uint16_t c1, uint16_t c2) {
      unsigned __int64 nEquChar;
      unsigned __int64 nC_1 = (((unsigned __int64)c1) << 32) |
                              (unsigned __int64)c2;
      unsigned __int64 nC_2 = (((unsigned __int64)c2) << 32) |
                              (unsigned __int64)c1;
      for (int i = 0; i < m_nCountEquChars; i += 2) {
            nEquChar = *(unsigned __int64*)(m_nEquChars + i);
            if (nEquChar == nC_1 || nEquChar == nC_2)
                  return 0;
      }
      return c1 != c2;
}
```

```
mov         rcx,qword ptr [rax]
cmp         rcx,r10
je          …
cmp         rcx,r9
je          …
add         rax,8
cmp         rax,rdx
jb          …
```

**Рис. 2.** Реализация оптимизированного варианта функции substCost.

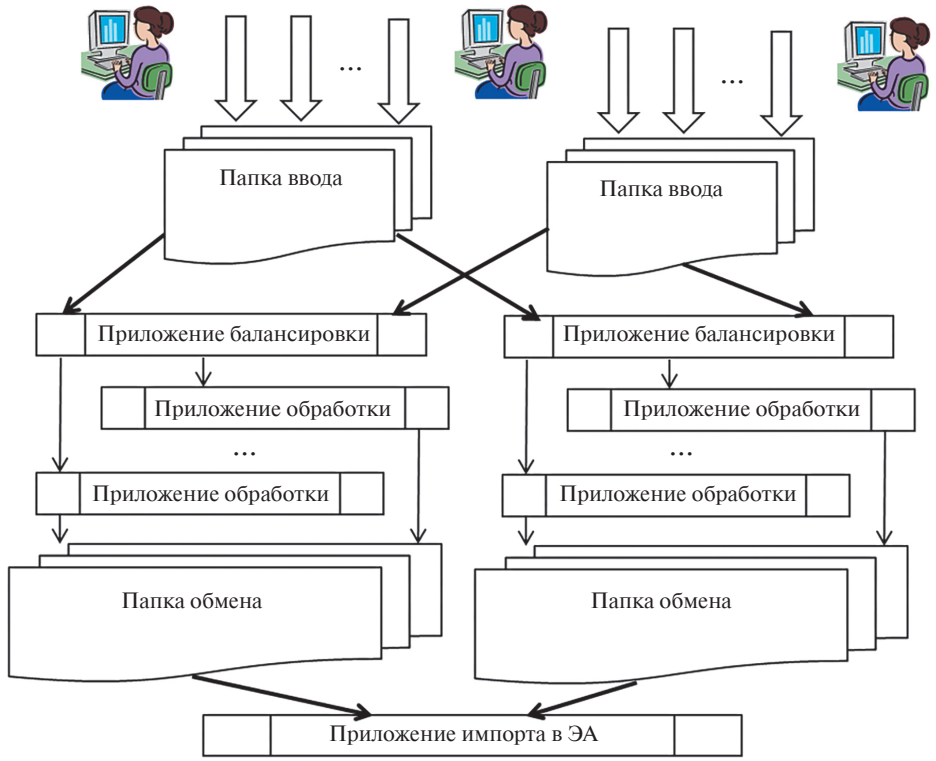



**Рис. 3.** Параллельная реализация системы сортировки.

Основным режимом работы SDK Smart Document Engine является параллельный режим. Параллелизация обеспечивается автоматически с помощью библиотеки Intel TBB [17].

Рассмотрим другой пример оптимизации быстродействия для задачи сортировки большого потока деловых документов (300 000 страниц за 8 ч). После распознавания OCR Tesseract [18] проводилась классификация для 45 типов известных документов. Отметим, что число классов, равное 45, существенно превышает число классов в публичных датасетах [19, 20]. Высокоуровневая оптимизация алгоритмов классификации проводилась с помощью выбора параметров и представления данных для компоненты OCR Tesseract. Существенный эффект был достигнут за счет ограничения области распознавания в каждой из страниц. Для этого на обучающем множестве была выбрана область, в которой находились необходимые для классификации всех документов текстовые особые точки. Результативной оказалась процедура бинаризации образов страниц перед распознаванием. Первоначальной целью бинаризации мы считали улучшение собственно точности распознавания благодаря снятию сложного фона и морфологическим операциям.

Низкоуровневая оптимизация проводилась с помощью выбора параметров компиляции компоненты Intel C++ Compiler XE15.0. Для компилятора была указана опция оптимизации для архитектуры AVX2. Компилятор Intel позволил оптимизировать быстродействие как Tesseract, так и для всех других компонент системы, прежде всего билатеральный фильтр. Ускорение за счет высокоуровневой и низкоуровневой оптимизации составило более 50%.

