ISSN 0132-3474

Номер 4

Июль - Август 2024

ПРОГРАММИРОВАНИЕ

НАУКА
1727

СОДЕРЖАНИЕ

Номер 4, 2024

Contents

No. 4, 2024

INFORMATION SECURITY

COMPUTER GRAPHICS AND VISUALIZATION

SOFTWARE ENGINEERING, TESTING AND VERIFICATION

ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 3–12

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.75

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ХЭШИРОВАНИЯ С ТОЧКИ ЗРЕНИЯ ПРИМЕНЕНИЯ В СХЕМАХ ZK-SNARK В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ

© 2024 г. Д. О. Кондырев[a,*]

[a] *Новосибирский государственный университет*
630090 Новосибирск, ул. Пирогова, д. 2, Россия
**E-mail: dkondyrev@gmail.com*

Поступила в редакцию 26.05.2023
После доработки 20.11.2023
Принята к публикации 22.01.2024

В работе представлен сравнительный анализ эффективности алгоритмов хэширования с точки зрения применимости в системах на основе протокола неинтерактивного доказательства знания с нулевым разглашением zk-SNARK. Были рассмотрены хэш-функции sha256, sha3, poseidon, mime, blake2, которые находят наибольшее применение в современных распределенных реестрах. Для проведения экспериментов с замером параметров была разработана инфраструктура на основе набора инструментов ZoKrates. На основе полученных результатов определены границы практической применимости алгоритмов для задачи доказательства знания прообраза хэш-функции с помощью схем zk-SNARK в распределенных реестрах, а также выявлены возникающие проблемы эффективности.

Ключевые слова: распределенные реестры, доказательство с нулевым разглашением, zk-SNARK, R1CS, хеш-функции, эффективность алгоритма

DOI: 10.31857/S0132347424040012, **EDN:** PTJOCY

1. ВВЕДЕНИЕ

Распределенные реестры находят все большее применение в различных задачах. Помимо чисто исследовательских проектов начинают появляться первые промышленные системы, построенные на базе технологии распределенного реестра. При этом одним из факторов, существенно ограничивающих применение подобных технологий, является безопасность информации, которая должна обеспечиваться такими системами. В связи с этим активно ведутся исследования, направленные на создание различных протоколов и способов защиты информации для распределенных реестров [1].

Существенным прорывом в этом направлении можно считать появление в 2013 г. zk-SNARK [2], который сделал возможным эффективное использование неинтерактивных протоколов доказательства с нулевым разглашением в распределенных реестрах.

При том что теоретически zk-SNARK позволяет задавать произвольные ограничения [2], на практике возникают ограничения по времени работы и по памяти. При этом традиционные алгоритмы, оптимизированные для эффективного программного вычисления или реализации в виде аппаратного обеспечения, оказываются не настолько эффективными для использования в zk-SNARK. В связи с этим встает задача разработки специальных алгоритмов, оптимизированных под zk-SNARK.

Отдельным важным классом криптографических алгоритмов, которые находят широкое применение в распределенных реестрах и, в частности, в сокрытии информации с помощью доказательства с нулевым разглашением, являются хэш-функции. Задача доказательства знания прообраза хэш-функции возникла уже в первых блокчейн-системах на основе zk-SNARK [3]. После этого применение хэш-функций в системах ограничений, накладываемых на скрываемые данные, расширялось.

При том что рассматриваемые криптографические хэш-функции довольно давно используются в различных блокчейн-приложениях zk-SNARK, не было проведено полноценного исследования производительности для конкретных схем.

Целью данной работы было провести полный сравнительный анализ производительности наиболее часто применяемых хэш-функций и определить для них границы практической применимости в системах на базе технологии распределенного реестра. В работе рассматриваются хэш-функции sha256 [4], sha3 [5], blake2 [6], poseidon [7], mimc [8]. Для проведения экспериментов с замером параметров была разработана инфраструктура на основе ZoKrates.

2. ДОКАЗАТЕЛЬСТВО С НУЛЕВЫМ РАЗГЛАШЕНИЕМ

Доказательство с нулевым разглашением – криптографический протокол, в котором принимают участие две стороны – доказывающая и проверяющая (верификатор).

Цель протокола заключается в том, чтобы верификатор мог убедиться, что доказывающая сторона обладает знанием секретного параметра. При этом сам секретный параметр не должен раскрываться верификатору или кому-либо еще [9].

Доказательство с нулевым разглашением по определению должно удовлетворять следующим трем свойствам:

• **Полнота:** если утверждение верно и обе стороны следуют одному и тому же протоколу, то верификатор может убедиться в истинности утверждения.

• **Устойчивость:** если утверждение ложно, верификатор с большой вероятностью не будет убежден в его истинности.

• **Нулевое разглашение:** верификатор не получает дополнительной информации.

Доказательства с нулевым разглашением применяются для решения широкого круга задач. Примерами практического применения могут служить системы анонимного проверяемого голосования, безопасные аукционы, протоколы аутентификации [10].

В распределенных реестрах основное применение протоколы доказательства с нулевым разглашением нашли в решении проблемы приватности транзакций и смарт-контрактов.

Одним из важных примеров использования протоколов неинтерактивного доказательства знания с нулевым разглашением (NIZK) является подтверждение права собственности на цифровые активы, хранящиеся в виде токенов в блокчейне [11]. Цифровые активы представляют собой набор двоичных данных, которые являются уникально идентифицируемыми и обладают определенной ценностью. Если два пользователя хотят обменять свои цифровые активы, в процессе обмена может произойти утечка информации о конфиденциальности пользователя, которая включает в себя идентификационные данные и содержимое обмененных цифровых активов. Благодаря использованию протоколов доказательства с нулевым разглашением можно обмениваться цифровыми активами без утечки конфиденциальной информации пользователя. Кроме того, они позволяют сгенерировать проверяемое доказательство, которое подтверждает корректность процесса обмена активами [10].

Доказательства с нулевым разглашением применяются также для решения проблемы масштабируемости в механизмах ZK Rollups [12].

Более подробно применение протоколов доказательства с нулевым разглашением в распределенных реестрах рассматривается в статьях [10–14].

3. ТРЕБОВАНИЯ К ПРОТОКОЛАМ ДОКАЗАТЕЛЬСТВА С НУЛЕВЫМ РАЗГЛАШЕНИЕМ В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ

Технология распределенных реестров позволяет построить децентрализованную систему хранения информации. Данные в таких системах хранятся распределенно на узлах сети, история транзакций не может быть изменена или удалена. При этом в большинстве реализаций каждый узел системы поддерживает в актуальном состоянии полную копию всего реестра. Такой подход позволяет повысить надежность и отказоустойчивость, но ценой дополнительных накладных расходов по памяти.

Особенности технологии распределенных реестров накладывают ряд ограничений на используемые протоколы доказательства с нулевым разглашением. Для протоколов, используемых в распределенных реестрах, следующие требования являются критическими:

• **Ограничение на количество раундов взаимодействия участников.** Это свойство вытекает из децентрализованного характера системы. Протоколы должны учитывать тот факт, что пользователи могут не быть в сети одновременно.

• **Ограничения по памяти.** Вся история транзакций может сохраняться на каждом узле. Это накладывает серьезные ограничения на размер транзакций (для большинства систем он не должен превышать нескольких сотен байт). Поэтому любые дополнительные данные, которые необходимо хранить в распределенном реестре, включая

данные криптографических протоколов, должны соответствовать этому требованию.

• **Вычислительная эффективность.** Проверка корректности транзакций в сети происходит на каждом узле. Поэтому вычислительная эффективность отдельных операций, выполняемых в ходе проверки, крайне важна. Если протокол предполагает вычисления, выполняемые во время валидации транзакций, то они должны быть максимально оптимизированы по времени выполнения. Иначе создание новых блоков будет происходить медленно. В итоге пропускная способность системы (количество транзакций в секунду) упадет, что может сделать применение протокола практически нецелесообразным [11, 15].

Стоит отметить, что перечисленные ограничения являются довольно серьезными и существенно ограничивают множество протоколов, которые могут быть использованы в системах на базе технологии распределенного реестра.

В последние годы активно ведутся исследования по разработке новых криптографических протоколов для сохранения приватности и анонимности в таких системах, а также адаптации и модификации существующих протоколов.

4. ПРОТОКОЛ ZK-SNARK

zk-SNARK (zero-knowledge Succinct Non-Interactive Argument of Knowledge) – это криптографический протокол неинтерактивного доказательства знания с нулевым разглашением. Он позволяет доказывать, что некоторые приватные данные удовлетворяют системе ограничений, выраженной в виде арифметической схемы C, не раскрывая эти данные.

zk-SNARK представляет собой тройку алгоритмов полиномиального времени выполнения (Gen, P, V):

• $Gen(\lambda, C) \to (pk, vk)$. Этот алгоритм принимает в качестве входных данных параметр безопасности λ и арифметическую схему C. На их основе генератор Gen создает пару ключей – ключ доказательства (pk, proving key) и ключ верификации (vk, verification key). Оба ключа публикуются как открытые параметры и могут использоваться любое количество раз для создания доказательства и проверки его корректности.

• $P(pk, x, a) \to \pi$. Принимая на вход ключ доказательства pk и любые (x, a), где x – публичные данные, a – секретный параметр, алгоритм P выводит неинтерактивное доказательство π.

• $V(vk, x, \pi) \to b$. Принимая на вход ключ верификации vk, публичные данные x и доказательство π, верификатор V выдает $b = 1$, если доказательство является корректным, и 0 иначе.

Данная конструкция удовлетворяет всем требованиям, предъявляемым к алгоритмам доказательства с нулевым разглашением [3].

Преимущество zk-SNARK над другими протоколами доказательства с нулевым разглашением заключается в гарантиях эффективности: длина доказательства зависит только от параметра безопасности, а время проверки не зависит от размера схемы и секретного параметра.

Таким образом, zk-SNARK можно рассматривать как неинтерактивный протокол с коротким доказательством и быстрым временем верификации, что сделало его оптимально подходящим для использования в распределенных реестрах [15].

Однако, при том что zk-SNARK подходит для применения в распределенных реестрах, из-за ограничений на память и вычислительную эффективность, про которые рассказывалось в предыдущем разделе, на практике может быть реализована не любая по сложности схема C. Поэтому не любой криптографический алгоритм может быть использован для описания ограничений, накладываемых на скрываемые с помощью zk-SNARK данные.

5. СИСТЕМЫ ОГРАНИЧЕНИЙ РАНГА 1 (R1CS)

zk-SNARK позволяет доказывать произвольные NP-утверждения. Следовательно, чтобы быть доказуемым, вычисление C должно быть сформулировано как пример NP-полной задачи, например, как задача о выполнимости арифметической схемы или удовлетворении ограничений.

Арифметическая схема C над полем F – это направленный ациклический граф, где каждой вершине соответствует операция сложения или умножения над полем F, а каждому ребру – значение из F.

Возьмем в качестве примера следующую арифметическую схему с входными значениями $i_0, ..., i_3$, выходами o_0, o_1 и промежуточными выходами w_0, w_1, w_2 (рис. 1).

Алгебраическое представление такой арифметической схемы задается следующими ограничениями:

$$w_0 = i_0 + i_1, \qquad o_0 = i_0 * w_0,$$
$$w_1 = w_0 * i_2, \qquad o_1 = w_1 + w_2.$$
$$w_2 = i_2 * i_3,$$

Системы ограничений ранга 1 (rank-1 constraint systems, R1CS) можно рассматривать как краткое алгебраическое представление арифметических

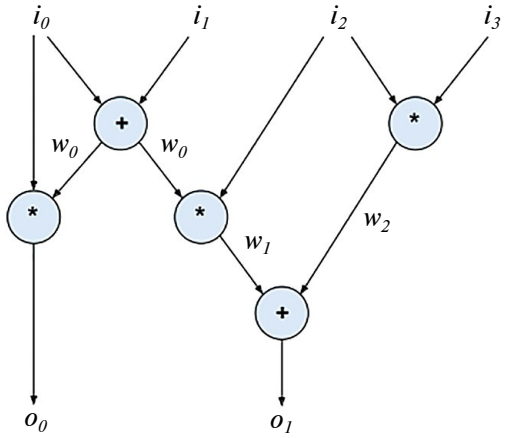

Рис. 1. Арифметическая схема.

схем. R1CS позволяют представить схему C в виде, удобном для дальнейшего эффективного доказательства выполнимости этой схемы с помощью zk-SNARK.

Чтобы привести арифметическую схему к такому представлению, ограничения умножения можно представить в виде

$$w_1 = w_0 * i_2 \Leftrightarrow$$

$$\Leftrightarrow (1 * w_1) = (1 * w_0) \times (1 * i_2) \Leftrightarrow$$

$$\Leftrightarrow \left\langle \begin{pmatrix} 0\\0\\0\\0\\0\\1\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix} \right\rangle * \left\langle \begin{pmatrix} 0\\0\\0\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix} \right\rangle = \left\langle \begin{pmatrix} 0\\0\\0\\0\\0\\1\\0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix} \right\rangle$$

Если перейти к выражению всей схемы через матричное произведение, то получим систему ограничений ранга 1:

$$\begin{pmatrix} 0&1&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0 \end{pmatrix} \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix}$$

$$\odot \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0 \end{pmatrix} \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix}$$

$$= \begin{pmatrix} 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \begin{pmatrix} 1\\i_0\\i_1\\i_2\\i_3\\w_0\\w_1\\w_2\\o_0\\o_1 \end{pmatrix}$$

Далее, имея систему ограничений в R1CS-представлении, можно сформулировать задачу выполнимости в следующем виде:

Можно ли дополнить вектор, состоящий из входных и выходных значений (i, o)*, вектором* w *таким образом, что*

$$\mathbf{A}(1, i, o, w)^{\mathrm{T}} \odot \mathbf{B}(1, i, o, w)^{\mathrm{T}} = \mathbf{C}(1, i, o, w)^{\mathrm{T}},$$

где $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ *– заданные матрицы.*

Благодаря краткости и лаконичности R1CS превратилась в де-факто стандартный формат ввода для реализаций zk-SNARK. Поскольку в R1CS используется стандартная линейная алгебра, они также хорошо подходят для описания криптографических протоколов и теоретического анализа [16].

Эффективность алгоритмов установочной фазы *Gen* и генерации доказательства *P* по времени работы и по памяти напрямую зависит от количества ограничений в R1CS-представлении. Таким образом, оптимизация, которая уменьшает количество ограничений в R1CS без изменения ее выполнимости, имеет решающее значение для эффективности.

6. ПРОБЛЕМА ЭФФЕКТИВНОСТИ

Сложность доказательства выполнения арифметической схемы с помощью zk-SNARK составляет $O(n \log n)$, где n – число операций умноже-

ния. Таким образом, эффективность напрямую определяется n.

Схема C более эффективна, чем C', тогда и только тогда, когда обе схемы вычисляют одну и ту же функцию и $n_C < n_{C'}$ [16].

В контексте реальных приложений zk-SNARK асимптотические результаты менее актуальны, и конкретная эффективность имеет решающее значение для обеспечения практической применимости. Особенно это актуально в распределенных реестрах с их дополнительными ограничениями по времени работы и по памяти.

Поэтому для конкретных алгоритмов важно понимать, насколько они эффективны с точки зрения количества ограничений.

7. ХЭШ-ФУНКЦИИ В РАСПРЕДЕЛЕННЫХ РЕЕСТРАХ

В ходе работы было проведено сравнение эффективности различных криптографических хэш-функций с точки зрения реализации в R1CS.

Хэш-функция H – это функция, которая принимает на вход данные произвольного размера и сопоставляет их с выходными данными фиксированного размера. Криптографические хэш-функции должны обладать дополнительными свойствами:

- стойкость к коллизиям – трудно найти такие a и b, что $H(a) = H(b)$;
- стойкость к поиску первого прообраза – для заданного y трудно найти такое входное значение a, что $H(a) = y$;
- стойкость к поиску второго прообраза – для заданных входных данных a и выходного значени $y = H(a)$ трудно найти вторые входные данные b такие, что $H(b) = y$ [13].

В распределенных реестрах хэш-функции применяются при решении многих задач, таких как:

- формирование адресов аккаунтов;
- защита данных в блоках транзакций;
- подпись транзакций;
- обеспечение работы алгоритмов консенсуса (например, нахождение подходящего значения в алгоритме на основе доказательства выполнения работы (Proof-of-Work)) [13, 17].

Отдельной категорией задач, в которых применяются хэш-функции, стало формирование доказательств с нулевым разглашением. Как было упомянуто выше, важным применением таких доказательств в распределенных реестрах является задача подтверждения права собственности на цифровые активы.

Впервые хэш-функции для решения подобной задачи были применены в системе Zerocash. Zerocash реализует схему децентрализованных анонимных платежей, которая позволяет пользователям осуществлять переводы, сохраняя приватность. Транзакции переводов скрывают отправителя платежа, адресата и сумму перевода. Для доказательства корректности проведения транзакций используются схемы zk-SNARK, которые строятся с использованием хэш-функции sha256 [3].

В более общем виде задачу можно сформулировать как доказательство владения определенными данными без необходимости их раскрытия какой-либо стороне. Один из вариантов решения – применить хэш-функцию к данным и с помощью схемы zk-SNARK сгенерировать доказательство знания прообраза. Это позволяет верификатору убедиться, что доказывающая сторона обладает определенными данными таким образом, чтобы не размещать сами эти данные в открытом виде в транзакциях распределенного реестра.

При таком подходе возрастает важность оптимизации хэш-функций для применения в схемах zk-SNARK, поскольку именно вычисление хэш-функции, как самая затратная операция, определяет сложность всей схемы.

В первых распределенных реестрах наибольшее применение нашли функции sha256. В более поздних системах стали применяться алгоритмы sha3 и blake2 [13, 17].

Однако стало понятно, что классические алгоритмы не максимально эффективно решают возникающие задачи. В связи с этим стали разрабатываться специально оптимизированные для применения в zk-SNARK алгоритмы хэширования, такие как mimc [8] и poseidon [7].

8. КРИПТОГРАФИЧЕСКАЯ СТОЙКОСТЬ И ЭФФЕКТИВНОСТЬ ХЭШ-ФУНКЦИЙ

В табл. 1 приведен сравнительный анализ рассматриваемых хэш-функций по уровню криптографической стойкости (security level) и производительности для решения стандартных задач. В качестве задач для сравнения взяты вычисление хэш-функции от данных размером не менее 512 бит и построение дерева Меркла с 2^{20} элементами.

Указанные значения скорости работы хэш-функций, а также более подробные данные о производительности приведены в статье [18].

В работах [7, 8] показано, что с помощью конструкций poseidon и mimc можно достичь

Таблица 1. Сравнительный анализ криптографической стойкости и производительности хэш-функций

	Уровень криптографической стойкости (бит)	Вычисление хэш-функции от данных размером 512 бит (мс)	Построение дерева Меркла с 2^{20} элементами (с)
sha256	128	0.32	0.624
sha3	128	0.42	0.439
blake2	128	0.21	0.222
poseidon	128	20	22.6
mimc	128	38	42.2

необходимого уровня криптографической стойкости. В рамках работы рассматривались варианты poseidon и mimc, которые обеспечивают стойкость 128 бит, что соответствует уровню алгоритмов sha256, sha3 и blake2.

Poseidon и mimc представляют из себя узкоспециализированные хэш-функции и, как видно из таблицы, для решения стандартных задач их результаты оказываются существенно слабее. Так происходит, поскольку каждый раунд этих схем требует выполнения операций умножения в конечном поле, что является относительно дорогой операцией, требующей сотни циклов CPU, по сравнению с битовыми операциями, используемыми в традиционных хэш-функциях [18]. По этой же причине эти алгоритмы оказываются эффективнее в схемах zk-SNARK.

9. ПРИМЕНИМОСТЬ АЛГОРИТМОВ ДЛЯ ПРОТОКОЛА ZK-SNARK

Теоретический анализ большей или меньшей применимости в zk-SNARK классических хэш-функций не проводился, поскольку при компиляции в R1CS-представление получаются системы ограничений большого размера. Поэтому сравнения производительности проводят на конкретных реализациях отдельных схем, замеряя количество ограничений R1CS [7, 18].

Одной из основных причин, почему традиционные алгоритмы плохо оптимизированы для zk-SNARK является тот факт, что доказательства создаются для утверждений, сформулированных в виде операций над конечными полями, тогда как классические алгоритмы работают над битовыми строками. Необходимый перевод кода, оперирующего битами, в арифметику конечных полей требует значительных накладных расходов [18].

Специально разработанные для zk-SNARK алгоритмы имеют конструкцию, в которую заложены операции, оптимальные с точки зрения внутреннего представления R1CS. Так, mimc и poseidon в качестве нелинейной функции в раундах используют x^{α} [7, 8]. Для рассматриваемой в работе реализации poseidon S-блоки имеют вид $S-box(x)=x^5$, что выражается с помощью трех ограничений R1CS.

10. ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Для проведения вычислительных экспериментов использовался ZoKrates – набор программных инструментов для создания доказательств знания с нулевым разглашением, использующий zk-SNARK в качестве системы проверки.

ZoKrates скрывает значительную сложность, присущую доказательствам с нулевым разглашением, и предоставляет разработчикам более высокоуровневые программные абстракции для реализации системы ограничений. Для этого он определяет предметно-ориентированный язык, который позволяет разработчикам задавать доказуемые вычисления без необходимости разбираться в низкоуровневых деталях реализации системы доказательств.

Компилятор ZoKrates транслирует код, написанный на этом языке, в доказуемые системы ограничений. Кроме того, ZoKrates позволяет выполнять реализованные программы, генерировать доказательства и выполнять проверку корректности доказательств [19].

В качестве задачи для замера эффективности работы различных алгоритмов использовалась задача проверки знания прообраза хэш-функций – чтобы исключить дополнительные накладные расходы и сравнивать непосредственно эффективность реализации хэш-функций.

Реализации сравниваемых хэш-функций взяты из стандартной библиотеки ZoKrates [20]. В качестве протокола zk-SNARK используется система доказательства Groth16 [21] с эллиптической кривой ALT_BN128.

Программа принимает значение входной строки в виде секретного параметра, далее в коде вычисляется хэш-функция от этих секретных данных, после чего происходит сравнение с заранее вычисленным правильным значением. Пример кода для функции sha3 приведен в листинге 1.

Листинг 1: Код функции проверки знания прообраза хэша на предметно-ориентированном языке ZoKrates

```
import "hashes/sha3/256bit" as sha3

def main(private u64[32] a):
  u64[4] h = sha3(a)
  assert(h[0] == 18011198171195110515)
  assert(h[1] == 11930925129043369621)
  assert(h[2] == 7871165761403032144)
  assert(h[3] == 16761886150217802019)
  return
```

zk-SNARK гарантирует, что размер доказательства и время его проверки не зависят от сложности системы ограничений. Однако, другие параметры могут отличаться и именно по ним можно определить эффективность того или иного алгоритма и возможность его применения для определенной задачи. В проведенных экспериментах измерялись следующие параметры сравниваемых алгоритмов:

• длина ключа доказательства и ключа верификации;

• время установочной фазы протокола (алгоритма генерации ключей *Gen*);

• количество ограничений в R1CS-представлении алгоритма;

• время генерации доказательства.

Для повышения точности каждый эксперимент с замером времени проводился $N = 10$ раз, после чего полученные значения времени усреднялись.

Инфраструктура для измерения была реализована на Python 3.10. Она включает в себя:

• автоматический генератор кода функций проверки прообраза для различных входных значений на основе шаблонов;

• автоматизацию запуска различных команд ZoKrates (компиляция кода, генерация параметра безопасности, генерация доказательства);

• функции многократного выполнения экспериментов и замера времени.

Вычислительные эксперименты проводились на компьютере с процессором Intel Core i5-11600K, 32 Гб оперативной памяти и SSD 500 Гб.

11. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА

Ниже приведены графики, на которых отображены полученные результаты для различных хэш-функций в зависимости от размера входных данных (рис. 2–5).

Исходя из полученных экспериментальных данных можно сделать следующие выводы:

• Количество ограничений во всех алгоритмах растет линейно в зависимости от входа. При этом наибольшую скорость роста демонстрируют sha3, sha256 и blake2.

• Как следствие количества ограничений в схеме с ростом размера входных данных увеличивается время выполнения установочной фазы и генерации доказательства. Алгоритмы sha3 и blake2 показывают линейный рост, при этом довольно быстрый. При значениях в 2 Кб генерация доказательства превышает 60 и 20 с соответственно. Алгоритм sha256 оказывается еще менее эффективным, поскольку показывает нелинейный рост времени генерации доказательства и на входных данных в 2 Кб превышает 10 мин.

Время установочной фазы начинает насчитывать несколько минут у всех трех алгоритмов уже при значениях больше 1 Кб. Однако это на так существенно влияет на применимость, поскольку эта часть протокола отрабатывает только один раз для каждой новой схемы.

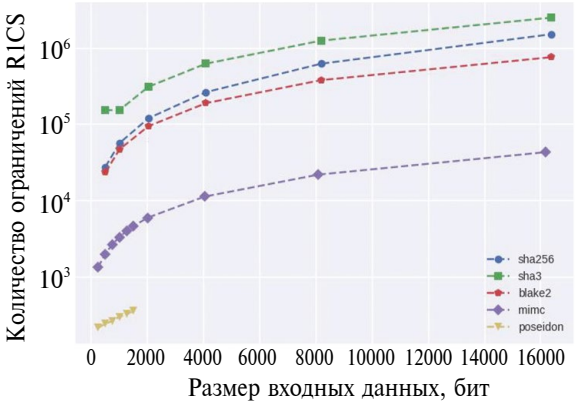

Рис. 2. Количество ограничений в зависимости от размера входных данных.

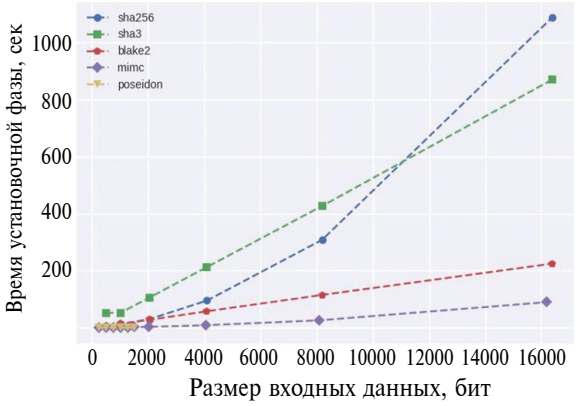

Рис. 3. Время работы алгоритма установочной фазы в зависимости от размера входных данных.

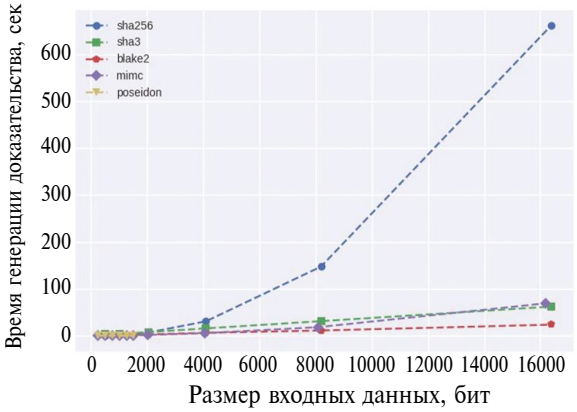

Рис. 4. Время работы алгоритма генерации доказательства в зависимости от размера входных данных.

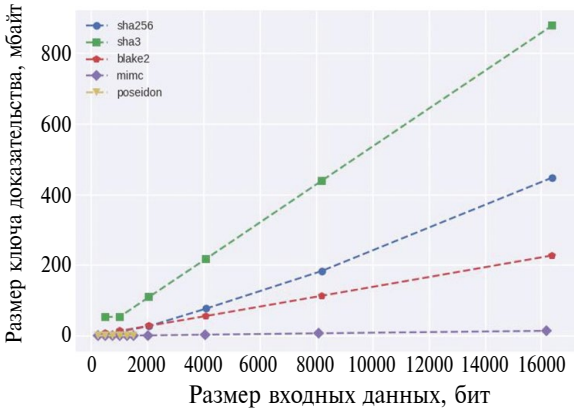

Рис. 5. Размер ключа доказательства в зависимости от размера входных данных.

Таким образом, классические алгоритмы плохо подходят для распределенных реестров. Уже при входных данных размером в несколько килобайт размер и время становятся слишком большими для практических задач, которые предполагают частую генерацию доказательств (сопоставимую со временем появления новых блоков транзакций в системе). Отсюда можно сделать вывод, что в таких системах они могут только ограниченно применяться для доказательства знания прообраза хэш-функции от небольшого объема данных.

Хэш-функции mimc и poseidon лучше справляются с этой задачей, также показывая линейный рост времени установочной фазы и генерации доказательства, но значительно более медленный.

• С увеличением размера входных данных и сложности схемы растет и размер ключа доказательства.

Даже для алгоритма mimc при входных данных в 1.5 Кб размер ключа доказательства превышает 10 Мб. У sha256 и blake2 он растет еще быстрее и при таких же входных данных занимает уже сотни Мб. sha3 оказывается наименее эффективным по памяти – при входных данных более 2.5 Кб получаем размер ключа больше 1 Гб.

Поскольку на размер блока в наиболее распространенных распределенных реестрах накладываются ограничения и он как правило не превышает нескольких мегабайт, это не позволяет эффективно хранить ключи доказательства непосредственно в распределенном реестре.

• Наилучшие результаты по всем параметрам показал алгоритм poseidon, однако его удалось проверить только на небольших размерах входных данных из-за ограничений реализации.

12. ЗАКЛЮЧЕНИЕ

В результате работы был проведен сравнительный анализ эффективности алгоритмов хэширования с точки зрения применимости в протоколах неинтерактивного доказательства знания с нулевым разглашением zk-SNARK в распределенных реестрах. Проведенные эксперименты подтвердили актуальность проблемы оптимизации криптографических алгоритмов для использования в схемах zk-SNARK. Лучшие результаты показали специально разработанные для использования в zk-SNARK алгоритмы poseidon и mimc. Классические алгоритмы sha256, sha3 и blake2 оказываются ограниченно применимыми для доказательства знания прообраза хэш-функции только от небольшого объема данных. При этом определенные проблемы характерны для всех исследованных алгоритмов, поскольку с ростом размера входных данных растет размер ключа доказательства, что при достаточно небольших объемах входных данных уже не позволяет эффективно хранить ключ в распределенном реестре.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке Математического Центра в Академгородке, соглашение с Министерством науки и высшего образования Российской Федерации № 075-15-2022-282.

СПИСОК ЛИТЕРАТУРЫ

1. *Kondyrev D.O.* Overview of privacy preserving technologies for distributed ledgers // Eurasian Journal of Mathematical and Computer Applications. 2021. V. 9. № 1. P. 55–68.
2. *Ben-Sasson E., Chiesa A., Genkin D., Tromer E., Virza M.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // CRYPTO'2013, LNCS. 2013. V. 8043. P. 90–108.

3. *Ben-Sasson E., Chiesa A., Garman C., Green M., Miers I., Tromer E., Virza M.* Zerocash: Decentralized anonymous payments from bitcoin // IEEE Symp. Security and Privacy, San Jose, USA. 2014. P. 459–474.
4. NIST. FIPS PUB180–2, Secure hash standard. 2002.
5. NIST. FIPS PUB202, SHA-3 standard: permutation-based hash and extendable-output functions. 2015.
6. *Aumasson J.P., Neves S., Wilcox-O'Hearn Z., Winnerlein C.* BLAKE2: simpler, smaller, fast as MD5 // ACNS2013, Proceedings of the 11th International Conference Applied Cryptography and Network Security, Banff, AB, Canada. 2013. V. 7954 of LNCS, P. 119–135.
7. *Grassi L., Khovratovich D., Rechberger C., Roy A., Schofnegger M.* Poseidon: A New Hash Function for Zero-Knowledge Proof Systems // Proceedings of the 30th USENIX Security Symposium. 2021. V. 2021.
8. *Albrecht M., Grassi L., Rechberger C., Roy A., Tiessen T.* MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity // ASIACRYPT 2016. Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam. 2016. Part I.V. 10031 of LNCS. P. 191–219.
9. *Шнайер Б.* Прикладная криптография: протоколы, алгоритмы и исходные коды на языке С. 2-е изд. СПб.: Альфа-книга, 2018. 1040 с.
10. *Sun X., Yu F.R., Zhang P., Sun Z., Xie W., Peng X.* A survey on zero-knowledge proof in blockchain // IEEE network. 2021. V. 35. № 4. P. 198–205.
11. *Konkin A., Zapechnikov S.* Zero knowledge proof and ZK-SNARK for private blockchains // Journal of Computer Virology and Hacking Techniques. 2023. P. 1–7.
12. *Thibault L.T., Sarry T., Hafid A.S.* Blockchain scaling using rollups: A comprehensive survey // IEEE Access. 2022.
13. *Raikwar M., Gligoroski D., Kralevska K.* SoK of Used Cryptography in Blockchain // IEEE Access. 2019. V. 7. P. 148550–148575.
14. *Wang L., Shen X., Li J., Shao J., Yang Y.* Cryptographic primitives in blockchains // Journal of Network and Computer Applications. 2019. V. 127. P. 43–58.
15. *Virza M.* On Deploying Succinct Zero-Knowledge Proofs // PhD Thesis. Massachusetts Institute of Technology. 2017. 131 p.
16. *Eberhardt J.* Scalable and Privacy-preserving Off-chain Computations // PhD Thesis. Technical University of Berlin. 2021. 284 p.
17. *Yaga D., Mell P., Roby N., Scarfone K.* Blockchain technology overview // NIST Interagency/Internal Report (NISTIR) – 8202. 2018.
18. *Grassi L., Khovratovich D., Luftenegger R., Rechberger C., Schofnegger M., Walch R.* Reinforced concrete: A fast hash function for verifiable computation // Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022. P. 1323–1335.
19. *Eberhardt J., Tai S.* ZoKrates – scalable privacy-preserving off-chain computations // IEEE Intern. Conf. Blockchain. Halifax, Canada. 2018. P. 1084–1091.
20. https://github.com/Zokrates/ZoKrates – ZoKrates.
21. *Groth J.* On the size of pairing-based non-interactive arguments // EUROCRYPT 2016. Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria. 2016. Part II 35. P. 305–326.

COMPARATIVE EFFICIENCY ANALYSIS OF HASHING ALGORITHMS FOR APPLICATIONS IN ZK-SNARK CIRCUITS IN DISTRIBUTED LEDGERS

© 2024 D. O. Kondyrev[a]

[a] *Novosibirsk State University*
Pirogova st. 2, Novosibirsk, 630090 Russia

The paper presents a comparative efficiency analysis of hashing algorithms in terms of applicability in zk-SNARK based systems. We have considered the hash functions sha256, sha3, blake2, mimc and poseidon, which are most widely used in modern distributed ledgers. To conduct experiments with measuring parameters, an infrastructure based on the ZoKrates toolbox was developed. A series of measurements with different input data was carried out for each algorithm. The number of constraints in the R1CS representation of the algorithm, the length of the proof key and the verification key, the running time of the setup phase of the protocol, and the proof generation time were measured. Based on the obtained results, we determined the boundaries of the practical applicability of algorithms for the problem of proving the knowledge of the preimage of a hash function using zk-SNARK circuits in distributed ledgers, and also identified emerging efficiency problems.

Keywords: distributed ledgers, zero-knowledge proof, zk-SNARK, R1CS, hash functions, algorithm efficiency

REFERENCES

1. *Kondyrev D.O.* Overview of privacy preserving technologies for distributed ledgers // Eurasian Journal of Mathematical and Computer Applications. 2021. V. 9. № 1. P. 55–68.
2. *Ben-Sasson E., Chiesa A., Genkin D., Tromer E., Virza M.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // CRYPTO'2013, LNCS. 2013. V. 8043. P. 90–108.
3. *Ben-Sasson E., Chiesa A., Garman C., Green M., Miers I., Tromer E., Virza M.* Zerocash: Decentralized anonymous payments from bitcoin // IEEE Symp. Security and Privacy, San Jose, USA. 2014. P. 459–474.
4. NIST. FIPS PUB180–2, Secure hash standard. 2002.
5. NIST. FIPS PUB202, SHA-3 standard: permutation-based hash and extendable-output functions. 2015.
6. *Aumasson J.P., Neves S., Wilcox-O'Hearn Z., Winnerlein C.* BLAKE2: simpler, smaller, fast as MD5 // ACNS2013, Proceedings of the 11th International Conference Applied Cryptography and Network Security, Banff, AB, Canada. 2013. V. 7954 of LNCS, P. 119–135.
7. *Grassi L., Khovratovich D., Rechberger C., Roy A., Schofnegger M.* Poseidon: A New Hash Function for Zero-Knowledge Proof Systems // Proceedings of the 30th USENIX Security Symposium. 2021. V. 2021.
8. *Albrecht M., Grassi L., Rechberger C., Roy A., Tiessen T.* MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity // ASIACRYPT 2016. Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam. 2016. Part I. V. 10031 of LNCS. P. 191–219.
9. *Schneier B.* Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons Inc., 1996, 2nd ed.
10. *Sun X., Yu F.R., Zhang P., Sun Z., Xie W., Peng X.* A survey on zero-knowledge proof in blockchain // IEEE network. 2021. V. 35. № 4. P. 198–205.
11. *Konkin A., Zapechnikov S.* Zero knowledge proof and ZK-SNARK for private blockchains // Journal of Computer Virology and Hacking Techniques. 2023. P. 1–7.
12. *Thibault L.T., Sarry T., Hafid A.S.* Blockchain scaling using rollups: A comprehensive survey // IEEE Access. 2022.
13. *Raikwar M., Gligoroski D., Kralevska K.* SoK of Used Cryptography in Blockchain // IEEE Access. 2019. V. 7. P. 148550–148575.
14. *Wang L., Shen X., Li J., Shao J., Yang Y.* Cryptographic primitives in blockchains // Journal of Network and Computer Applications. 2019. V. 127. P. 43–58.
15. *Virza M.* On Deploying Succinct Zero-Knowledge Proofs // PhD Thesis. Massachusetts Institute of Technology. 2017. 131 p.
16. *Eberhardt J.* Scalable and Privacy-preserving Off-chain Computations // PhD Thesis. Technical University of Berlin. 2021. 284 p.
17. *Yaga D., Mell P., Roby N., Scarfone K.* Blockchain technology overview // NIST Interagency/Internal Report (NISTIR) – 8202. 2018.
18. *Grassi L., Khovratovich D., Luftenegger R., Rechberger C., Schofnegger M., Walch R.* Reinforced concrete: A fast hash function for verifiable computation // Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022. P. 1323–1335.
19. *Eberhardt J., Tai S.* ZoKrates – scalable privacy-preserving off-chain computations // IEEE Intern. Conf. Blockchain. Halifax, Canada. 2018. P. 1084–1091.
20. https://github.com/Zokrates/ZoKrates – ZoKrates.
21. *Groth J.* On the size of pairing-based non-interactive arguments // EUROCRYPT 2016. Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria. 2016. Part II 35. P. 305–326.

ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 13–26

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.021

ПОСТРОЕНИЕ ВНУТРЕННЕЙ ДИАГРАММЫ ВОРОНОГО МНОГОУГОЛЬНОЙ ФИГУРЫ МЕТОДОМ ЗАМЕТАНИЯ

© 2024 г. Л. М. Местецкий[a, b, *], Д. А. Коптелов[a, **]

[a] *Московский государственный университет им. М. В. Ломоносова*
119991 Москва, ГСП-1, Ленинские горы, д. 1, Россия
[b] *Национальный исследовательский университет "Высшая школа экономики"*
109028 Москва, Покровский бульвар, д. 11, Россия
**E-mail: mestlm@mail.ru*
***E-mail: dimitar98@list.ru*

Поступила в редакцию: 01.09.2023
После доработки: 15.03.2024
Принята к публикации: 15.03.2024

В статье рассматривается задача построения внутренней диаграммы Вороного многоугольной фигуры – многоугольника с многоугольными дырами. Предлагается метод, основанный на парадигме плоского заметания. Прямое построение только внутренней части диаграммы Вороного позволяет уменьшить объем вычислений и повысить робастность по сравнению с известными решениями. Другим фактором снижения вычислительной сложности является использование свойства попарной инцидентности линейных отрезков, образованных сторонами многоугольной фигуры. Для учета этих особенностей предлагается строить структуру данных статус заметающей прямой в виде упорядоченного множества зон ответственности сайтов. Структура реализуется в виде комбинации сбалансированного дерева и двунаправленного списка. Вычислительные эксперименты иллюстрируют численную надежность и эффективность предложенного метода.

Ключевые слова: диаграмма Вороного

DOI: 10.31857/S0132347424040026, **EDN:** PTJOCY

1. ВВЕДЕНИЕ

Многоугольная фигура (МФ) представляет собой замкнутую область на плоскости, граница которой состоит из непересекающихся простых многоугольников, т. е. это многоугольник с многоугольными дырами. Скелетом или срединной осью такой фигуры является множество точек центров максимальных кругов, содержащихся внутри фигуры. Скелеты находят применение в задачах распознавания формы, в частности, для распознавания рукописного текста по цифровым изображениям. Растровое изображение текста аппроксимируется многоугольными фигурами, и для этого множества многосвязных фигур строится скелет (рис. 1) [1]. В примере на рис. 1 аппроксимация дает 4 фигуры, их границы состоят из 12 многоугольников с 476 вершинами. Скелеты фигур представляют собой геометрические графы с общим числом вершин 690 и ребер 694.