Параллелизация была реализована с помощью самостоятельных компонент, позволяющих запустить на нескольких многоядерных узлах по нескольку приложений, обрабатывающих страницы в нескольких потоках, а входной поток страниц назначается этим приложениям согласно некоторому алгоритму балансировки (см. рис. 3). Система была реализована с параллелизмом без использования явного управления.

## ЗАКЛЮЧЕНИЕ

В разделах 4 и 5 были рассмотрены примеры высокоуровневой и низкоуровневой оптимизации на примере программ распознавания документов. Описанные методы оптимизации быстродействия программного обеспечения пригодны для более широкого класса приложений для обработки изображений (ПОИ).

Для разработки ПОИ важен выбор готовых или вновь разрабатываемых программных компонент. Этот выбор зависит от формы разработки (проект или собственная разработка). Во всех случаях

уместны оценка сложности выбранных алгоритмов и макетирование ПОИ. Необходимым этапом оптимизации является определение требований к быстродействию программного приложения.

С самого начала разработки важнейшим инструментом высокоуровневой и низкоуровневой оптимизации является профайлер. Этот инструмент применяется для анализа профиля, выбора горячих точек и уточнения ограничений на время выполнения горячих точек. На начальных этапах разработки является полезным анализ исходного кода на эмуляторах будущей вычислительной платформы, в том числе моделирование задержек в подсистеме памяти [21]. Выбор представления изображения может внести существенный вклад в ускорение алгоритма. В качестве примера можно привести использование интегрального представления для извлечения признаков Хаара [22]. При реализации искусственных нейронных сетей для ускорения эффективен выбор размеров слоев сети и представления данных [23, 24].

Расширенные системы инструкций SIMD (MMX, XMM, NEON) эффективно ускоряют алгоритмы обработки изображений и распознавание. Для применения возможностей конкретной платформы могут использоваться как средства компиляторов, так и интринсики. Однако для различных вычислительных платформ различны не только компиляторы, но и наборы интринсиков. Последнее следует учесть при проектировании на предыдущем этапе представления данных, например, для нейронных сетей [23].

В предположении, что обработка изображения занимает время, существенно превышающее время кванта операционной системы, например, это время превышает 100 мс, возможны два способа распараллеливания. Первый способ состоит в использовании средств автоматического распараллеливания [17], второй – непосредственное формирование потоков параллельного управления. Распараллеливание имеет смысл проводить после завершения высокоуровневой и низкоуровневой оптимизации ПОИ.

Предлагаемые методы оптимизации быстродействия программного обеспечения пригодны не только для реализации ПОИ, но и для вычислительных алгоритмов, в которых проводится циклическая обработка информации большого объема.

## СПИСОК ЛИТЕРАТУРЫ


1. *Acar U.A., Blelloch G.E., Harper R.* Selective memorization. ACM SIGPLAN Notices. 2003. V. 38. № 1. P. 14–25. https://doi.org/10.1145/640128.604133
2. *Tatarowicz A.L., Curino C., Jones E.P.C. and Madden S.* Lookup Tables: Fine-Grained Partitioning for Distributed Databases. IEEE28th International Conference on Data Engineering. 2012. P. 102–113. https://doi.org/10.1109/ICDE.2012.26
3. *Harris D.M., Harris S.L.* Digital Design and Computer Architecture, 2nd Edition. Morgam Kaufmann is an imprint of Elsevier Inc., Waltham, 2013. 720 p.
4. *Rusiñol M., Frinken V., Karatzas D., Bagdanov A.D., Lladós J.* Multimodal page classification in Administrative document image streams. In: IJDAR. 2014. V. 17. № 4. P. 331–341. https://doi.org/10.1007/s10032-014-0225-8
5. *Slavin O.A., Pliskin E.L.* Method for analyzing the structure of noisy images of administrative documents. Bulletin of the South Ural State University. Ser. Mathematical Modelling, Programming & Computer Software (Bulletin SUSU MMCS). 2022. V. 15. № 4. P. 80–89. https://doi.org/10.14529/mmp220407
6. *Slavin O.A., Farsobina V., Myshev A.V.* Analyzing the content of business documents recognized with a large number of errors using modified Levenshtein distance. Cyber-Physical Systems: Intelligent Models and Algorithms. Springer Nature Switzerland AG. 2022. V. 417. P. 267–279. https://doi.org/10.1007/978-3-030-95116-0
7. *Bellavia F.* SIFT Matching by Context Exposed. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022. https://doi.org/10.1109/TPAMI.2022.3161853
8. *Bay H., Tuytelaars T., Van Gool Luc.* SURF: Speeded Up Robust Features. Computer Vision and Image Understanding – CVIU. 2003. V. 110. № 3. P. 404–417.