В практических задачах распознавания рукописных документов фигуры и скелеты имеют большие размеры. Например, текст на рукописной странице тотального диктанта [2], отсканированной с разрешением 300 dpi, имеет порядка 300 связных компонент, каждая из которых аппроксимируется многоугольной фигурой (рис. 13). Границы фигур состоят из 1200–1500 многоугольников, имеющих в общей сложности 60–70 тыс. вершин. Скелеты этих фигур имеют до 100 тыс. вершин и ребер. Ввиду большой размерности и разнообразия изображений рукописных документов большое значение для практического использования в архивах оцифрованных рукописей имеет реальное быстродействие и надежность работы алгоритмов.

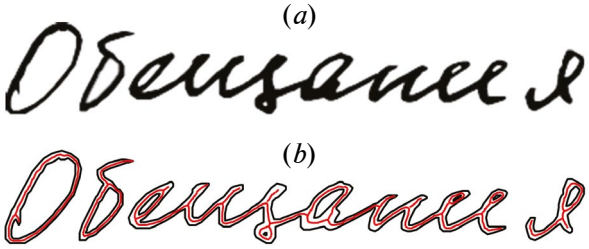

Рис. 1. (*a*) Растровое изображение текста, (*b*) многоугольные фигуры и их скелеты.

Теоретическая оценка вычислительной сложности скелетизации МФ получается на основе сведения задачи к построению обобщенной диаграммы Вороного (ДВ) множества точек и отрезков, образованных из вершин и сторон фигуры. Скелет МФ является подграфом ДВ МФ, выделение этого подграфа имеет сложность $O(n)$, где n – число вершин МФ. Для задачи построения ДВ МФ известна нижняя оценка сложности $\Omega(n \log n)$.

Известны также эффективные алгоритмы построения ДВ множества точек и отрезков, имеющие вычислительную сложность $O(n \log n)$ [3, 4]. Они основываются на разных алгоритмических парадигмах: “разделяй и властвуй” и «плоское заметание”. Эти алгоритмы носят теоретический характер, их практическая реализация на реальных процессорах, имеющих конечную точность, является серьезной проблемой. Несмотря на значительное время, прошедшее после их публикации, полные реализации этих алгоритмов неизвестны.

Одной из проблем, возникающих при построении ДВ МФ большого размера, является так называемая робастность вычислительных алгоритмов [6, 7]. Под этим термином понимается возможность получения корректного решения при наличии ошибок округления, сказывающихся на точности представления данных и результатах операций процессора. Эта проблема возникает при решении многих задач вычислительной геометрии. Она состоит в разрыве между теоретически правильными геометрическими алгоритмами и практическими компьютерными программами, выполняемыми на процессоре с конечной точностью. Фактические вычисления содержат числовые ошибки, эти ошибки вызывают несоответствия в топологии и иногда приводят к аварийному завершению программ.

При построении ДВ числовые ошибки сказываются на геометрических вычислениях предикатов, которые определяют положение точки относительно линии (прямой или окружности), и на вычислениях конструкторов, которые создают новые геометрические объекты, в частности, окружности, касающиеся трех сайтов (точек или отрезков) [7]. В обоих случаях вероятность ошибки возрастает при работе с окружностями очень большого радиуса. При создании такой окружности ищется точка пересечения прямых на основе вычисления матричного определителя. Если эти прямые “почти параллельны”, то значение определителя близко к нулю. В результате окружность имеет очень большой радиус, а центр ее расположен далеко от МФ. Небольшие ошибки вычислений приводят к неверному вычислению предикатов положения точек и отрезков относительно такой окружности. С учетом этого фактора алгоритмы, которые не требуют построения больших окружностей, являются более робастными.

Решение, предлагаемое в данной работе, основано на парадигме плоского заметания. Оно использует особенность задачи построения ДВ МФ, позволяющую избежать построения больших окружностей. Эта особенность состоит в том, что требуется найти лишь внутреннюю часть ДВ точек и отрезков, т. е. часть, лежащую внутри фигуры. А все внутренние вписанные окружности фигуры не превосходят размером саму фигуру. В алгоритмах “разделяй-и-властвуй” избежать построения больших окружностей не удается. Сравнение алгоритмов построения ДВ, основанное на этих двух парадигмах, предпринималось в работе [8]. Сравнение выполнялось по критерию вычислительной эффективности. Сравнение этих алгоритмов с точки зрения робастности не рассматривалось ранее.

Предложенное решение включает новые элементы.

1. Вводится понятие ориентированных односторонних сайтов-сегментов, для которых определена внутренняя сторона МФ. Благодаря этому появляется возможность строить только внутреннюю часть ДВ МФ. В результате не приходится строить большие окружности, а также сокращается объем вычислений, поскольку не нужно строить внешнюю часть ДВ.

2. Структура данных Статус заметающей прямой (ЗП), традиционно присутствующая в алгоритмах заметания [9], строится в виде упорядоченного множества зон влияния сайтов. Такая структура является альтернативой традиционно используемому при построении ДВ понятию волнового фронта или береговой линии.

3. В алгоритме учитывается специфика множества сайтов, образованных границей многоугольной фигуры, а именно инцидентность каждого сайта-точки двум сайтам-сегментам и каждого сайта-сегмента двум сайтам-точкам.

4. Структура данных Статус ЗП строится в виде комбинации сбалансированного дерева и двунаправленного списка, что позволяет выполнять значительную долю операций со Статусом ЗП за константное время.

Предлагаемый алгоритм построения ДВ МФ имеет существенные преимущества по сравнению с известными аналогами. Алгоритм [10]

имеет сложность $O(n\log n)$, однако он строит внутренний скелет лишь для простого n-угольника, т. е. для односвязной МФ. Для многосвязных МФ (многоугольников с многоугольными дырами) – известно решение [11], имеющее сложность $O\big(n(\log n + m)\big)$, где m – количество многоугольников-дыр. Однако для изображений рукописных документов обычно $m = O(n)$ и расходы времени составляют $O\left(n^2\right)$, что при $n \sim 10^5$ является неприемлемым. Для обеспечения практической надежности алгоритмов разрабатываются различные эвристические приемы, ориентированные на особенности конкретных программных решений [12–14]. При этом теоретическая вычислительная эффективность $O(n \log n)$ не достигается.

Известны также практические решения задачи построения ДВ точек и отрезков, которые не достигают теоретических оценок эффективности [12–16].

Предлагаемый алгоритм реализует теоретическую оценку вычислительной сложности $O(n \log n)$ и при этом имеет высокую робастность, что подтверждается вычислительными экспериментами и практическим использованием.

Оставшаяся часть статьи имеет следующую структуру. В разделе 2 дается определение внутренней диаграммы Вороного, в разделах 3–4 описывается концепция предлагаемого алгоритма. Разделы 5–7 содержат описание методов обработки событий в процессе заметания. В разделе 8 описана организация данных и связанные с нею оценки вычислительной сложности алгоритма. Описание вычислительных экспериментов представлено в разделе 9.

2. ВНУТРЕННЯЯ ДИАГРАММА ВОРОНОГО МНОГОУГОЛЬНОЙ ФИГУРЫ

Граница многоугольной фигуры состоит из одного внешнего и нескольких внутренних многоугольников. Все они описываются последовательностями своих вершин так, что внутренность МФ находится слева от границы. Вершины внешнего многоугольника упорядочены против часовой стрелки, а внутренних – по часовой стрелке. Каждый граничный многоугольник разбивается на подмножества, называемые сайтами. Вершины фигуры задают множество сайтов-точек, а стороны фигуры – множество сайтов-сегментов. На сайтах-сегментах задано направление в соответствии с направлением границы фигуры, т. е. внутренность фигуры лежит слева от сайта-сегмента.

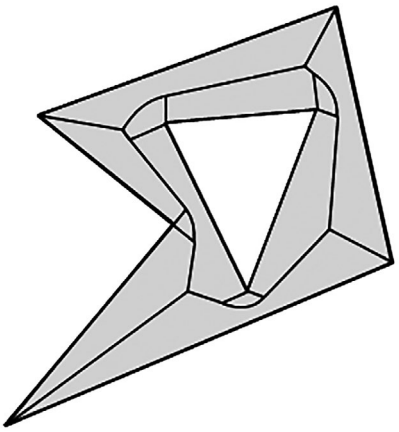

Рис. 2. Внутренняя диаграмма Вороного многоугольной фигуры.

Разбиением Вороного многоугольной фигуры будем называть представление фигуры в виде так называемых локусов. Локусом сайта называется множество точек фигуры, для которых этот сайт является ближайшим. А ближайший сайт для точки фигуры определяется положением ближайшей к ней точки на границе фигуры. Если ближайшая точка границы совпадает с вершиной фигуры, то соответствующий вершине сайт-точка является ближайшим. Если для точки фигуры ближайшей точкой границы является ее ортогональная проекция на сайт-сегмент, то этот сайт-сегмент является ближайшим. Этот ближайший сайт называется порождающим сайтом локуса.

Локус представляет собой замкнутую область. Граница локуса имеет внешнюю часть, совпадающую с границей фигуры и образованную порождающим сайтом. Остальная часть границы локуса – это разделяющая линия с соседними локусами. Граница между парой локусов называется бисектором их порождающих сайтов. Совокупность всех внутренних границ локусов многоугольной фигуры называется внутренней диаграммой Вороного этой фигуры. Внутренняя диаграмма Вороного имеет вид геометрического графа, вершинами которого являются точки фигуры, а ребрами – отрезки прямых и квадратичных парабол (рис. 2).

3. УЧАСТКИ И ЗОНЫ ЗАМЕТАЮЩЕЙ ПРЯМОЙ

Граница МФ может быть представлена в виде конечного множества монотонных ветвей. Каждая ветвь – это ломаная линия, вершины которой упорядочены лексикографически слева направо (рис. 3).

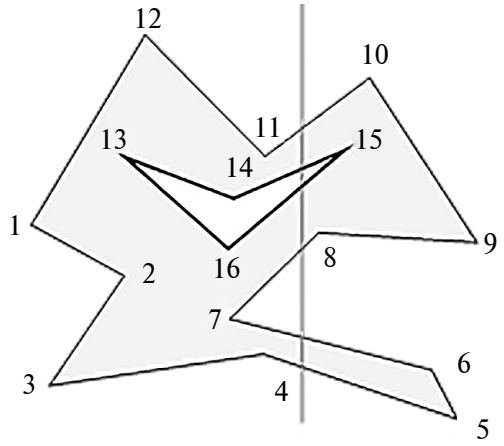

Рис. 3. Монотонные ветви границы фигуры: 1–12–11–10–9, 7–6–5, 7–8–9, 3–4–5, 1–2, 3–2, 13–14–15, 13–16–15.

Заметающая прямая (ЗП) – это вертикальная прямая линия, которая перемещается слева направо параллельно самой себе.

Ветви разбивают заметающую прямую на связные подмножества, так называемые участки. Участки внутри фигур называются внутренними, вне фигур – внешними.

Внутренние участки ЗП в свою очередь разбиваются на так называемые зоны ответственности сайтов, определяемые следующим образом. Для каждой точки внутреннего участка существует максимальный круг, лежащий внутри фигуры в левой полуплоскости относительно заметающей прямой и касающийся ее в этой точке. Поскольку круг максимальный, он также касается изнутри границы фигуры в одной или нескольких точках. Каждая точка касания принадлежит какому-то сайту. Эти сайты и круг называются инцидентными.

Зоной ответственности сайта называется отрезок заметающей прямой, все точки которого имеют касательные круги инцидентные этому сайту. Будем называть этот инцидентный сайт генератором зоны (рис. 4).

На множестве зон задано отношение порядка – снизу вверх. По мере перемещения заметающей прямой это множество зон изменяется: какие-то новые зоны на ней появляются, а какие-то зоны исчезают. Зона в момент порождения имеет длину 0, т. е. это точка на ЗП. По мере продвижения прямой вправо она превращается в отрезок, размер зоны увеличивается. Перед исчезновением размеры зоны уменьшаются до нуля, и она вырождается в точку.

Структура данных, описывающая упорядоченное множество зон на ЗП, носит название “Статус заметающей прямой”, для краткости

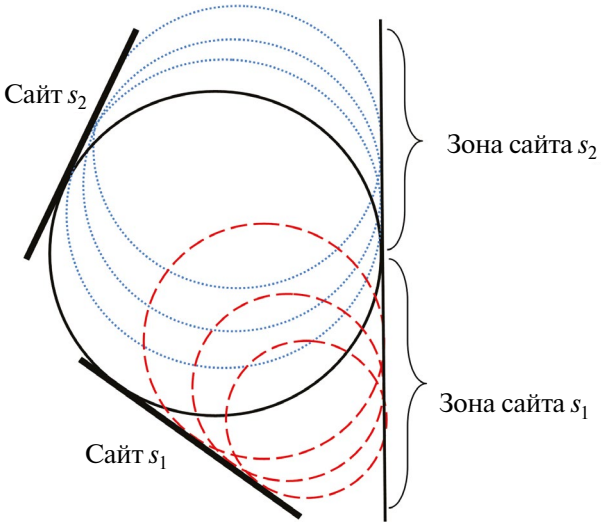

Рис. 4. Зоны сайтов-сегментов s_1, s_2. Касательные круги зоны s_1 – пунктир, зоны s_2 – точечный пунктир, общий круг двух зон – сплошная линия.

Статус, как это принято в алгоритмах плоского заметания [8].

4. ДИАГРАММА ВОРОНОГО И ПРОЦЕСС ЗАМЕТАНИЯ

По мере перемещения ЗП все зоны на ней проходят одинаковый жизненный цикл: порождение, рост и сжатие, расщепление (возможное, но не обязательное), исчезновение. Рост и сжатие представляет собой изменение размеров отрезка зоны. Сначала это монотонное увеличение, а потом уменьшение, тоже монотонное. Расщепление зоны состоит в разделении ее на две части, каждая из которых представляет собой зону с тем же самым генератором.

Между множеством зон и ДВ существует связь, которая позволяет строить ДВ в процесс заметания. Динамически изменяющееся соседство зон в Статусе ЗП полностью определяет структуру ДВ. Поэтому задача состоит в том, чтобы проследить все изменения Статуса и выявить все соседние пары зон в меняющейся структуре Статуса ЗП.

Связь между соседством зон в Статусе и ребрами и вершинами ДВ определяется на основе следующих утверждений.

1. Если два сайта имеют общий касательный вписанный круг, то найдется такое положение заметающей прямой и такая пара соседних зон на ней, у которых генераторами являются эти два сайта.

2. Если две зоны являются соседними на ЗП, то сайты-генераторы этих зон имеют смежные локусы в ДВ и общая граница этих локусов описывает ребро в ДВ.

3. Если три сайта имеют общий касательный вписанный круг, то найдется такое положение заметающей прямой и такая тройка соседних зон на ней, у которых генераторами являются эти три сайта.

Из утверждений 1 и 2 следует, что каждая пара соседних зон в Статусе задает ребро Вороного в ДВ. Отсюда вытекает, что если в процессе перемещения заметающей прямой какие-либо две зоны стали соседними, то бисектор их сайтов-генераторов образует ребро ДВ, поскольку он состоит из точек, равноудаленных от этих сайтов.

В процессе перемещения заметающей прямой соседство зон меняется только в точках событий. Изменение соседства зон в Статусе происходит при порождении новых зон и при исчезновении существующих. Порождение зон происходит, когда ЗП проходит через вершину МФ. Такое положение ЗП называется событием.

Новая зона, включенная в Статус, образует две новые соседние пары с зонами над и под ней. При исчезновении зоны и исключении ее из Статуса две зоны, расположенные над и под нею, образуют новую пару соседних зон. Эти новые пары автоматически порождают ребра ДВ.

Утверждение 3 позволяет вычислить вершины ДВ в процессе заметания. Как только какие-либо три зоны стали соседними в Статусе, нужно проверить, существует ли общий вписанный касательный круг для сайтов-генераторов этих зон.

Таким образом, для нахождения всех вершин и ребер ДВ нужно в процессе заметания отследить все пары и тройки соседних смежных зон в структуре Статус заметающей прямой.

5. ОТКРЫТИЕ ЗОН И ВЫЧИСЛЕНИЕ РЕБЕР ДИАГРАММЫ ВОРОНОГО

Общая идея процесса заметания выглядит следующим образом. Заметающая прямая перемещается слева направо дискретно, занимая положения, соответствующие моментам событий. Каждое событие приводит к изменению Статуса. Событие-вершина приводит к появлению новых зон. При этом множество зон в Статусе сохраняет упорядоченность при всех изменениях.

Изменения в Статусе выражаются в образовании новых пар и новых троек соседних зон. С каждой соседней парой зон связан соответствующий бисектор, разделяющий локусы сайтов-генераторов этих зон. А каждая тройка соседних зон соответствует вершине ДВ, являющейся центром круга, инцидентного трем сайтам-генераторам этих зон. Отслеживая изменения Статуса, можно выявить все соседние пары и тройки зон, и построить соответствующие им ребра и вершины ДВ.

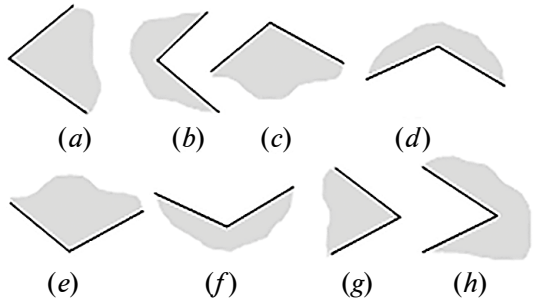

Рис. 5. Типы вершин многоугольной фигуры: левые (a, b), проходные (c, d, e, f), правые (g, h), выпуклые (a, c, e, g), вогнутые (b, d, f, h), нижние (d, e) и верхние (c, f). Внутренность многоугольной фигуры отмечена серым цветом.

Таким образом, в процессе заметания нужно для каждого события внести изменения в Статус, выявить все вновь образовавшиеся соседние пары и тройки зон, и вычислить соответствующие им вершины и ребра ДВ.

Изменения Статуса при наступлении события-вершины определяются типом вершины. Все вершины МФ разделяются на несколько типов в зависимости от пары смежных с ними сторон фигуры. Всего выделяется 8 типов вершин (рис. 5).

Обозначим:

v – сайт-точка, образованный вершиной многоугольника;

s_{pr}, s_{sc} – сайты-сегменты, образованные сторонами многоугольника, стоящими перед и после сайта-точки v в циклическом списке сайтов многоугольника;

$z(s)$ – зона, имеющая в качестве генератора сайт s;

z_* – внешняя зона, не имеющая сайта-генератора.

Событие-вершина приводит к изменению состава зон в Статусе. Эти изменения однозначно определяются в зависимости от типа вершины, с которой связано событие. Варианты этого преобразования Статуса описываются ниже для всех типов вершин, показанных на рис. 5. В строке “Вход” представлен фрагмент Статуса до наступления события, а в строке “Выход” – тот же фрагмент после события.

Левая выпуклая вершина. Сайт-точка v попадает во внешнюю зону z_*. Поскольку направление обхода многоугольника таково, что внутренность МФ лежит слева, в этом случае сайт-сегмент s_{pr} лежит выше сайта-сегмента s_{sc}. В результате обработки события зона z_* расщепляется на две внешние зоны, а между ними порождается новый

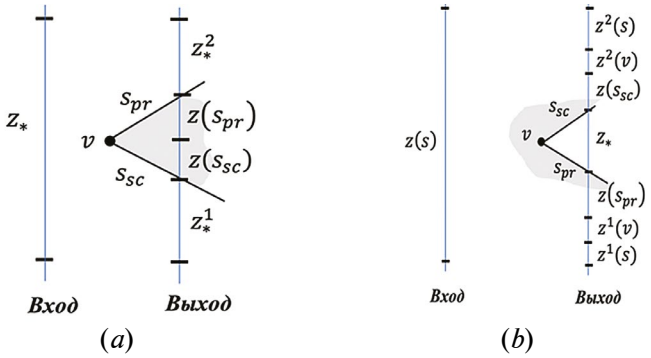

Рис. 6. Изменение Статуса при событиях “Левая вершина”: (*a*) выпуклая, (*b*) вогнутая.

внутренний участок, который состоит из двух новых зон с сайтами-генераторами s_{sc} и s_{pr}. Результат преобразования Статуса выглядит следующим образом (рис. 6*a*):

Вход: $\ldots, z_*, \ldots$

Выход: $\ldots, z_*^1, z(s_{sc}), z(s_{pr}), z_*^2, \ldots$

Образуется одна новая пара соседних зон $\{z(s_{sc}), z(s_{pr})\}$.

Левая вогнутая вершина. Сайт-точка v попадает во внутренний участок в одну из зон ЗП $z(s)$ с сайтом-генератором s. В результате обработки события зона $z(s)$ расщепляется на две зоны $z^1(s)$ и $z^2(s)$, имеющие тот же сайт-генератор s. А между $z^1(s)$ и $z^2(s)$ порождается и вставляется цепочка из пяти новых зон. Одна из них – зона z_* образована внешним участком ЗП, она лежит внутри многоугольника-дыры. Две зоны имеют сайт-генератор v и в двух зонах генераторами являются сайты-сегменты s_{pr} и s_{sc} (рис. 6*b*).

Вход: $\ldots, z(s), \ldots$

Выход: $\ldots, z^1(s), z^1(v), z(s_{pr}), z_*, z(s_{sc}), z^2(v), z^2(s), \ldots$

Здесь образуются 4 новые пары соседних зон: $\{z^1(s), z^1(v)\}$, $\{z^1(v), z(s_{pr})\}$, $\{z(s_{sc}), z^2(v)\}$, $\{z^2(v), z^2(s)\}$.

Проходная выпуклая вершина. В случае проходной вершины ЗП перед пересечением сайта-точки v пересекает инцидентный ей сайт-сегмент, лежащий левее v. Это один из соседних сайтов-сегментов s_{pr} или s_{sc} в зависимости от ориентации многоугольника. Преобразование зависит от расположения многоугольника относительно этой вершины (выше или ниже). В зависимости от этих факторов получаются два варианта преобразования множества зон.

Вершина v выпуклая верхняя (рис. 7*a*):

Вход: $\ldots, z(s_{sc}), \ldots$

Выход: $\ldots, z(s_{sc}), z(s_{pr}), \ldots$

Вершина v выпуклая нижняя (рис. 7*b*):

Вход: $\ldots, z(s_{pr}), \ldots$

Выход: $\ldots, z(s_{pr}), z(s_{sc}), \ldots$

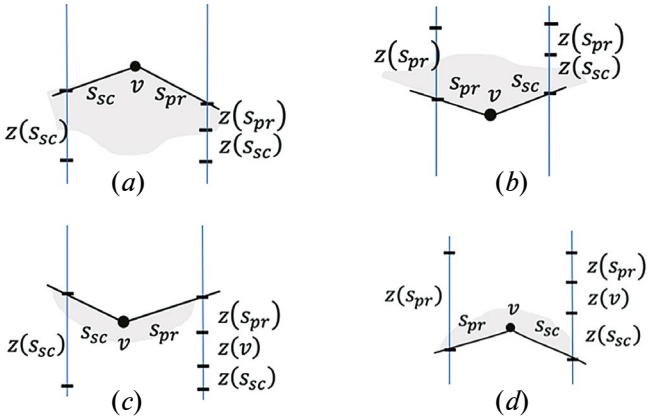

Рис. 7. Изменение Статуса при событиях “Проходная вершина”: (*a*) верхняя выпуклая, (*b*) нижняя выпуклая, (*c*) верхняя вогнутая, (*d*) нижняя вогнутая.

В обоих случаях, представленных на рис. 7*a* и 7*b*, образуется одна новая пара соседних зон $\{z(s_{pr}), z(s_{sc})\}$.

Проходная вогнутая вершина. Аналогично с рассмотренным случаем выпуклой вершины это преобразование зависит от того, находится многоугольник выше или ниже вершины. Получаются два варианта преобразования зон:

Вершина v вогнутая верхняя (рис. 7*c*):

Вход: $\ldots, z(s_{sc}), \ldots$

Выход: $\ldots, z(s_{sc}), z(v), z(s_{pr}), \ldots$

Вершина v вогнутая нижняя (рис. 7*d*):

Вход: $\ldots, z(s_{pr}), \ldots$

Выход: $\ldots, z(s_{sc}), z(v), z(s_{pr}), \ldots$

В обоих случаях на рис. 7*c* и 7*d* образуются две новые пары соседних зон: $\{z(s_{sc}), z(v)\}$, $\{z(v), z(s_{pr})\}$.

Правая выпуклая вершина. Преобразование состоит в удалении двух зон сайтов-сегментов и “сращивании” двух внешних зон (рис. 8*a*).

Вход: $\ldots, z_*^1, z(s_{pr}), z(s_{sc}), z_*^2, \ldots$

Выход: $\ldots, z_*, \ldots$

В этом случае новые соседние пары зон не образуются.

Правая вогнутая вершина. Происходит “замена” внешней зоны на зону с сайтом-генератором v (рис. 8*b*).

Вход: $\ldots, z(s_{sc}), z_*, z(s_{pr}), \ldots$

Выход: $\ldots, z(s_{sc}), z(v), z(s_{pr}), \ldots$

Образуются две новых пары соседних зон: $\{z(s_{sc}), z(v)\}$, $\{z(v), z(s_{pr})\}$.

Анализ событий-вершин показывает, что они приводят к коррекции Статуса ЗП, которая состоит в порождении до пяти новых зон. Эти зоны вставляются в Статус непосредственно одна за другой в известной последовательности. При

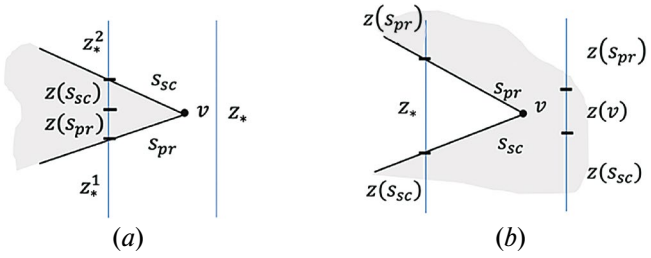

Рис. 8. Изменение Статуса при событиях “Правая вершина”: (*a*) выпуклая, (*b*) вогнутая.

этом образуется до четырех новых пар соседних зон. Каждой паре соседних зон соответствует ребро ДВ. Форма этого ребра определяется сайтами-генераторами пары соседних зон. Если это пара однотипных сайтов, т. е. они оба сайты-точки или оба сайты-сегменты, то ребро ДВ является линейным отрезком. Если же это сайт-точка и сайт-сегмент, то ребро представляет собой отрезок квадратичной параболы. Для того чтобы в явном виде построить это ребро, необходимо кроме двух сайтов-генераторов вычислить его концевые точки – вершины ДВ.

6. ЗАКРЫТИЕ ЗОН И ВЫЧИСЛЕНИЕ ВЕРШИН ДИАГРАММЫ ВОРОНОГО

В геометрическом графе, каковым является ДВ МФ, имеются вершины первой, второй и третьей степени. Вершины первой и второй степени – это точки на границе МФ. Выпуклые вершины МФ имеют в ДВ степень 1, а вогнутые – степень 2. Образование этих вершин осуществляется при наступлении события-вершина.

Вершина ДВ третьей степени – это внутренняя точка МФ, являющаяся граничной для трех локусов. Такая точка является центром вписанного круга, касающегося трех сайтов. Вычисление этих вершин ДВ также выполняется в процессе заметания. Однако их появление связано с другим типом событий – это события-круги.

Новая зона после порождения помещается в Статус ЗП, где она может образовать с другими зонами новые тройки соседних зон. Количество таких новых троек может составить от 0 до 3. Если для сайтов-генераторов тройки соседних зон существует касательный круг, то его центр может оказаться точкой вершиной ДВ. Но для этого нужно, чтобы круг этот был пустым. А достоверно установить его пустоту можно будет лишь в тот момент, когда ЗП при своем движении займет положение касательной к кругу справа, пройдя его полностью. Соответствующее событие, когда ЗП станет касательной к кругу справа, называется событием-кругом. С этим событием связано порождение новой вершины ДВ и удаление из Статуса одной зоны – средней из тройки соседних зон.

Таким образом, для нахождения всех вершин Вороного в процессе заметания необходимо выполнить следующие действия.

При порождении новой зоны в Статусе нужно проверить условие существования общих касательных кругов для троек сайтов-генераторов образовавшихся новых троек соседних зон. Если такой круг существует, то точка центр этого круга является кандидатом на объявление ее вершиной ДВ. Для этого круга нужно запланировать событие-круг. Далее при наступлении этого события точка центр круга объявляется вершиной ДВ.

Событие-круг приводит к исчезновению и исключению из Статуса одной зоны. Удаляется средняя зона из тройки, определяющей круг.

Иллюстрация последовательности событий и изменений, происходящих в Статусе, представлена на рис. 9. Здесь фигура – четырехугольник с сайтами-сегментами s_1, s_2, s_3, s_4. События-вершины связаны с положениями ЗП, обозначенными A, B, C, G.

В положении B для тройки соседних зон $z(s_1)$, $z(s_2)$, $z(s_3)$ образуется вписанный круг сайтов-генераторов s_1, s_2, s_3. Этот круг изображен жирной линией. При его порождении создается событие-круг, привязанное к позиции F. Далее в положении ЗП C в Статус вводится зона $z(s_4)$, в результате чего образуется новая тройка соседних зон $z(s_2)$, $z(s_3)$, $z(s_4)$. Для сайтов s_2, s_3, s_4 существует вписанный круг, для которого создается событие-круг D. Это событие привязано к средней зоне

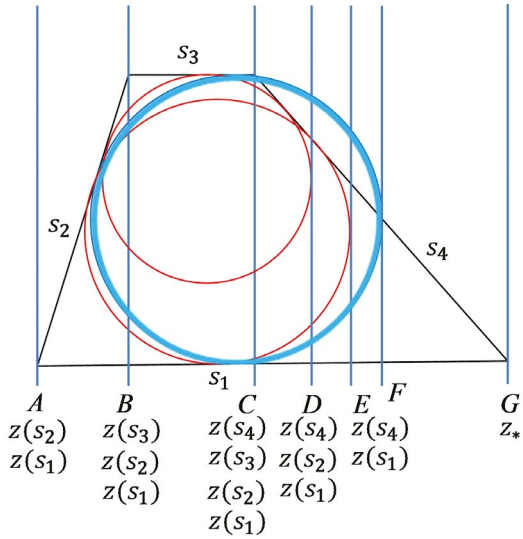

Рис. 9. Заметание четырехугольника, события вершины – A, B, C, G, события-круги D, E, F, изменения Статуса по событиям.

тройки $z(s_3)$. При наступлении события D зона $z(s_3)$ вырождается в точку и удаляется из Статуса.

После ее удаления появляется новая тройка соседних зон $z(s_1)$, $z(s_2)$, $z(s_4)$. Для сайтов-генераторов строится круг, который порождает событие-круг в момент E. Это событие привязано к средней зоне тройки $z(s_2)$. Поскольку эта зона ранее была привязана к кругу, который больше не имеет контакта с ЗП, а лежит левее ее, производится перепланирование событий. Событие F удаляется вместе с привязанным к нему кругом, а для зоны $z(s_2)$ планируется новое событие-круг E.

События-вершины и события-круги полностью описывают процесс получения ребер и вершин ДВ при заметании.

7. ВЫЧИСЛЕНИЕ КАСАТЕЛЬНЫХ КРУГОВ

Задача вычисления координат вершин ДВ возникает при использовании любых алгоритмов. Эта задача имеет чисто геометрическую природу. Нужно построить касательную окружность для трех сайтов, причем касание должно быть в заданной последовательности. Возникает шесть возможных геометрических задач на построение. Это разнообразие определяется составом тройки сайтов (3 точки, 2 точки и отрезок, точка и 2 отрезка, 3 отрезка), а также особыми случаями, когда точка совпадает с концом отрезка (рис. 10).

Поскольку внутренняя вершина ДВ всегда есть точка пересечения пары бисекторов, для ее нахождения могут быть использованы уравнения бисекторов сайтов [17]. Пересечение бисекторов находится на основании решения системы из двух уравнений первой или второй степени. Полученное решение этой системы нуждается в постобработке, поскольку оно может оказаться геометрически некорректным. Кроме того, решение может быть не единственным и необходимо выбрать один вариант из нескольких.

Похожие задачи возникают при определении границ зон. Здесь нужно строить касательные

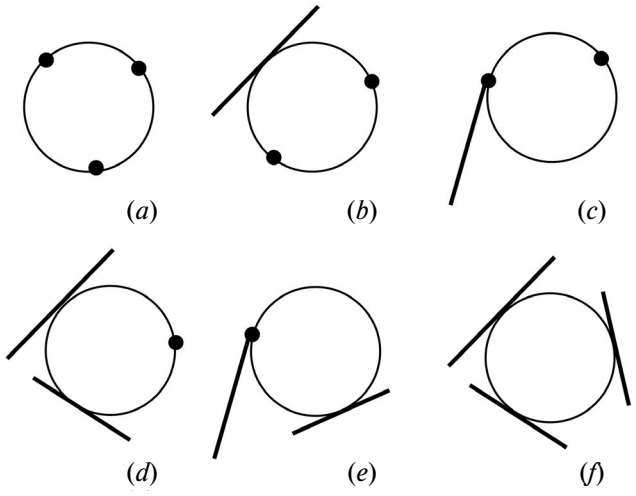

Рис. 10. Геометрические задачи построения касательной окружности для точек и отрезков.

окружности для двух сайтов и заметающей прямой (рис. 11).

Для решения этих задач мы применили методы аналитической геометрии, которые предоставляют ясную геометрическую интерпретацию результатов на любом этапе вычислений. В основе алгоритмов лежит использование классических методов решения геометрических задач на построение, т. е. с помощью циркуля и линейки.

8. ВЫБОР СТРУКТУР ДАННЫХ И ТЕОРЕТИЧЕСКАЯ СЛОЖНОСТЬ

Алгоритмическая парадигма плоского заметания реализуется в виде двух структур данных, описывающих перечень событий и Статус ЗП [9]. В предлагаемом алгоритме перечень событий представлен в виде структуры данных “очередь с приоритетами”, которая эффективно реализуется с помощью АВЛ-дерева. Перечень событий включает события-вершины и события-круги, общее число которых для МФ с n вершинами оценивается как $O(n)$. Для работы с перечнем событий требуются операции вставки, удаления, поиска события и определения самого первого

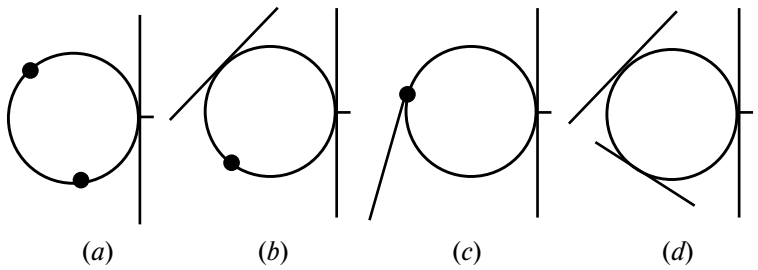

Рис. 11. Вычисление касательного круга для двух сайтов и заметающей прямой: (*a*) –сайты-точки, (*b*, *c*) – сайт-точка и сайт-сегмент, (*d*) – два сайта-сегмента.

события среди оставшихся. Сложность каждой из этих операций в АВЛ-дереве составляет $O(\log n)$.

Статус ЗП должен поддерживать операции, используемые в структуре Словарь: вставка, удаление, поиск, под и над. Как известно, при использовании для реализации словаря АВЛ-дерева, все эти операции также можно выполнять за время $O(\log m)$, где m – число элементов в словаре. В нашем случае элементами являются зоны сайтов. Общее количество зон, образованных в процессе заметания, составляет величину $O(n)$. Если реализовать эту структуру также в виде АВЛ-дерева, то получится эффективное решение, в котором сложность выполнения перечисленных операций составит $O(\log n)$.

Однако особенности рассматриваемой задачи построения ДВ МФ требуют разработки специальной структуры для Статуса ЗП.

Первая особенность состоит в том, что вставка зон в Статус осуществляется во многих случаях целыми пакетами. Как видно из проведенного анализа событий-вершин, вставляется цепочка от 1 до 5 зон одновременно. Это значит, что нерационально использование механизма АВЛ-дерева, где каждая вставка занимает логарифмическое время. Нужно найти способ, при котором эта вставка сможет за один заход вставить весь пакет зон.

Вторая особенность связана с тем, что все зоны в одном пакете в начальный момент при порождении имеют нулевые размеры, т. е. это точки на ЗП. При этом точки эти имеют одну и ту же ординату, т. е. их положение на ЗП просто совпадает. Для того чтобы вводить зоны последовательно по одной с помощью АВЛ-дерева нужно иметь какой-то способ сравнения этих зон по критерию выше-ниже. Но стандартная операция вставки элементов в АВЛ-дерево не предоставляет таких средств.

Обойти проблемы, связанные с этими особенностями предлагается с помощью комбинированной структуры данных, включающей АВЛ-дерево и двунаправленный список.

Двунаправленный список представляет собой упорядоченное снизу вверх множество зон. При этом внешние участки ЗП представлены как отдельные зоны в том же формате, что и зоны внутренних участков. Главное достоинство представления Статуса таким образом состоит в том, что большая часть операций вставки зон при наступлении событий-вершин, описанных в разделе “Открытие зон”, может быть выполнена за константное время. События, связанные с прохождением ЗП через проходные и правые вершины, состоят в порождении нескольких зон и вставке их в заданном порядке в Статус. При этом место вставки известно, оно определяется зоной сайта-сегмента, инцидентного вершине и лежащего левее ее в монотонной ветви. Таким образом, такая вставка зон в количестве до 5 не зависит от числа зон в Статусе и выполняется за константное время.

Единственное событие, обработка которого не укладывается в этот простой механизм – это “левая вершина”. Для вставки левой вершины место в Статусе заранее неизвестно, поэтому требуется локализация ее на ЗП. Это значит, что нужно найти зону, в которую попадает левая вершина. Двунаправленный список дает возможность это сделать только путем последовательного просмотра зон. Такой просмотр имеет сложность $O(n)$ в худшем случае. Поскольку число левых вершин в МФ имеет порядок $O(n)$, получается, что операции по вставке зон таких вершин в Статус имеют сложность $O(n^2)$, что является неприемлемым.

Предлагаемое решение состоит в том, чтобы дополнить двунаправленный список зон структурой АВЛ-дерева, описывающей упорядоченное снизу вверх множество ветвей, пересекающих ЗП. Эта структура позволяет определить для точки на ЗП пару ветвей, лежащих выше и ниже точки. Такая пара ветвей задает участок на ЗП. Каждая ветвь содержит ссылки на зоны, прилегающие к ней при текущем положении ЗП. Эта конструкция обеспечивает локализацию точки в множестве участков за время $O(\log n)$, а локализацию всех левых вершин – $O(n \log n)$.

Если найденный участок оказался внешним, то никаких дополнительных действий не требуется и вставка новой группы зон выполняется по правилу, представленному на рис. 6*а*. Если же найденный участок является внутренним, то для дальнейшего поиска зоны предлагается алгоритм, который можно назвать “вилкой”.

Предположим, что найденный внутренний участок содержит k зон $z_1, \ldots, z_k$. Пара (z_{low}, z_{up}) задает нижнюю и верхнюю зоны интервала поиска. Вначале $z_{low} = z_1$, $z_{up} = z_k$. Обозначим также $z.above$ и $z.under$ зоны в Статусе, расположенные выше и ниже зоны z.

Поиск зоны, содержащей левую вершину v, осуществляется итеративно, каждая итерация включает две проверки:

• если вершина v локализуется в зоне z_{low}, возвращается z_{low}, иначе интервал поиска сжимается снизу, $z_{low} = z_{low}.above$;

• если вершина v локализуется в зоне z_{up}, возвращается z_{up}, иначе интервал поиска сжимается сверху, $z_{up} = z_{up}.under$.

Этот итерационный процесс гарантированно закончится успешно, поскольку вершина v обязательно попадает в одну из зон на участке. Вычислительная сложность при этом составит $O(k)$. При этом максимальное число проверок k потребуется в том случае, когда вершина v лежит в медианной зоне z_t, $t = \left\lceil \frac{k}{2} \right\rceil$, занимающей серединное положение в участке.

Покажем, что локализация всех левых вершин в зонах ЗП имеет сложность $O(n\log n)$. Число зон, порождаемых в процессе заметания, составляет $O(n)$. Без ограничения общности будем считать, что зон ровно n. Обозначим m – число левых вершин МФ, $m = O(n)$. Оценим максимальное число проверок за все время работы. При наступлении события “левая вершина” максимальное число проверок потребуется при выполнении двух условий:

1. Вершина локализуется в участке с наибольшим числом зон.

2. При локализации на участке вершина попадает в медианную зону, т. е. совершается k проверок, где k – число зон на участке.

Таким образом, если всегда выполняется максимальное число проверок, то для первой левой вершины в участке из n зон совершается n проверок. После образования новых зон, связанных с этой вершиной, образуются два новых участка, которые будут содержать по $\frac{n}{2}$ зон. Две последующие локализации с максимальным числом проверок в участках длины $\frac{n}{2}$ потребуют по $\frac{n}{2}$ проверок каждый. После вставки соответствующих зон в Статус образуются четыре участка по $\frac{n}{4}$ зон, и т. д. На каждом уровне j образуется 2^{j-1} участков, включающих по $\frac{n}{2^{j-1}}$ зон каждый. На этом уровне происходит 2^{j-1} локализаций левых вершин с суммарным числом проверок n. За j уровней обрабатывается $1 + 2 + 4 + \ldots + 2^{j-1} = 2^j - 1$ событий типа “левая вершина”. При $j = \log(m+1)$ получаем, что все левые вершины будут обработаны за j уровней. А так как на каждом уровне всего n проверок, получаем общее число проверок $O(n\log m)$. Так как $m = O(n)$, получаем итоговую сложность $O(n\log n)$.