9. *Du X., Wumo P., Bui T.D.* Text line segmentation in handwritten documents using Mumford–Shah model. Pattern Recognition. 2009. V. 42. P. 3136–3145. https://doi.org/10.1016/j.patcog.2008.12.021
10. *Maraj A., Martin M.V., Makrehchi M.* A More Effective Sentence-Wise Text Segmentation Approach Using BERT. In: Lladós J., Lopresti D., Uchida S. (eds) Document Analysis and Recognition – ICDAR2021. Lecture Notes in Computer Science, Springer, Cham. 2021. V. 12824. https://doi.org/10.1007/978-3-030-86337-1_16
11. *Kravets A.G., Salnikova N.A., Shestopalova E.L.* Development of a Module for Predictive Modeling of Technological Development Trends. Cyber-Physical Systems. 2021. P. 125–136. https://doi.org/10.1007/978-3-030-67892-0_11
12. *Sabitov A., Minnikhanov R., Dagaeva M., Katasev A., Asliamov T.* Text Classification in Emergency Calls Management Systems. Cyber-Physical Systems. 2021. P. 199–210. https://doi.org/10.1007/978-3-030-67892-0_17
13. *Deza M.M., Deza E.* Encyclopedia of distances. Springer-Verlag, Berlin, xiv+590 pp. (2009)

14. *Yujian L., Bo L.* A Normalized Levenshtein Distance Metric // IEEE Transactions on Pattern Analysis and Machine Intelligence. V. 29. № 6. P. 1091–1095. https://doi.org/10.1109/TPAMI.2007.1078 (2007)
15. Intel® VTune™ Profiler Performance Analysis Cookbook. https://www.intel.com/content/www/us/en/docs/vtune-profiler/cookbook/2023–2/overview.html. Accessed 23 Sep. 2023.
16. Smart Document Engine – automatic analysis and data extraction from business documents for desktop, server and mobile platforms. https://smartengines.com/ocr-engines/document-scanner. Accessed 23 Sep. 2023.
17. Intel(R) oneAPI Threading Building Blocks (oneTBB) Developer Guide and API Reference. https://www.intel.com/content/www/us/en/docs/onetbb/developer-guide-api-reference/2021–10/overview.html. Accessed 23 Sep. 2023.
18. OCR Tesseract. https://github.com/tesseract-ocr/tesseract. Accessed 23 Sep. 2023.
19. NIST Special Database. https://www.nist.gov/srd/nist-special-database-2. Accessed 23 Sep. 2023.
20. Tobacco-3482. https://www.kaggle.com/patrickaudriaz/tobacco3482jpg. Accessed 23 Sep. 2023.
21. *Kravets A.G., Egunov V.* The Software Cache Optimization-Based Method for Decreasing Energy Consumption of Computational Clusters // Energies [Special Issue Smart Energy and Sustainable Environment]. 2022. V. 15. № 20. P. 7509. https://doi.org/10.3390/en15207509
22. *Crow F.C.* Summed-area tables for texture mapping ACM SIGGRAPH Computer Graphics. 1984. V. 18. № 3. P. 207–212.
23. *Trusov A., Limonova E., Nikolaev D., Arlazarov V.V.* 4.6-bit Quantization for Fast and Accurate Neural Network Inference on CPUs // Mathematics. 2024. V. 12. № 5. P. 651. https://doi.org/10.3390/math12050651
24. *Rybakova E.O., Limonova E.E., Nikolaev D.P.* Fast Gaussian Filter Approximations Comparison on SIMD Computing Platforms // Applied Sciences. 2024. V. 14. № 11. P. 4664. https://doi.org/10.3390/app14114664

# OPTIMIZATION OF SOFTWARE PERFORMANCE FOR CLASSIFICATION AND LINKING OF ADMINISTRATIVE DOCUMENTS

**© 2024 O. A. Slavin[a, b]**

[a] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences*
*44-2 Vavilova str, Moscow, 119333 Russia*
[b] *LLC Smart Engines Service*
*9 Prospect 60-Letiya Oktyabrya, Moscow, 117312 Russia*

The paper discusses technologies for optimizing software performance. Optimization methods are divided into high-level and low-level, as well as parallelization. An algorithm for classifying and linking fields in a recognized image of an administrative document is described. The features of the implementation of classification and linking tasks are listed, consisting of the use of constellations of text feature points and the modified Levenshtein distance. SDK Smart Document Engine and OCR Tesseract were used. Several ways are described to optimize the performance of the functions for classifying and linking document content. Optimization of the performance of the system for sorting a stream of images of administrative documents is also described. The proposed methods for optimizing software performance are suitable not only for implementing image processing algorithms but also for computational algorithms in which cyclic information processing is carried out. The method can be applied in modern CAD systems to analyze the content of recognized textual files.

*Keywords*: document recognition, flexible document, rigid document, text feature keypoint, acceleration