Таким образом, предложенный алгоритм построения ДВ многоугольной фигуры с использованием двухступенчатого иерархического Статуса имеет асимптотическую сложность $O(n\log n)$, где n – суммарное число вершин в МФ.

9. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

Для практической проверки алгоритма на корректность, эффективность и надежность используется набор изображений различной сложности. Сложность определяется числом многоугольников и вершин в МФ, полученных при аппроксимации границ изображения.

Проверка эффективности проводилась путем сравнения времени работы предложенного алгоритма с реализацией алгоритма Форчуна из библиотеки C++ Boost [18]. Данная реализация включает построение обобщенной ДВ для произвольного набора точек и отрезков на плоскости. То есть алгоритм в библиотеке Boost никак не учитывает взаимное расположение точек и сегментов в виде многоугольников, а также строит разбиение всей плоскости, а не только внутренней части МФ.

После построения ДВ всей плоскости алгоритмом Форчуна необходимо провести постобработку, состоящую в удалении внешних относительно МФ ребер ДВ. Однако в экспериментах время, затрачиваемое на постобработку в алгоритме Форчуна, не учитывается. Таким образом, цифры затрат времени для библиотеки Boost представляют собой нижние оценки для построения ДВ МФ, в них не включено время постобработки. Результаты временных замеров приведены в табл. 1. Примеры некоторых изображений, использованных в экспериментах, (Лошадь, Нейрон, Дерево, План) представлены на рис. 12.

Сравнение двух алгоритмов проводилось в единой среде при одинаковых условиях, время усреднялось по 10 замерам. Тем не менее абсолютное время работы программы в значительной степени есть характеристика реализации, а не самого алгоритма. Более показательным в данном эксперименте является относительное изменение времени работы на изображениях разного размера. На картинках небольшого размера (до 5000 вершин МФ) алгоритм из Boost проигрывает нашему алгоритму в 2–4 раза, на больших же картинках, с более чем 100 тысячами вершин, время работы алгоритмов отличается примерно в 50 раз.

Результаты еще одного эксперимента приведены в табл. 2. Здесь представлены результаты

Таблица 1. Вычисление ДВ тестовых примеров многоугольных фигур

Картинка	Число вершин	Число контуров	Время предложенного алгоритма, мс (T_1)	Время алгоритма из библиотеки Boost, мс (T_2)	Отношение расходов времени (T_2/T_1)
Лошадь	374	1	2.6	3.5	1.35
Нейрон	3449	7	25.5	60	2.35
Дерево	175375	6373	1894	84537	44.63
План	185115	5395	1659	94908	57.20

Рис. 12. Тестовые примеры многоугольных фигур: лошадь (1 многоугольник, 374 вершины), нейрон (7, 3449), дерево (6373, 175375), план (5395, 185115).

Таблица 2. Вычисление ДВ фигур изображений рукописного текста

Документ	Количество контуров	Количество вершин в контурах	Количество вершин в ДВ	Время, мс
1	917	27991	54824	253
2	1710	85938	104558	914
3	1332	100040	108427	1106
4	1696	112746	131366	1255
5	1470	106961	115050	1175

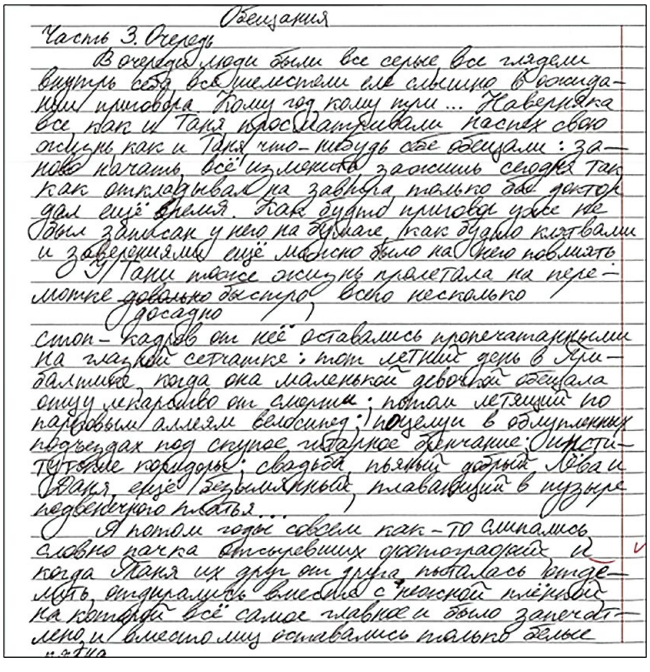

Рис. 13. Страница тотального диктанта – пример множества многоугольных фигур, аппроксимирующих изображение рукописного текста.

работы алгоритма с пятью цифровыми изображениями рукописного текста. Пример такого изображения представлен на рис. 13. Исходные бинарные растровые изображения аппроксимируются МФ с помощью алгоритмов [1]. Размеры получаемых фигур приведены в таблице. Затраты времени на построение ДВ выражены в миллисекундах.

Этот эксперимент показал применимость алгоритма к решению практических задач обработки больших изображений рукописных текстов размера 4000×2500 пикселов. Полученные ДВ используются для получения непрерывного скелета и на его основе построения штриховой сегментации рукописи. Штриховая сегментация находит применение в алгоритмах распознавания рукописного текста при решении задач расшифровки текста, навигации в больших рукописных архивах, идентификации авторов.

10. ЗАКЛЮЧЕНИЕ

В статье представлен алгоритм построения внутренней ДВ МФ, ориентированный на практическую программную реализацию и приложения к задачам большого размера, в частности, на работу с большими изображениями рукописных документов. За счет того, что ДВ строится только для внутренней части МФ, достигается несколько полезных свойств алгоритма, способствующих повышению вычислительной эффективности и числовой надежности.

Основная концепция алгоритма основана на парадигме плоского заметания, использованной в алгоритме Форчуна. Предложенный алгоритм реализует редукцию задачи к построению ДВ линейных отрезков, но при этом включает новые элементы, ориентированные на использование свойств отрезков, составленных из многоугольников. За счет этого достигается уменьшение вычислительной сложности, поскольку значительная часть операций, которые в классическом алгоритме Форчуна имеют логарифмическую сложность, реализуется за константное время. Кроме того, сокращается объем вычислений по сравнению с известными алгоритмами, которые строят внутреннюю и внешнюю части ДВ МФ.

Предложенный алгоритм обладает высокой численной надежностью, поскольку внутренняя ДВ МФ не требует вычислений вписанных кругов большого размера, а также нахождения вершин ДВ, расположенных на очень большом расстоянии от фигуры.

Предложенный алгоритм реализован в полном объеме и используется при решении практических задач анализа и распознавания цифровых изображений рукописных текстов.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке Российского научного фонда, грант № 22-68-00066, https://rscf.ru/project/22-68-00066/.

СПИСОК ЛИТЕРАТУРЫ

1. *Местецкий Л.М.* Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры. М.: Физматлит, 2009.
2. *Отургашева Н.В.* Послание URBI ET ORBI: тотальный диктант как культурный проект // Вестник Томского государственного университета. 2019. № 35. С. 105–113. https://doi.org/10.17223/22220836/35/10
3. *Fortune S.* A sweepline algorithm for Voronoi diagrams. Algorithmica. 2 (1987). P. 153–174.
4. *Yap C.K.* An O(n log n) algorithm for the Voronoi diagram of the set of simple curve segments. Discrete Comput. Geom. 2 (1987). P. 365–393.
5. *Местецкий Л.М.* Скелетизация многосвязной многоугольной фигуры на основе дерева смежности ее границы // Сиб. журн. вычисл. матем. 2006. Т. 9. № 3. С. 299–314.
6. *Fortune S.* (1996). Robustness issues in geometric algorithms. In: Lin, M.C., Manocha, D. (eds) Applied Computational Geometry Towards Geometric Engineering. WACG 1996. Lecture Notes in Computer Science. V. 1148. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0014476
7. *Shewchuk J.R.* (2013). Lecture Notes on Geometric Robustness. University of California at Berkeley, Berkeley, CA 94720.
8. *Лагно Д., Соболев А.* Модифицированные алгоритмы Форчуна и Ли скелетизации многоугольной фигуры. Графикон-2001, Н. Новгород.
9. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение: Пер. с англ. – М.: Мир, 1989. – 478 с.
10. *Lee D.T.* Medial axes transform of planar shape // IEEE Trans. Patt. Anal. Mach. Intell. PAMI-4. 1982. P. 363–369.
11. *Srinivasan V., Nackman L.R.* Voronoi diagram for multiply-connected polygonal domains I: Algorithm // In IBM Journal of Research and Development, V. 31. No. 3. P. 361–372. May 1987. https://doi.org/10.1147/rd.313.0361.
12. *Held M.* Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments // Computational Geometry. 2001. V. 18. P. 95–123.

13. *Sugihara K., Iri M., Inagaki H. et al.* Topology-Oriented Implementation – An Approach to Robust Geometric Algorithms. Algorithmica 27, 5–20 (2000). https://doi.org/10.1007/s004530010002
14. *Karavelas M.I.* A robust and efficient implementation for the segment Voronoi diagram. In Proc. Internat. Symp. on Voronoi diagrams in Science and Engineering (VD2004), 2004. P. 51–62.
15. *Imai T.* A topology oriented algorithm for the voronoi diagram of polygons. cccg1996, 1996. P. 107–112.
16. *Bae, S.W.* (2015). An Almost Optimal Algorithm for Voronoi Diagrams of Non-disjoint Line Segments. In: Rahman M.S., Tomita E. (eds) WALCOM: Algorithms and Computation. WALCOM 2015. Lecture Notes in Computer Science. V. 8973. Springer, Cham. P. 34–43.
17. *Marsden D.* Exact Generalized Voronoi Diagram Computation using a Sweepline Algorithm (2020). All Graduate Theses and Dissertations. 7947. https://digitalcommons.usu.edu/etd/7947
18. https://www.boost.org/doc/libs/1_59_0/libs/polygon/doc/voronoi_main.htm

CONSTRUCTING THE INTERNAL VORONOI DIAGRAM OF A POLYGONAL FIGURE USING THE SWEEP METHOD

© 2024 L. M. Mestetskiy[a, b], D. A. Koptelov[a]

[a] *Lomonosov Moscow State University*
Leninskie Gory 1, GSP-1, Moscow, 119991 Russia
[b] *National Research University Higher School of Economics*
Pokrovsky Boulevard 11, Moscow, 109028 Russia

The article considers the problem of constructing the internal Voronoi diagram of a polygonal figure – a polygon with polygonal holes. A method based on the flat sweeping paradigm is proposed. Direct construction of only the internal part of the Voronoi diagram allows us to reduce the amount of calculations and increase robustness compared to known solutions. Another factor in reducing computational complexity is the use of the property of pairwise incidence of linear segments formed by the sides of a polygonal figure. To take these features into account, it is proposed to build the data structure Status of the sweeping line in the form of an ordered set of sites' areas of responsibility. The structure is implemented as a combination of a balanced tree and a bidirectional list. Computational experiments illustrate the numerical reliability and efficiency of the proposed method.

Keywords: Voronoi diagram

REFERENSES

1. *Mestetskiy L.M.* Continuous morphology of binary images: figures, skeletons, circulars. Moscow, Fizmatlit, 2009 (in Russian).
2. *Oturgasheva N.V.* URBI ET ORBI message: total dictation. as a cultural project. Bulletin of Tomsk State University. 2019. No. 35. P. 105–113 (in Russian). https://doi.org/10.17223/22220836/35/10
3. *Fortune S.* A sweepline algorithm for Voronoi diagrams. Algorithmica. 2 (1987). P. 153–174.
4. *Yap C.K.* An O(n log n) algorithm for the Voronoi diagram of the set of simple curve segments. Discrete Comput. Geom. 2 (1987). P. 365–393.
5. *Mestetskiy L.M.* Skeletonization of a multiconnected polygonal figure based on the adjacency tree of its boundary // Sibirsk. magazine Comput. Mat. 2006. V. 9. No. 3. P. 299–314 (in Russian).
6. *Fortune, S.* (1996). Robustness issues in geometric algorithms. In: Lin, M.C., Manocha, D. (eds) Applied Computational Geometry Towards Geometric Engineering. WACG 1996. Lecture Notes in Computer Science. V. 1148. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0014476
7. *Shewchuk J.R.* (2013). Lecture Notes on Geometric Robustness. University of California at Berkeley, Berkeley, CA 94720.
8. *Lagno D., Sobolev A.* Modified Fortune and Lee algorithms for skeletonization of a polygonal figure. Graphicon-2001, Nizhny Novgorod (in Russian).
9. *Preparata F., Sheimos M.* Computational geometry: Introduction: Transl. from English. – M.: Mir, 1989. – 478 p.
10. *Lee D.T.* Medial axes transform of planar shape // IEEE Trans. Patt. Anal. Mach. Intell. PAMI-4. 1982. P. 363–369.
11. *Srinivasan V., Nackman L.R.* Voronoi diagram for multiply-connected polygonal domains I: Algorithm // In IBM Journal of Research and Development, V. 31. No. 3. P. 361–372. May 1987. https://doi.org/10.1147/rd.313.0361.
12. *Held M.* Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments // Computational Geometry. 2001. V. 18. P. 95–123.
13. *Sugihara K., Iri M., Inagaki H. et al.* Topology-Oriented Implementation – An Approach to Robust Geometric

Algorithms. Algorithmica 27, 5–20 (2000). https://doi.org/10.1007/s004530010002

14. *Karavelas M.I.* A robust and efficient implementation for the segment Voronoi diagram. In Proc. Internat. Symp. on Voronoi diagrams in Science and Engineering (VD2004), 2004. P. 51–62.
15. *Imai T.* A topology oriented algorithm for the voronoi diagram of polygons. cccg1996, 1996. P. 107–112.
16. *Bae, S.W.* (2015). An Almost Optimal Algorithm for Voronoi Diagrams of Non-disjoint Line Segments. In: Rahman M.S., Tomita E. (eds) WALCOM: Algorithms and Computation. WALCOM 2015. Lecture Notes in Computer Science. V. 8973. Springer, Cham. P. 34–43.
17. *Marsden D.* Exact Generalized Voronoi Diagram Computation using a Sweepline Algorithm (2020). All Graduate Theses and Dissertations. 7947. https://digitalcommons.usu.edu/etd/7947
18. https://www.boost.org/doc/libs/1_59_0/libs/polygon/doc/voronoi_main.htm

ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 27–40

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.89

АНАЛИТИЧЕСКИЙ ОБЗОР КОНФИДЕНЦИАЛЬНОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА: МЕТОДЫ И АЛГОРИТМЫ РЕАЛИЗАЦИИ В ОБЛАЧНЫХ ВЫЧИСЛЕНИЯХ

© 2024 г. Е. М. Ширяев[a,*], А. С. Назаров[a,], Н. Н. Кучеров[a,***], М. Г. Бабенко[a,b,****]**

[a] *Северо-Кавказский федеральный университет*
355017 Ставрополь, ул. Пушкина, д. 1, Россия
[b] *Институт системного программирования РАН им. В. П. Иванникова*
109004 Москва, ул. А. Солженицына, д. 25, Россия
**E-mail: eshiriaev@ncfu.ru*
***E-mail: anazarov@ncfu.ru*
****E-mail: nkucherov@ncfu.ru*
*****E-mail: mgbabenko@ncfu.ru*

Поступила в редакцию 21.02.2024
После доработки 18.03.2024
Принята к публикации 18.03.2024

Технологии искусственного интеллекта и облачных систем в последнее время активно развиваются и внедряются. В связи с этим обострился вопрос их совместного использования, актуальный уже несколько лет. Проблема сохранения конфиденциальности данных в облачных вычислениях приобрела статус критической задолго до возникновения необходимости их совместного использования с искусственным интеллектом, который сделал ее еще более сложной. В данной статье представлен обзор как самих методов искусственного интеллекта и облачных вычислений, так и методов обеспечения конфиденциальности данных. В обзоре рассмотрены методы, использующие дифференциальную конфиденциальность; схемы разделения секрета; гомоморфное шифрование; гибридные методы. Проведенное исследование показало, что каждый рассмотренный метод имеет свои плюсы и минусы, обозначенные в работе, однако универсальное решение отсутствует. Было установлено, что теоретические модели гибридных методов, основанных на схемах разделения секрета и полностью гомоморфном шифровании, позволяют существенно повысить конфиденциальность обработки данных с использованием искусственного интеллекта.

Ключевые слова: облачные вычисления, искусственный интеллект, нейронная сеть, схемы разделения секрета, гомоморфное шифрование, система остаточных классов

DOI: 10.31857/S0132347424040036, **EDN:** PTEUNV

1. ВВЕДЕНИЕ

Технологии искусственного интеллекта (ИИ) получают все большее распространение в производственной и повседневной жизни общества. Рост популярности и быстрое развитие технологий приводят к усложнению задач, решаемых с помощью ИИ, и, как следствие, к тому, что обработка информации на стандартном пользовательском устройстве выполняется критически неэффективно, что неприемлемо для конечного потребителя. В данном случае выходом является применение облачных технологий (ОТ), которые в свою очередь уже обрели широкую популярность и получили развитие своей методологии. Стоит также отметить, что ИИ и ОТ вызывают большой интерес и в научном сообществе. Например, такие проекты как [1, 2], обрели популярность в повседневной сфере деятельности при этом являются научными проектами в области ИИ. При совместном использовании методов ИИ и ОТ появляется ряд как стандартных, так и специфических проблем, связанных с надежностью, безопасностью и конфиденциальностью. Учитывая, что из всего функционала, реализуемого ОТ, ИИ использует в основном облачные вычисления (ОВ), для ИИ с использованием ОТ имеют место все проблемы, характерные для ОВ. Угрозы безопасности ОВ в целом можно разделить на две категории: внешние угрозы и внутренние. К внешним угрозам относятся различные атаки

злоумышленников в целях кражи информации (например взлом) или повреждения информации (например DDOS-атаки) [3]. Внутренние угрозы в целом представляют собой совокупность всех возможных способов компрометации системы безопасности изнутри. Для борьбы с последними используют, например, схемы разделения секрета [4]. Однако наиболее высокий уровень безопасности достигается при использовании методов, которые позволяют обрабатывать данные в зашифрованном виде. В таком случае вероятность компрометации системы сокращается до минимума. Возможным решением проблемы конфиденциальной обработки данных является гомоморфное шифрование. В данной работе представлено исследование современных используемых на практике методов и алгоритмов обеспечения конфиденциальности ИИ и предиктивное исследование методов, которые возможно будут применяться в будущем.

Работа состоит из 4 разделов. В разделе 2 рассмотрены основные аспекты, связанные с ИИ и облачными технологиями. В разделе 3 рассмотрены подходы к построению сохраняющего конфиденциальность ИИ. В разделе 4 представлены результаты проведенного аналитического обзора.

2. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И ОБЛАЧНЫЕ ТЕХНОЛОГИИ

2.1. Искусственный интеллект

Искусственный интеллект (ИИ) изначально развивался в рамках интеллектуальных систем и до сих пор считается одной из их составляющих, а именно способностью выполнять творческие функции. Изначально необходимость в интеллектуальных системах была обусловлена автоматизацией принятия решений. То есть от системы ожидалась определенная реакция на определенные события. С развитием вычислительной техники, методов и алгоритмов, а также способов разработки систем и приложений данные реакции усложнялись и становились более гибкими.

Исторически первыми методами ИИ были методы машинного обучения (МО). МО представляет собой класс методов ИИ, задачей которых является не прямое решение задачи, а нахождение его за счет обучения на основе анализа множества решений сходных задач [5]. Постановку задачи МО можно определить следующим образом. Существует некая неизвестная зависимость между двумя множествами. Известны только прецеденты, т. е. пары из этих двух множеств, которые называются обучающей выборкой. На основе этих данных ставится задача восстановления зависимости, т. е. построения алгоритма, который с заданной точностью создаст новую пару. По сути, перед МО ставится задача аппроксимации функции, но не обязательно другой функцией, а неким алгоритмом.

Методы ИИ также можно разделить по способу обучения. Так, например, существуют модели обучения с подкреплением [6], среди таких моделей можно выделить генетические алгоритмы. Генетические алгоритмы представляют собой эвристические алгоритмы поиска, использующиеся для решения задач оптимизации и моделирования, путем случайного подбора, комбинирования и вариации исходных параметров, которые основаны на методах, подобных естественному отбору в природе [7]. Также существует обучение без учителя, подобными методами решается, например, задача кластеризации [8]. Однако наибольшую группу методов составляют методы, использующие обучение с учителем, где для множества прецедентов (известных пар входных и выходных данных) необходимо построить алгоритм, возвращающий требуемое решение [9].

Искусственные нейронные сети, или просто нейронные сети (НС), представляют собой математическую модель, которая построена по принципу функционирования сетей нервных клеток живого организма [10]. Развитие вычислительной техники позволило создавать модели НС большой сложности. В этом контексте можно выделить несколько событий, которые позволили расширить применение ИИ, это появление различных аппаратных и графических ускорителей [11–13], а также вентильных матриц [14–17] для ИИ, которые позволяют решать более широкий спектр задач, недоступный ранее. Работа таких моделей НС осуществляется за счет применения, так называемого глубокого обучения (ГО). По сути, ГО это процесс обучения многослойных НС. Теоретически 2–3-слойных НС достаточно для решения широкого круга задач, однако, для решения сложных задач зачастую используется ГО, которое показывает хорошие результаты [18]. К ГО относят такие методы как ограниченная машина Больцмана. Также на основе ГО строятся сверточные нейронные сети, которые используют на различных слоях различные формы сверток [19]. Применяются они зачастую для распознавания образов [20]. Одной из наиболее популярных современных технологий ИИ является

технология GPT. Даже в начале развития GPT модели требовали больших обучающих выборок, которые занимают десятки гигабайт на стадии предварительного обучения, а учитывая тот факт, что количество пользователей моделей с GTP превышает 100 млн, объемы потребляемых ресурсов превышают ресурсы доступные одному устройству, если рассматривать среднестатистический офисный компьютер. Для эффективной работы таких технологий требуется применение распределенных вычислительных систем.

2.2. Облачные вычисления

Рассмотрим более подробно облачные технологии (ОТ). По сути, ОТ и ОВ являются синонимами, так как любая обработка данных в облаке предполагает какие-либо вычисления. Аналогично ситуация обстоит с облачными хранилищами, так как при обработке информации (например, поиске) также производятся определенные вычисления.

ОВ являются развитием модели распределенных вычислений [21] за некоторыми исключениями. Распределенные вычисления предполагают наличие параллелизма в вычислениях, а именно объединение вычислительных ресурсов в параллельную вычислительную систему. Реализация такой системы возможна и на одном физическом устройстве, например серверной стойке или суперкомпьютере [22]. Переходной точкой между распределенными вычислениями и облачными вычислениями можно считать грид вычисления [23]. Главное же отличие ОВ заключается в концептуализации [24]. В отличие от распределенных и грид вычислений, которые являются прежде всего совокупностью средств и способов решения вычислительно сложных задач, ОВ прежде всего являются сервисом (услугой) по предоставлению возможностей для реализации эластичных вычислений. Принято классифицировать ОВ по видам моделей, в рамках которых предоставляется данная услуга:

• Software as a Service (SaaS) – эта модель подразумевает использование облачной инфраструктуры для реализации ОВ посредством предоставления пользователю пакета прикладного программного обеспечения. Контроль и управление инфраструктурой производится исключительно провайдером услуг [25];

• Platform as a Service (PaaS) – в таком случае пользователю представляется инфраструктура для размещения различного базового программного обеспечения. Обычно это инструментальные средства для создания программного обеспечения и, например, системы управления базами данных. Также провайдером могут предоставляться различные среды для работы с языками программирования [26];

• Infrastructure as a Service (IaaS) – в данном варианте поставщик предоставляет пользователю наиболее полные права по пользованию облаком. Здесь пользователю предлагается базовая инфраструктура, в рамках которой он самостоятельно организует процессы управления вычислительными ресурсами, а также построения сети и хранения данных. Также пользователь самостоятельно контролирует операционные системы, которые разворачиваются в облаке. Еще одно отличие от предыдущих категорий – пользователю предоставляется ограниченный контроль над сетевыми сервисами выделенного ему облачного пространства [27, 28].

На основе данных трех моделей выделяют и различные гибридные, такие как Data Base as a Service [29], Monitoring as a Service [30] и т. п., однако гибридные виды, по сути, разделяются с точки зрения потребностей клиента и скорее направлены на узкую специализацию применения того или иного вида ОВ. На основе вышеизложенной информации можно выделить преимущества, которые дают ОВ для ИИ. Как уже было сказано, методы ИИ являются достаточно вычислительно сложными. При проведении исследований и разработке приложений и сервисов, сопряженных с ИИ, исследователь/разработчик сталкивается с трудностями, связанными с ограниченностью вычислительных ресурсов. В случае исследователя, в теории, можно воспользоваться распределенной вычислительной системой своего учреждения, однако не в каждом учреждении имеется собственная мощная вычислительная система, либо к ней затруднен доступ. ОВ позволяют исследователю/разработчику арендовать вычислительные ресурсы у поставщика.

2.3. Облачные вычисления и искусственный интеллект

Рассмотрим применимость различных моделей ОВ для ИИ. В целом ИИ может быть развернут в рамках любой из трех моделей предоставления услуги. Однако есть несколько нюансов, в случае SaaS поставщик помимо вычислительных ресурсов поставляет и ИИ, в данном случае ИИ также является услугой. Примером такой услуги является MLaaS [31]. В случае PaaS поставщик предоставляет пользователю, помимо

вычислительных мощностей, инструментарий по разработке ИИ. В данную категорию можно отнести различные облачные сервисы для разработки ПО. Ярким примером является Google Colab [32]. Несмотря на то что данное решение не является специализированным, оно все же предоставляет пользователю возможности по разработке и исследованию ИИ. При этом существуют решения, направленные именно на работу с ИИ [33]. IaaS и ИИ достаточно трудно выделить в специализированную категорию, так как в данном случае поставщик услуг предоставляет исключительно вычислительные ресурсы и инфраструктуру связывающую их. В данном случае специализация на ИИ достигается за счет двух факторов: за счет того, что аппаратное обеспечение технической составляющей ОВ подобрано таким образом, чтобы ИИ работал максимально эффективно; а также за счет самой позиции поставщика услуг [34]. В рассмотренных случаях ИИ выступает как объект, либо являющийся частью услуги, либо являющийся фактическим потребителем вычислительных мощностей, однако существуют работы, в которых рассматриваются и другие возможности по применению ИИ в ОВ [35–38]. Например, ИИ может применяться для построения инфраструктуры и балансировки нагрузки.

2.4 Конфиденциальность данных в облачных вычислениях

Если проекты, как в случаях [1, 2], являются открытыми, т. е. изначально не содержат конфиденциальной информации, то требования к их безопасности невелики. Однако любой ИИ может применяться в задачах, связанных с обработкой конфиденциальных данных. Это может быть медицинская информация [39–44], банковские [45–48], государственные [49, 50] и всевозможные личные данные. В данных случаях возникает ситуация, в связи с которой ОВ часто подвергаются критике как в обществе в целом, так и в научном сообществе. При обработке какой-либо информации в облаке к ней имеют доступ обе стороны – пользователь и поставщик услуги. В этом случае степень конфиденциальности данных устанавливается соглашением между пользователем и поставщиком, а то, как соблюдается данное соглашение находится в зоне ответственности поставщика услуг. Не редки случаи, когда конфиденциальные данные, хранимые или обрабатываемые различными поставщиками услуг, были скомпрометированы. В таком случае пользователь вынужденно отдает предпочтение поставщикам, которые либо зарекомендовали себя, как надежные с точки зрения обеспечения конфиденциальности обрабатываемых данных, либо утверждают о низкой вероятности компрометации за счет применения эффективных методов защиты. При этом возникает ряд вопросов. Какой способ обеспечения конфиденциальности является эффективным для хранения данных? Какие методы способны обеспечить необходимый уровень конфиденциальности в условиях обработки данных с использованием ИИ в целом и Больших данных в частности?

Целью данной работы является ответить на заданные вопросы и предоставить читателю возможность ознакомится с актуальными методами обеспечения конфиденциальности ОВ, использующих ИИ.

Для более наглядной демонстрации ретроспективного обзора распределенных вычислительных технологий и ИИ была составлена таблица (табл. 1). Рассмотренные выше работы представлены не в хронологическом порядке, табл. 1 призвана его восстановить. Отметим, что некоторые технологии впервые представлены намного раньше, чем они получили развитие и были изучены в полном объеме.

Анализируя данные табл. 1 и представленный выше обзор, можно заметить, что теоретические модели ИИ активно развивались с середины XX в., однако практические модели получили свое развитие в конце XX – начале XXI в. Так же из таблицы видно, что развитие практических моделей ИИ совпадает по времени с поздними этапами развития распределенных вычислений (до конца XX в.) и зарождением ОВ в начале XXI в. Данное совпадение не случайно и связано с тем, что развитие распределенных и облачных вычислений все это время шло по пути агрегирования все больших вычислительных мощностей, которых так не хватало большим моделям ИИ. Таким образом, справедливо утверждение, что распределенные вычисления внесли немалый вклад в развитие моделей ИИ, а ОВ позволяют развивать и создавать современные вычислительно затратные модели ИИ, сложность которых растет с каждым днем.

Учитывая вышесказанное, можно утверждать, что проблема конфиденциальности ОВ имеет такое же значение для ИИ, как для любого “потребителя” ОВ, коим ИИ по сути и является. Далее будут рассмотрены методы обеспечения конфиденциальности ИИ в ОВ.

Таблица 1. Историческая справка по рассмотренным технологиям

Год	Распределенные вычисления	Облачные технологии	Искусственный интеллект
1943	—	—	Концепция НС [51, 52]
1954	—	—	Зарождение генетических алгоритмов [53]
1959	—	—	Машинное обучение [54]
1962	Модель коллективных вычислений [55]	—	—
1966	—	—	Появление языковых моделей [56]
1978	Принципы распределения работы между процессорами [57]	—	—
1980		—	Теоретическое описание ГО [58]
1992	Зарождение GRID [59]	—	—
1996	Проект GIMPS по поиску целых чисел [60]	—	—
1999	Проект SETI на базе BOINC [61]	—	—
2000	—	—	Начало практического применения ГО [62]
			Компьютерное зрение [63]
2006	—	Зарождение концепции облачных вычислений [64]	—
2008	—	Определение концепции облачных вычислений как услуги [65]	—
2009	—	Запуск Google Apps [66]	—
2011	—	Стандартизация SaaS, PaaS и IaaS как моделей обслуживания в ОВ [25–27]	—
2015	—	Развитие туманных вычислений как основы для Интернета Вещей [67]	—
		Запуск OpenFog [68]	
2018	—	—	GPT [69]

3. СОВРЕМЕННЫЕ МЕТОДЫ И АЛГОРИТМЫ ОБЕСПЕЧЕНИЯ КОНФИДЕНЦИАЛЬНОСТИ ИИ

3.1. Дифференциальная конфиденциальность

Понятие конфиденциальности в некоторой степени является не строгим. В зависимости от ситуации она может рассматриваться с разных сторон. Например, с точки зрения медицины достаточно, чтобы данные были анонимны, т. е. история болезни была обезличена. В работе [70] проводится исследование методов ГО для обеспечения мягкой конфиденциальности. Авторы достигают своей цели путем внедрения дифференциальной конфиденциальности, т. е. путем перемешивания частных обезличенных данных вместе с синтезированными данными. Авторы продемонстрировали высокую точность данного метода. Однако данный метод имеет ряд недостатков: основой конфиденциальности является внесение дополнительного шума и расстояний между точками во время выполнения градиентного спуска. Это не только повышает вычислительную сложность расчетов, но и не обеспечивает надлежащей безопасности данных, так как если злоумышленник получит доступ к данным во время обработки – он сможет убрать лишний шум.

С другой стороны, в работе [71] рассмотрен принципиально отличающийся от вышеприведенного подход к обеспечению конфиденциальности данных. Авторы рассматривают возможность обучения и использования нейронной сети группой пользователей без необходимости раскрытия конфиденциальных данных друг другу. Подобная возможность обеспечивается особенностью стохастического градиентного спуска, которая позволяет выполнять его параллельно и асинхронно. Авторы также утверждают, что их решение позволяет участникам обучать нейронную сеть независимо на своих собственных (конфиденциальных) наборах, делясь подмножествами ключевых параметров. В целом, данный метод позволяет обеспечить конфиденциальность среди участников группы, однако, оказывается неэффективным в случае предварительного сговора участников либо внешней атаки.

3.2. Схемы разделения секрета

В контексте построения конфиденциального ИИ в ОВ необходимо также рассмотреть схемы разделения секрета (СРС) [4]. Приведем кратко основные теоретические аспекты, связанные с СРС. Секрет S разделяется дилером D между N участниками таким образом, что для дешифровки какой-либо информации потребуется объединение долей S_i всех участников схемы обратно в секрет, $i \in 1,2,\ldots,N$, где N – количество участников. Если секретом является, например, ключ для схемы шифрования, конфиденциальность системы возрастает.

Существуют два типа схем разделения секрета: полные [4] и пороговые [72]. Полные предполагают, что для восстановления необходимы все доли секрета, пороговые же, что необходимо определенное количество долей, но не все. Пороговые схемы разделения секрета на практике используются гораздо чаще ввиду своей гибкости. Пороговые СРС позволяют снизить вычислительную нагрузку, при этом порог задается такой, чтобы злоумышленник не смог овладеть необходимым количеством долей секрета, либо чтобы не смог состояться предварительный сговор необходимого количества участников.

В работе [73] авторы предлагают распределенное ГО, когда участники обучают нейронную сеть с помощью своих конфиденциальных наборов. Авторы приводят результаты исследования, в котором удалось сохранить конфиденциальность при распределенном ГО в облаке с недоверенными участниками. Работа демонстрирует функционал, предоставляемый СРС при работе с ИИ, и сам факт возможности конфиденциального обучения. Авторами применяется схема Шамира [72]. За время своего существования схема доказала свою пригодность с точки зрения безопасности. Однако если мы рассматриваем применение методов СРС с точки зрения ОВ, то на проблемы безопасности также накладываются проблемы надежности, дополнительные корректирующие коды накладывают на систему дополнительные нагрузки. Для нивелирования необходимости применения дополнительных корректирующих кодов можно применять СРС, основанные на системе остаточных классов (СОК) [74], например, такие как СРС Асмута–Блума [75] и СРС Миньотта [76].

В работе [77] рассмотрена организация федеративного обучения нейронных сетей в облачных системах с применением СРС Асмута–Блума для обеспечения конфиденциальности [78]. Также авторами рассматривается СРС Миньотта, однако существуют работы доказывающие ее непригодность для обеспечения безопасности.

В работе [79] авторы предлагают протокол конфиденциальной передачи данных в условиях работы с нейронными сетями. Протокол передачи основан на СРС. Основной упор в статье сделан на скорость обработки данных при сохранении их конфиденциальности. Авторами показана эффективность их решения по сравнению с некоторыми методами гомоморфного шифрования, которое также применимо при построении конфиденциального ИИ в ОВ.

3.3. Гомоморфное шифрование

Первые работы по гомоморфному шифрованию (ГШ) появились несколько десятилетий назад. ГШ допускает обработку зашифрованных данных без необходимости проведения операции дешифрования. Гомоморфные схемы шифрования бывают гомоморфными по сложению, умножению или одновременно по обеим перечисленным арифметическим операциям. Такая особенность характерна для различных ассиметричных шифров. Так, например, гомоморфное сложение поддерживается схемами, представленными в работах [80, 81], а гомоморфное умножение схемами из работ [82, 83]. Стоит отметить, что применение таким схемам шифрования находится и сейчас, например, в работе [84] авторы применяют модифицированную схему Эль-Гамаля для обучения нейронной сети, основываясь на том факте, что схема Эль-Гамаля гомоморфна по умножению. Авторы используют линейную аппроксимацию функции активации. Данный

метод является довольно узконаправленным и трудномасштабируем.

В 2009 г. Джентри разработал полностью гомоморфную схему шифрования (ПГШ) [85]. Данная схема была не достаточно вычислительно эффективна, однако дальнейшие исследования его последователей позволили повысить эффективность до уровня, достаточного для реального практического применения ПГШ [86–91]. ПГШ позволяет выполнять как гомоморфное сложение, так и умножение (однако имеет ограничение на количество умножений с одним ключом), кроме того, существуют схемы, которые позволяют выполнять полиномиальные операции над шифротекстами [92].

Такой набор операций позволяет реализовать большое количество алгоритмов ИИ, что было достаточно быстро обнаружено многими исследователями ИИ. Рассмотрим разработанные ими методы.

CryptoNets [93]. В данном решении авторы предлагают применение ГШ в работе конфиденциального облачного хранилища с возможностью обработки нейронными сетями. Основным достигнутым результатом является точность работы нейронной сети по распознаванию образов, равная 99%.

В случае SEALion [94] предлагается решение, основанное на TensorFlow [95] и SEAL [96]. TensorFlow реализует вычисления в тензорах, SEAL само ГШ. С помощью разработанного решения авторы реализуют сверточные нейронные сети (СНС), в качестве метода обучения применяется метод опорных векторов. В работе представлены несколько моделей НС, точность определяется на основе качества распознавания изображений цифр. Авторы показывают, что их решения более эффективны чем, например, в работе [93]. Похожим решением является TenSEAL [97], эта библиотека предоставляет возможности по работе с МО, НС и СНС совместно с ГШ. В качестве недостатка можно выделить сложности при обучении на зашифрованных данных, при этом обученная НС демонстрирует быстродействие и дает точные результаты.

В работе [98] рассматриваются вариации конфиденциальных СНС с ГШ. Авторами предлагается СНС, в которой применяется целочисленная схема ПГШ – BGV [99]. Основной акцент сделан на возможности вычисления значения функции активации ReLU от шифротекстов, полученных согласно схеме BGV. Авторами было определенно три требования для НС – точность, конфиденциальность и эффективность, результатом работы является полиномиально приближенная функция ReLU, которая позволила уменьшить мультипликативную глубину, тем самым повысив эффективность работы сети. Данную работу можно считать достаточно важной по нескольким причинам: во-первых, показан сам факт возможности полностью конфиденциальных расчетов; во-вторых, показана возможность построения и применения нейронных сетей с ПГШ.

В работе [100] рассматривается сохраняющее конфиденциальность ГО. Авторы противопоставляют свои исследования многим работам, которые были рассмотрены ранее, сосредотачиваясь на устранении недостатков, которые были озвучены выше. Основной результат основан на применении методов аппроксимации для различных функций активации, например, таких как ReLU и Softmax. Для обеспечения большого количества гомоморфных умножений в схеме CKKS авторы предлагают свою модификацию процедуры bootstrapping, которая основана на том факте, что схема CKKS [92] применяет систему остаточных классов для ускорения арифметических вычислений. Этот результат интересен тем, что повышает быстродействие методов ГШ для конкретных задач, тогда как повышение быстродействия и применимости ГШ в целом является основным направлением исследований в этой области в настоящее время.

ГШ позволяет решить проблемы безопасности ОВ и ОТ в целом, создав прозрачную среду для свободных и конфиденциальных вычислений в облаках. Проведенный анализ методов ГШ в ИИ, позволяет заявить о том, что такого мнения придерживаются многие исследователи. Выше рассмотрены схемы ПГШ, которые в большей степени основаны на криптографических решетках, однако существуют схемы ПГШ, которые построены на СРС. Такой подход позволяет расширить возможности ПГШ в ОТ. Далее рассмотрены гибридные модели, которые совмещают методы ПГШ и СРС.

3.4. Гибридные системы гомоморфного шифрования и схем разделения секрета

Учитывая результаты в направлении повышения конфиденциальности ИИ, полученные за счет применения СРС и ПГШ, перспективной выглядит идея создания гибридной системы ПГШ–СРС. Такая гибридная система позволит усилить безопасность СРС за счет обработки информации в зашифрованном виде с помощью ПГШ в пространстве одного локального узла. Для передачи данных между узлами применяются алгоритмы безопасности на основе СРС. Таким

образом появляется возможность регулировать баланс, смещая акцент в сторону безопасности либо производительности.

В работе [101] предложено решение, в котором используются СРС и ПГШ для реализации сохраняющей конфиденциальность НС. Авторы описывают различные алгоритмы, применяемые в данной системе, например, алгоритмы безопасности, генерации ключей, работы СРС и т. п., а также подробно разбирают, как происходит работа НС при таком подходе к обеспечению безопасности. Основной упор делается на демонстрации эффективности, безопасности, точности полученной НС и ее сравнении с другими решениями, однако полученные данные являются теоретическими. Рассмотренная работа замечательна тем, что несмотря на отсутствие конкретики в плане применяемых ПГШ и СРС, она показывает саму возможность реализации такой системы и описывает ее характеристики, побуждая других исследователей заниматься разработками в данной сфере.

4. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В ходе исследования были проанализированы три группы методов обеспечения конфиденциальности искусственного интеллекта (табл. 2). Представленные ниже данные основаны на информации, представленной в обозреваемых работах.

Анализируя полученные результаты (табл. 2), можно подвести следующий итог.

Методы дифференциальной конфиденциальности преимущественно основаны на изменениях обучающей выборки. Предлагается добавление лишних шумов в данные, за счет которого возрастает степень конфиденциальности, одновременно с которой серьезно снижается эффективность расчетов. В строках 1 и 2 жирным шрифтов выделены значения вычислительной сложности. В первом случае ε – означает добавочный шум. Во втором случае δ – количество асинхронных узлов. Учитывая достигаемый при этом уровень конфиденциальности, при передаче данных требуется применение дополнительных мер защиты. При обработке степень конфиденциальности средняя – для кражи данных злоумышленнику необходимо будет либо отфильтровать информацию от шума, либо взять под контроль несколько узлов, чтобы отследить поток данных. Кроме того, дополнительное влияние на эффективность системы окажет необходимость применения корректирующих кодов во избежание возникновения коллизий либо потери данных.

В случае СРС схемы, основанные на СОК, позволяют повысить надежность системы за счет

Таблица 2. Результаты аналитического обзора методов

№	Метод	Вычислительная сложность	Конфиденциальность при передаче данных	Конфиденциальность при обработке данных	Обеспечение надежности	Рассмотренный метод ИИ
1	Модифицированный градиентный спуск	$\boldsymbol{O\left(D^2N+(D+\varepsilon)^3\right)}$	Низкая	Средняя	Отсутствует	AC-GAN [70]
2	Асинхронный градиентный спуск	$\boldsymbol{O\left(D^2(N\delta)+D^3\right)}$	Низкая	Средняя	Отсутствует	СНС [71]
3	СРС Шамира	$O(N^2)$	Высокая	Низкая	Отсутствует	ГО [73]
4	СРС Асмута–Блума	$O\left(\log_2^2(N)+\log_2(N^2)\right)$	Высокая	Низкая	Корректирующие коды СОК	Федеративное обучение [77, 79]
5	СРС Миньота	$O(N^2)$	Низкая	Низкая	Корректирующие коды СОК	Федеративное обучение [77]
6	BFV	$O(N\log N)$	Высокая	Высокая	Корректирующие коды СОК (в теории)	НС, СНС [93, 94]
7	BGV	$O(N\log N)$	Высокая	Высокая	Отсутствует	НС, СНС [98]
8	СККС	$O(N\log N)$	Высокая	Высокая	Корректирующие коды СОК (в теории)	НС, СНС, ГО [93, 94, 97, 100]
9	Гибрид СРС–ПГШ	$O(N\log N)$	*Высокая*	*Высокая*	*Корректирующие коды СОК*	*НС*, *СНС*, *ГО* [101]

использования самокорректирующих свойств СОК. Однако конфиденциальность во время обработки данных необходимо обеспечивать дополнительными методами шифрования. Также, в данном случае, для перехвата данных злоумышленнику потребуется взять под контроль пороговое количество узлов. При построении СРС порог для восстановления секрета рассчитывается так, чтобы время, затраченное на компрометацию долей секрета, было большим, чем время актуальности данных.

Обеспечить полную конфиденциальность позволяет ПГШ. Однако оно имеет существенный недостаток. Несмотря на то, что вычислительная сложность шифрования является невысокой, по сравнению с другими алгоритмами, вычислительная сложность обработки данных – намного выше, чего не скрывают авторы схем.

Последней из рассмотренных категорий были гибридные СРС–ПГШ-методы. В настоящий момент они представлены лишь теоретическими моделями, однако, анализируя их, можно оценить возможные характеристики системы.

Проведенное исследование показало, что разные исследовательские группы заинтересованы в разработке ИИ обладающего высоким уровнем конфиденциальности. Однако в настоящий момент релевантных методов достижения подобного уровня не существует. В дальнейшем планируется исследование наиболее перспективных с точки зрения обеспечения конфиденциальности методов, основанных на СРС–ПГШ. В частности планируется разработка НС, использующей схему ПГШ CKKS, а также СРС Асмута–Блума. Обосновывая выбор именно этих алгоритмов, можно заметить, что среди множества схем ПГШ именно CKKS представляет для исследователей наибольший интерес, а среди множества СРС, основанных на СОК, именно СРС Асмута–Блума имеет наилучшие характеристики с точки зрения безопасности. Использование СОК в CKKS позволит повысить эффективность решения, а в СРС Асмута–Блума – надежность обработки данных. Важной частью исследования будет являться определение точности результатов вычислений, ввиду целочисленной природы СОК, а также ошибки приближения в CKKS.

5. ЗАКЛЮЧЕНИЕ

В данной работе были исследованы методы построения, сохраняющего конфиденциальность ИИ в ОВ. На первом этапе исследования проведен аналитический обзор как методов ИИ, так и методов ОВ. По результатам обзора были определены критерии безопасности ИИ в ОВ. Далее был выполнен второй этап аналитического обзора, а именно обзор методов обеспечения конфиденциальности ИИ, на основе которого было выделено четыре группы методов:

- дифференциальная конфиденциальность;
- схемы разделения секрета;
- гомоморфное шифрование;
- гибридные методы, основанные на СРС–ПГШ.

На основании результатов проведенного исследования были определенны положительные и отрицательные стороны рассмотренных методов, сформировано представление о современном состоянии проблемы обеспечения конфиденциальности ИИ в ОВ, а также обозначены подходы к ее решению. Аналитический обзор показал, что в настоящий момент релевантного решения не существует. Решения, обеспечивающие наиболее высокий уровень конфиденциальности, имеют низкую эффективность ввиду необходимости выполнения сложных вычислений. ПГШ поддерживает операции сложения и умножения над зашифрованными значениями, такие операции как определение знака числа, деление, матричные операции – не реализованы в полном объеме. СРС не позволяют обрабатывать информацию в зашифрованном виде. Конфиденциальность обеспечивается за счет неразглашения источника части набора данных одного участника перед другими. Дифференциальная конфиденциальность обеспечивает анонимность во время работы нейронной сети, при этом не имеет защиты от перехвата данных. В качестве возможного решения теоретически установлен гибридный метод, основанный на СРС–ПГШ, однако он требует детального исследования.

В будущих работах будет проведено исследование СРС–ПГШ, а именно разработка прототипа и исследование его характеристик.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке Российского научного фонда 19-71-10033, https://rscf.ru/project/19-71-10033/.

СПИСОК ЛИТЕРАТУРЫ

1. *Brown T. et al.* Language models are few-shot learners // Advances in neural information processing systems. 2020. V. 33. P. 1877–1901.
2. OpenAI, GPT-4 Technical Report. arXiv, 27 март 2023 г. https://doi.org/10.48550/arXiv.2303.08774

3. *Douligeris C., Mitrokotsa A.* DDoS attacks and defense mechanisms: classification and state-of-the-art // Computer networks. 2004. V. 44. № 5. P. 643–666.
4. *Beimel A.* Secret-Sharing Schemes: A Survey // Coding and Cryptology, Y.M. Chee, Z. Guo, S. Ling, F. Shao, Y. Tang, H. Wang, and C. Xing, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. 2011. P. 11–46. https://doi.org/10.1007/978-3-642-20901-7_2
5. *Mahesh B.* Machine learning algorithms-a review // International Journal of Science and Research (IJSR). [Internet]. 2020. V. 9. № 1. P. 381–386.
6. *Kaelbling L.P., Littman M.L., Moore A.W.* Reinforcement learning: A survey // Journal of artificial intelligence research. 1996. V. 4. P. 237–285.
7. *Srinivas M., Patnaik L.M.* Genetic algorithms: A survey // Computer. 1994. V. 27. № 6. P. 17–26.
8. *Spragins J.* Learning without a teacher // IEEE Transactions on Information Theory. 1996. V. 12. № 2. P. 223–230.
9. *Liu B.* Supervised Learning // Web Data Mining. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. P. 63–132. https://doi.org/10.1007/978-3-642-19460-3_3
10. *Wang S.-C.* Artificial Neural Network // Interdisciplinary Computing in Java Programming. Boston, MA: Springer US, 2003. P. 81–100. https://doi.org/10.1007/978-1-4615-0377-4_5
11. *Park H., Kim S.* Chapter Three – Hardware accelerator systems for artificial intelligence and machine learning // Advances in Computers. V. 122, S. Kim and G.C. Deka, Eds., in Hardware Accelerator Systems for Artificial Intelligence and Machine Learning. V. 122. Elsevier, 2021. P. 51–95. https://doi.org/10.1016/bs.adcom.2020.11.005
12. *Hwang D. H., Han C.Y., Oh H.W., Lee S.E.* ASimOV: A Framework for Simulation and Optimization of an Embedded AI Accelerator // Micromachines. 2021. V. 12. № 7. https://doi.org/10.3390/mi12070838
13. *Mishra A., Yadav P., Kim S.* Artificial Intelligence Accelerators // Artificial Intelligence and Hardware Accelerators, A. Mishra, J. Cha, H. Park, and S. Kim, Eds. Cham: Springer International Publishing, 2023. P. 1–52. https://doi.org/10.1007/978-3-031-22170-5_1
14. *Carminati M., Scandurra G.* Impact and trends in embedding field programmable gate arrays and microcontrollers in scientific instrumentation // Review of Scientific Instruments. 2021. V. 92.№ 9. https://pubs.aip.org/aip/rsi/article-abstract/92/9/091501/1030652
15. *Shawash J., Selviah D.R.* Real-time nonlinear parameter estimation using the Levenberg–Marquardt algorithm on field programmable gate arrays // IEEE Transactions on industrial electronics. 2012. V. 60. № 1. P. 170–176.
16. *Ruiz-Rosero J., Ramirez-Gonzalez G., Khanna R.* Field programmable gate array applications – A scientometric review // Computation. 2019. V. 7. № 4. P. 63.
17. *Mellit A., Kalogirou S.A.* MPPT-based artificial intelligence techniques for photovoltaic systems and its implementation into field programmable gate array chips: Review of current status and future perspectives // Energy. 2014. V. 70. P. 1–21.
18. *Goodfellow I., Bengio Y., Courville A.* Deep learning. MIT press, 2016. https://books.google.com/books?hl=ru&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=Deep+Learning&ots=MNV5aolzSS&sig=waXAS6C-_v-48H2qbW9rMFkEhFY
19. *Bouvrie J.* Notes on convolutional neural networks. 2006. http://web.mit.edu/jvb/www/papers/cnn_tutorial.pdf
20. *Rawat W., Wang Z.* Deep convolutional neural networks for image classification: A comprehensive review // Neural computation. 2017. V. 29; № 9. P. 2352–2449.
21. *Needham R.M., Herbert A.J.* The Cambridge distributed computing system, 1983.
22. *Adiga N.R. et al.* An overview of the BlueGene/L supercomputer // SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, IEEE, 2002. P. 60–60. https://ieeexplore.ieee.org/abstract/document/1592896/
23. *Jacob B., Brown M., Fukui K., Trivedi N.* Introduction to grid computing // IBM redbooks, 2005. P. 3–6.
24. *Foster I., Zhao Y., Raicu I., Lu S.* Cloud computing and grid computing 360-degree compared // 2008 grid computing environments workshop, IEEE, 2008. P. 1–10. https://ieeexplore.ieee.org/abstract/document/4738445/?casa_token=TbNOHOEaljQAAAAA:j6MuEJKmrGL8iCvH-HzRnmI2k5UKn5y1w7hC4MNJanJXZPfiBC_XKLoTFsCImP1RYzyKfRKiCE0
25. *Cusumano M.* Cloud computing and SaaS as new computing platforms // Commun. ACM, April, 2010. V. 53. № 4. P. 27–29. https://doi.org/10.1145/1721654.1721667
26. *Rodero-Merino L., Vaquero L.M., Caron E., Muresan A., Desprez F.* Building safe PaaS clouds: A survey on security in multitenant software platforms // Computers & security. 2012. V. 31. № 1. P. 96–108.
27. *Bhardwaj S., Jain L., Jain S.* Cloud computing: A study of infrastructure as a service (IAAS) // International Journal of engineering and information Technology. 2010. V. 2. № 1. P. 60–63.
28. *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. 2014. V. 41. P. 424–440.
29. *Lehner W., Sattler K.-U.* Database as a service (DBaaS) // 2010 IEEE 26th International Conference on Data Engineering (ICDE2010), IEEE, 2010. P. 1216–1217.

https://ieeexplore.ieee.org/abstract/document/5447723/?casa_token=uaXogPZV0C0AAAAA:4Dg_40-GvhUsuHXFKUOgxZ_ZyGlCOqjcZtpRo-K6UosB-k-_Wh5wAmJIBtHYRE9OLXZ1xwVKuLAE

30. *Meng S., Liu L.* Enhanced monitoring-as-a-service for effective cloud management // IEEE Transactions on Computers. 2012. V. 62. № 9. P. 1705–1720.
31. *Weng Q. et. al.* {MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters // 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), 2022. P. 945–960. https://www.usenix.org/conference/nsdi22/presentation/weng
32. *Bisong E.* Google Colaboratory // Building Machine Learning and Deep Learning Models on Google Cloud Platform. Berkeley, CA: Apress, 2019. P. 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7
33. H2O AI Cloud. https://h2o.ai/platform/ai-cloud/
34. NVIDIA NGC | NVIDIA. https://www.nvidia.com/en-us/gpu-cloud/
35. *Tang J.* Artificial intelligence-based e-commerce platform based on SaaS and neural networks // 2020 Fourth International Conference on Inventive Systems and Control (ICISC). IEEE, 2020. P. 421–424. https://ieeexplore.ieee.org/abstract/document/9171193/?casa_token=TmYwFdLDXq0AAAAA:8P5V-VcZS_KWCXEnEm8xk2RPMV5kfWF27K9S9O9Z-5fYh273EkseT7j0Jf7jZYAMOnPUX0l-5sCbs
36. *Yathiraju N.* Investigating the use of an Artificial Intelligence Model in an ERP Cloud-Based System // International Journal of Electrical, Electronics and Computers. 2022. V. 7. № 2. P. 1–26.
37. *Mishra S., Tripathi A.R.* AI business model: an integrative business approach // J. Innov. Entrep. Dec. 2021. V. 10. № 1. P. 18. https://doi.org/10.1186/s13731-021-00157-5
38. *Mishra D., Shekhar S.* Artificial Intelligence Candidate Recruitment System using Software as a Service (SaaS) Architecture // International Research Journal of Engineering and Technology. 2018. V. 05. № 05. P. 3804–3808.
39. *Cadario R., Longoni C., Morewedge C.K.* Understanding, explaining, and utilizing medical artificial intelligence // Nature human behaviour. 2021. V. 5. № 12. P. 1636–1642.
40. *Kim M., Song Y., Wang S., Xia Y., Xiang X.* Secure logistic regression based on homomorphic encryption: Design and evaluation // JMIR medical informatics. 2018. V. 6. № 2. P. e8805.
41. *Klonoff D.C.* Fog computing and edge computing architectures for processing data from diabetes devices connected to the medical internet of things // Journal of diabetes science and technology. 2017. V. 11. № 4. P. 647–652.
42. *Kocabas O., Soyata T.* Utilizing homomorphic encryption to implement secure and private medical cloud computing // 2015 IEEE8th International Conference on Cloud Computing. IEEE, 2015. P. 540–547.
43. *Liu R., Rong Y., Peng Z.* A review of medical artificial intelligence // Global Health Journal. 2020. V. 4. № 2. P. 42–45.
44. *Sun X., Zhang P., Sookhak M., Yu J., Xie W.* Utilizing fully homomorphic encryption to implement secure medical computation in smart cities // Personal and Ubiquitous Computing. 2017. V. 21. № 5. P. 831–839.
45. *Kaya O., Schildbach J., AG D.B., Schneider S.* Artificial intelligence in banking // Artificial intelligence. 2019. https://www.dbresearch.com/PROD/RPS_EN-PROD/PROD0000000000495172/Artificial_intelligence_in_banking%3A_A_lever_for_pr.pdf
46. *Rahman M., Ming T.H., Baigh T.A., Sarker M.* Adoption of artificial intelligence in banking services: an empirical analysis // International Journal of Emerging Markets. 2021. https://www.emerald.com/insight/content/doi/10.1108/IJOEM-06-2020-0724/full/html
47. *Sadok H., Sakka F., El Maknouzi M.E.H.* Artificial intelligence and bank credit analysis: A review // Cogent Economics & Finance. Dec. 2022. V. 10. № 1. P. 2023262. https://doi.org/10.1080/23322039.2021.2023262
48. *Smith A., Nobanee H.* Artificial intelligence: in banking A mini-review // Available at SSRN3539171, 2020. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3539171
49. *Reis J., Santo P.E., Melão N.* Artificial Intelligence in Government Services: A Systematic Literature Review // New Knowledge in Information Systems and Technologies. V. 930. Á. Rocha, H. Adeli, L.P. Reis, and S. Costanzo, Eds., in Advances in Intelligent Systems and Computing. V. 930. Cham: Springer International Publishing. 2019. P. 241–252. https://doi.org/10.1007/978-3-030-16181-1_23
50. *Valle-Cruz D., Alejandro Ruvalcaba-Gomez E., Sandoval-Almazan R., Ignacio Criado J.* A Review of Artificial Intelligence in Government and its Potential from a Public Policy Perspective // Proceedings of the 20th Annual International Conference on Digital Government Research. Dubai United Arab Emirates: ACM, June 2019. P. 91–99. https://doi.org/10.1145/3325112.3325242
51. *Pitts W.* The linear theory of neuron networks: The dynamic problem // The bulletin of mathematical biophysics. 1943. V. 5. P. 23–31.
52. *Khare S.S., Gajbhiye A.R.* Literature Review on Application of Artificial Neural Network (Ann) In Operation of Reservoirs // International Journal of computational Engineering research (IJCER). June 2013. V. 3. № 6. P. 63.
53. *Seesing A.* Evotest: Test case generation using genetic programming and software analysis // Operations Research. 1954. V. 2. P. 393–410.

54. *Samuel A.L.* Machine learning // The Technology Review. 1959. V. 62. № 1. P. 42–45.
55. *Evreinov È.V., Kosarev I.* Однородные универсальные вычислительные системы высокой производительности (No Title), 1966. https://cir.nii.ac.jp/crid/1130282272859765760
56. *Gold E.M.* Language identification in the limit // Information and control. 1967. V. 10. № 5. P. 447–474.
57. *Глушков В.М.* Вычислительная система, 1996. https://elibrary.ru/item.asp?id=41074434
58. *Huang X.* Deep-learning based climate downscaling using the super-resolution method, 1981. https://pdfs.semanticscholar.org/cf5c/3b29559ababba5a889444632e1c91d6b78fc.pdf
59. *Smarr L., Catlett C.E.* Metacomputing // Grid Computing, 1st ed., F. Berman, G. Fox, and T. Hey, Eds., Wiley, 2003. P. 825–835. https://doi.org/10.1002/0470867167.ch37
60. *Buske D., Keith S.* GIMPS Finds Another Prime! // Math Horizons. April 2000. V. 7. № 4. P. 19–21. https://doi.org/10.1080/10724117.2000.11975124
61. *Anderson D.P.* Boinc: A system for public-resource computing and storage // Fifth IEEE/ACM international workshop on grid computing. IEEE, 2004. P. 4–10. https://ieeexplore.ieee.org/abstract/document/1382809/?casa_token=cjAKtADFAKwAAAAA:-WGH_xmovZAUi-kr_PA-h3nXtuizBL829DPFlC0B6pbcCoApRKDCZLwFWxzfYdT0WauFC5c6EQw1
62. *Du T., Shanker V.* Deep learning for natural language processing // Eecis. Udel. Edu, 2009. P. 1–7.
63. *Davies E.R.* Machine vision: theory, algorithms, practicalities. Elsevier, 2004. https://books.google.com/books?hl=ru&lr=&id=uY-Z3vORugwC&oi=fnd&pg=PP1&dq=Machine+Vision+:+Theory,+Algorithms,+Practicalities&ots=QOl9U9_MBf&sig=w0poN6d3IGeXs4oacagO4MlnxYs
64. *Mell P., Grance T.* The NIST Definition of Cloud Computing // National Institute of Standards and Technology Special Publication. 2011. V. 53. P. 1–7.
65. *Finkelstein R.* Analyzing Trend of Cloud Computing and it's Enablers using Gartner Strategic Technology, 2004. https://www.researchgate.net/profile/Amol-Adamuthe/publication/308747055_Analyzing_Trend_of_Cloud_Computing_and_it's_Enablers_using_Gartner_Strategic_Technology/links/59a929d3a6fdcc2398414d6f/Analyzing-Trend-of-Cloud-Computing-and-its-Enablers-using-Gartner-Strategic-Technology.pdf
66. A history of cloud computing // Computer Weekly. https://www.computerweekly.com/feature/A-history-of-cloud-computing
67. *Dolui K., Datta S.K.* Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing // 2017 Global Internet of Things Summit (GIoTS), IEEE. 2017. P. 1–6.
68. OpenFog, OPC Foundation. https://opcfoundation.org/markets-collaboration/openfog/
69. *Radford A., Narasimhan K., Salimans T., Sutskever I.* Improving language understanding by generative pre-training" 2018. https://www.mikecaptain.com/resources/pdf/GPT-1.pdf
70. *Beaulieu-Jones B.K. et al.* Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing // Circ: Cardiovascular Quality and Outcomes. Jul. 2019. V. 12. № 7. P. e005122. https://doi.org/10.1161/CIRCOUTCOMES.118.005122
71. *Shokri R., Shmatikov V.* Privacy-Preserving Deep Learning // Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. Denver Colorado USA: ACM, Oct. 2015. P. 1310–1321. https://doi.org/10.1145/2810103.2813687.
72. *Shamir A.* How to share a secret // Communications of the ACM. 1979. V. 22. № 11. P. 612–613.
73. *Duan J., Zhou J., Li Y.* Privacy-preserving distributed deep learning based on secret sharing // Information Sciences. 2020. V. 527. P. 108–127.
74. *Akushsky I.A., Yuditsky D.I.* Modular arithmetic in residue classes // Soviet Radio, 1968.
75. *Asmuth C., Bloom J.* A modular approach to key safeguarding // IEEE transactions on information theory. 1983. V. 29. № 2. P. 208–210.
76. *Mignotte M.* How to share a secret // Workshop on cryptography. Springer, 1982. P. 371–375.
77. *Tian T., Wang S., Xiong J., Bi R., Zhou Z., Bhuiyan M.Z.A.* Robust and privacy-preserving decentralized deep federated learning training: Focusing on digital healthcare applications // IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2023. https://ieeexplore.ieee.org/abstract/document/10058838/
78. *Barzu M., Ţiplea F.L., Drăgan C.C.* Compact sequences of co-primes and their applications to the security of CRT-based threshold schemes // Information Sciences. 2013. V. 240. P. 161–172.
79. *Ge Z., Zhou Z., Guo D., Li Q.* Practical Two-party Privacy-preserving Neural Network Based on Secret Sharing. http://arxiv.org/abs/2104.04709
80. *Paillier P.* Public-Key Cryptosystems Based on Composite Degree Residuosity Classes // Advances in Cryptology – EUROCRYPT '99. V. 1592, J. Stern, Ed., in Lecture Notes in Computer Science. V. 1592. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. P. 223–238. https://doi.org/10.1007/3–540–48910-X_16.
81. *Benaloh J.* Dense probabilistic encryption // Proceedings of the workshop on selected areas of cryptography, 1994. P. 120–128. https://sacworkshop.org/proc/SAC_94_006.pdf

82. *Rivest R. L., Shamir A., Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. Feb. 1978. V. 21. № 2. P. 120–126. https://doi.org/10.1145/359340.359342
83. *ElGamal T.* A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE transactions on information theory. 1985. V. 31. № 4. P. 469–472.
84. *Chen T., Zhong S.* Privacy-preserving backpropagation neural network learning // IEEE Transactions on Neural Networks. 2009. V. 20. № 10. P. 1554–1564.
85. *Gentry C.* A fully homomorphic encryption scheme // Stanford university, 2009.
86. *Gentry C.* Computing arbitrary functions of encrypted data // Communications of the ACM. 2010. V. 53. № 3. P. 97–105.
87. *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.
88. *Gentry C., Halevi S., Peikert C., Smart N.P.* Ring Switching in BGV-Style Homomorphic Encryption // Security and Cryptography for Networks. V. 7485. I. Visconti and R. De Prisco, Eds. Lecture Notes in Computer Science. V. 7485. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. P. 19–37. https://doi.org/10.1007/978-3-642-32928-9_2
89. *Gentry C., Sahai A., Waters B.* Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based // Annual Cryptology Conference. Springer, 2013. P. 75–92.
90. *van Dijk M., Gentry C., Halevi S., Vaikuntanathan V.V.* Fully homomorphic encryption over the integers // Annual international conference on the theory and applications of cryptographic techniques. Springer, 2010. P. 24–43.
91. *van Dijk M., Gentry C., Halevi S., Vaikuntanathan V.* Fully Homomorphic Encryption over the Integers // Advances in Cryptology – EUROCRYPT 2010. V. 6110. H. Gilbert, Ed., Lecture Notes in Computer Science. V. 6110. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 24–43. https://doi.org/10.1007/978-3-642-13190-5_2
92. *Cheon J. H., Kim A., Kim M., Song Y.* Homomorphic encryption for arithmetic of approximate numbers // International conference on the theory and application of cryptology and information security. Springer, 2017. P. 409–437.
93. *Gilad-Bachrach R., Dowlin N., Laine K., Lauter K., Naehrig M., Wernsing J.* Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy // International conference on machine learning, PMLR, 2016. P. 201–210. https://proceedings.mlr.press/v48/gilad-bachrach16.html
94. *van Elsloo T., Patrini G., Ivey-Law H.* SEALion: a Framework for Neural Network Inference on Encrypted Data. http://arxiv.org/abs/1904.12840
95. TensorFlow. https://www.tensorflow.org/?hl=ru
96. Microsoft SEAL. Microsoft. https://github.com/microsoft/SEAL
97. *Benaissa A., Retiat B., Cebere B., Belfedhal A.E.* TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption. http://arxiv.org/abs/2104.03152
98. *Chabanne H., De Wargny A., Milgram J., Morel C., Prouff E.* Privacy-preserving classification on deep neural network // Cryptology ePrint Archive, 2017. https://eprint.iacr.org/2017/035
99. *Brakerski Z., Gentry C., Vaikuntanathan V.* (Leveled) fully homomorphic encryption without bootstrapping // ACM Transactions on Computation Theory (TOCT). 2014. V. 6. № 3. P. 1–36.
100. *Lee J.-W. et al.* Privacy-preserving machine learning with fully homomorphic encryption for deep neural network // IEEE Access. 2022. V. 10. P. 30039–30054.
101. *Ryffel T., Tholoniat P., Pointcheval D., Bach F.* ARIANN: Low-Interaction Privacy-Preserving Deep Learning via Function Secret Sharing. arXiv, October 28, 2021. http://arxiv.org/abs/2006.04593

ANALYTICAL REVIEW OF CONFIDENTIAL ARTIFICIAL INTELLIGENCE: METHODS AND ALGORITHMS FOR DEPLOYMENT IN CLOUD COMPUTING

© 2024 E. M. Shiriaev[a], A. S. Nazarov[a], N. N. Kucherov[a], M. G. Babenko[a, b]

[a] *North Caucasus Federal University, Pushkin st. 1, Stavropol, 355017, Russia*

[b] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, Alexander Solzhenitsyn st. 25, Moscow, 109004, Russia*

The technologies of artificial intelligence and cloud systems have recently been actively developed and implemented. In this regard, the issue of their joint use, which has been topical for several years, has become more acute. The problem of data privacy preservation in cloud computing acquired the status of critical long before the necessity of their joint use with artificial intelligence, which made it even more complicated. This paper presents an overview of both the artificial intelligence and cloud computing techniques themselves, as well as methods to ensure data privacy. The review considers methods that utilize differentiated privacy; secret sharing schemes; homomorphic encryption; and hybrid methods. The conducted research has shown that each considered method has its pros and cons outlined in the paper, but there is no universal solution. It was found that theoretical models of hybrid methods based on secret sharing schemes and fully homomorphic encryption can significantly improve the confidentiality of data processing using artificial intelligence.

Keywords: cloud computing, artificial intelligence, neural network, secret sharing scheme, homomorphic encryption, residue number system

ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 41–48

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.89

RuGECToR: НЕЙРОСЕТЕВАЯ МОДЕЛЬ НА ОСНОВЕ ПРАВИЛ ДЛЯ ИСПРАВЛЕНИЯ ГРАММАТИЧЕСКИХ ОШИБОК НА РУССКОМ ЯЗЫКЕ

© 2024 г. И. А. Хабутдинов[a, b, *], А. В. Чащин[b, **], А. В. Грабовой[a, b, *], А. С. Кильдяков[b, ****], Ю. В. Чехович[b, c, *****]**

[a] *Московский физико-технический институт (национальный исследовательский университет)*
141700 Московская область, г. Долгопрудный, Институтский пер., 9, Россия
[b] *АО "Антиплагиат"*
117105 Москва, Варшавское шоссе, д. 33, эт. 14, комната 15в, Россия
[c] *Федеральный исследовательский центр "Информатика и управление" РАН*
119333 Москва, ул. Вавилова, д. 44, кор. 2, Россия
**E-mail: khabutdinov@ap-team.ru*
***E-mail: chashchin@ap-team.ru*
****E-mail: grabovoy@ap-team.ru*
*****E-mail: kildyakov@ap-team.ru*
******E-mail: chehovich@ap-team.ru*

Поступила в редакцию 21.02.2024
После доработки 18.03.2024
Принята к публикации 18.03.2024

Исправление грамматических ошибок является одной из основных задач обработки естественного языка. В настоящий момент наиболее эффективной моделью, использующей подход Sequence Tagging с открытым исходным кодом, для английского языка является модель GECToR. Для русского языка данная задача не имеет настолько эффективных решений ввиду отсутствия достаточного количества размеченных данных. Это послужило причиной проведения данного исследования. В исследовании описан процесс создания синтетического набора данных и обучения на нем модели. Архитектура GECToR адаптирована для русского языка и названа соответствующим образом – RuGECToR. Выбор архитектуры обусловлен тем, что в отличие от подхода Sequence-to-Sequence, она проста в интерпретации и не требует большого количества обучающих данных. Целью исследования было обучить модель таким образом, чтобы она обобщала морфологические свойства языка, а не подстраивалась под обучающую выборку. Представленная модель показала результат 82.5 на синтетических данных и 22.2 на наборе данных RULEC с точки зрения метрики $F_{0.5}$, при этом набор данных RULEC не использовался на этапе обучения.

Ключевые слова: обработка естественного языка, исправление грамматических ошибок, тегирование последовательности, генерация наборов данных, настройка параметров

DOI: 10.31857/S0132347424040048, **EDN:** PTJOCY

1. ВВЕДЕНИЕ

В современном мире проблема автоматического исправления грамматических ошибок (ИГО) является актуальной [1–3] в связи с увеличением объема текстовых данных [4]. Ручная проверка больших текстов является трудоемкой задачей. Кроме того, решение данной задачи имеет специфические приложения: проверка сочинений [5]; исправление сообщений в социальных сетях [6]; исправление текстов, написанных иностранными студентами [7]; поиск текстовых заимствований [8]. Последнее объясняется тем, что грамматические ошибки являются одним из способов обхода системы поиска текстовых заимствований [9]. Для обучения моделей поиска текстовых заимствований используются тексты научных статей, содержащие небольшое количество ошибок. Пользователи при этом могут нарочно допускать множество ошибок, влияя тем самым на предсказания модели. Это приводит к появлению состязательных атак (англ. adversarial attacks) на модели обработки естественного языка [10].

Большинство исследований в области ИГО ориентировано на английский язык [11]. Для многих других языков, включая русский, число исследований значительно меньше [12]. Особенно стоит отметить сложную морфологию русского языка, что усложняет решение задачи ИГО.

В настоящее время существуют два наиболее эффективных подхода решения задачи ИГО для английского языка, – Sequence-to-Sequence (Seq2Seq) [13, 14] и Sequence Tagging (ST) [15, 16]. В подходе Seq2Seq исходная последовательность представляет собой предложение с ошибками, а целевая последовательность – предложение без ошибок. Такой подход решения задачи ИГО работает хорошо, но обладают низкой скоростью работы, а также плохой интерпретируемостью, поскольку для определения типа ошибки необходимо использовать дополнительный функционал. Модели, основанные на подходе ST, не имеют данных проблем: не требуется полностью генерировать предложение без ошибок, достаточно лишь отметить ошибки в исходном предложении. Кроме того, они легко интерпретируемы, так как решают задачу классификации для каждого токена, сопоставляя ему нужное правило из заданного словаря корректирующих правил.

В [12] авторы сравнивают данные подходы для решения задачи ИГО в текстах на русском языке. Результаты показывают, что на небольшом объеме размеченных данных методы, основанные на подходе ST, существенно превосходят методы, основанные на подходе Seq2Seq. Для сравнения методов авторы представляют набор данных RULEC, содержащий предложения с размеченными грамматическими ошибками. Набор данных состоит из сочинений иностранных студентов.

Для решения задачи ИГО на английской языке наиболее эффективной ST-моделью является GECToR [15]. Данная модель состоит из кодировщика на основе архитектуры трансформер [17] и двухголового классификатора. Первая голова предсказывает наличие ошибки, а вторая – соответствующее правило для исправления ошибки. После этого соответствующее правило применяется к каждому токену последовательности для исправления. Если токен не содержит ошибок, то данному токену ставится в соответствие правило “$KEEP”, которое оставляет данный токен без изменений. Обучение модели GECToR состоит из трех этапов:

1) обучение на синтетических данных;

2) дообучение на корпусе данных, содержащем ошибки;

3) дообучение на комбинации из корпусов данных с ошибками и без ошибок.

В работе предложен алгоритм генерации синтетического набора данных, содержащего тексты на русском языке для первого этапа обучения. Модель GECToR адаптирована в модель RuGECToR для ИГО в русскоязычных текстах с использованием сгенерированных синтетических данных и данных из открытых источников. Данная модель не требует большого количества данных для обучения, в отличие от моделей на основе архитектуры Seq2Seq, и показывает конкурентноспособное качество для английского языка. Целью данного исследования является обучение модели, способной обобщать морфологические свойства русского языка, а не подстраиваться под обучающую выборку.

2. ПОСТАНОВКА ЗАДАЧИ ИСПРАВЛЕНИЯ ГРАММАТИЧЕСКИХ ОШИБОК

Задано множество пар, в котором каждая пара состоит из предложения s_i и соответствующего ему эталонного исправления t_i:

$$S = \left\{\left(s_i, t_i\right)\right\}_{i=1}^{N},$$

где N – количество пар. В данной работе предполагается, что каждое предложение представлено в виде последовательности токенов $s_i = \left\{x_1, x_2, \ldots, x_{n_i}\right\}$, а каждое эталонное исправление – в виде последовательности корректирующих правил $t_i = \left\{y_1, y_2, \ldots, y_{n_i}\right\}$, где n_i – длина i-го предложения. Под понятием “токен” подразумевается слово, таким образом разбиение предложения на токены происходит на уровне слов. Каждое корректирующее правило является элементом заданного нами словаря: $y_j \in t_i$, $j = 1, \ldots, n_i$. Составление словаря описано ниже.

Целью задачи является нахождение функции T для построения отображения из последовательности токенов s_i в последовательность корректирующих правил t_i:

$$T : s_i \mapsto t_i \in \left\{0, 1, \ldots, k\right\}^{n_i}$$

где k – размер словаря корректирующих правил.

3. ПРЕДЛАГАЕМЫЙ ПОДХОД ДЛЯ РЕШЕНИЯ ЗАДАЧИ ИСПРАВЛЕНИЯ ГРАММАТИЧЕСКИХ ОШИБОК

На вход модели GECToR подается предложение, которое требуется исправить. Далее предложение разбивается на последовательность

токенов. Таким образом, задача ИГО сводится к нахождению отображения T для сопоставления каждого токена с соответствующим правилом из словаря корректирующих правил. Для русского языка построен словарь, состоящий из 5183 правил, которые будут подробно описаны ниже.

Данная постановка задачи имеет недостаток: каждый токен сопоставляется только с одним правилом из словаря, но иногда для исправления ошибки требуется большее количество изменений. Для решения данной проблемы предлагается использовать итеративное исправление: предложение, полученное в результате работы модели, снова подается модели на вход. Таким образом, модель предсказывает правила для новой последовательности токенов. Правила разработаны таким образом, что последовательность токенов с ошибками может быть преобразована в последовательность токенов без ошибок за конечное число итераций. Проделаны следующие адаптации модели GECToR для русского языка:

1) проведено обучение модели RuGECToR в два этапа:
   a) обучение на синтетических данных с ошибками;
   b) дообучение на комбинации из синтетических данных с ошибками и без ошибок;
2) изменен этап применения модели для исправления ошибок с использованием библиотеки pymorphy2 [18];
3) составлен словарь правил для исправления ошибок;
4) в качестве кодировщика использован Multilingual BERT [19].

4. ОПИСАНИЕ НАБОРА ДАННЫХ

Для генерации синтетических ошибок на первом этапе обучения взяты предложения из русскоязычной Википедии [20] и школьных сочинений [21]. Для второго этапа обучения использованы школьные сочинения [21] и литературные тексты [22]. Сгенерированы специализированные ошибки для следующих частей речи: глагол, имя прилагательное, имя существительное, местоимение, причастие и числительное. Перед началом процесса генерации ошибок пронумерованы позиции токенов для указанных частей речи во всех предложениях.

Процесс генерации ошибок осуществляется следующим образом:

1) случайным образом выбирается предложение, где для каждого токена случайным образом генерируется ошибка, соответствующая части речи токена;
2) после этого каждому такому токену ставится в соответствие правило из словаря корректирующих правил, которое необходимо применить для исправления ошибки.

Распределение ошибок близко к равномерному. В табл. 1 показан пример генерации ошибок в предложении.

В табл. 2 показано, что для первого этапа обучения использовалось 10.000.000 предложений, где каждое предложение содержит произвольные типы ошибок кроме пунктуационных. Для второго этапа обучения мы использовали 1.000.000 предложений: 500.000 предложений не содержат ошибок; 250.000 предложений содержат все ошибки, кроме пунктуационных; оставшиеся 250.000 предложений содержат только пунктуационные ошибки. Для проведения второго этапа обучения добавлены литературные произведения и исключена Википедия с целью сделать модель более устойчивой к различиям между наборами данных [23]. В качестве тестовых наборов данных использовались школьные сочинения и тестовое подмножество набора данных RULEC.

В работе [15] авторы разделяют правила, применяемые к исходным токенам $\{x_1, x_2, \ldots, x_{n_i}\}$ для получения целевого предложения, на два типа – базовые и грамматические. В нашем исследовании сохранено данное разделение. Грамматические правила подобраны таким образом, чтобы словарь корректирующих правил покрывал множество правил русского языка. Базовые правила выполняют наиболее распространенные операции редактирования на уровне токенов: сохранение текущего токена x_i без изменений – правило $KEEP, удаление текущего токена x_i – правило $DELETE, добавление нового

Таблица 1. Пример генерации ошибок в предложении

Исходное предложение	Предложение с ошибками	Корректирующие правила
'Человеческий', **'дух'**, 'непостижимо', 'могуч', ',', 'и', 'убить', 'его', 'в', **'человеке'**, 'почти', 'невозможно', '.'	**'Человеческий-дух'**, 'непостижимо', 'могуч', ',', 'и', 'убить', 'его', 'в', **'чел', 'овеке'**, 'почти', 'невозможно', '.'	**$TRANSFORM_SPLIT_HYPHEN**, $KEEP, $KEEP, $KEEP, $KEEP, $KEEP, $KEEP, $KEEP, **$MERGE_SPACE**, $KEEP, $KEEP, $KEEP

Таблица 2. Информация об обучающем наборе данных

Набор данных	Число предложений	Доля предложений с ошибками	Этап обучения
Википедия + сочинения	10.000.000	≈ 100%	I
Лит. произведения + сочинения	1.000.000	≈ 50%	II

токена t_1 после текущего токена x_i – правило $APPEND\_t_1$ или замена текущего токена x_i на другой токен t_2 – правило $REPLACE\_t_2$. Для генерации правил $APPEND и $REPLACE использованы 2500 наиболее употребляемых русских слов с точки зрения коэффициента Жуайна [24]. Для каждого слова w_i добавлены соответствующие правила $APPEND_w_i и $REPLACE_w_i.

Грамматические правила выполняют операции, специфичные для конкретного случая. Для русского языка использованы следующие правила, которые применяются непосредственно к токену: изменение времени, падежа, рода, лица и числа. Добавлены правила для часто допускаемых ошибок, например, правописание “ться”/“тся” и “при”/“пре”. Также использованы правила, которые не зависят от языка: объединение двух слов с помощью пробела или дефиса; разделение слова, написанного через дефис, на две части; изменение регистра первой буквы слова.

Рассмотрим пример корректирующего правила для исправления “красивая” на “красивый”. Таким образом, необходимо преобразовать прилагательное из женского рода в мужской. Изначально правила для грамматических преобразований создавались следующим образом:

$TRANSFORM_ADJF_GEND_femn_masc

Такие правила содержали подробную информацию:

1) о части речи слова;
2) грамматическом признаке, который необходимо изменить;
3) граммеме как для исходного, так и для исправленного слова.

Далее было решено объединить правила для разных частей речи, исключив название части речи. Кроме того, решено отказаться от указания граммемы для исходного (неправильного) слова, поскольку для его исправления необходима информация только о новой граммеме. В данном исследовании предполагается, что модель сама научится сопоставлять правила соответственно частям речи. Например, глагол и прилагательное могут менять род, а существительное – нет. Таким образом, итоговые правила, используемые в нашей модели, выглядят следующим образом:

$TRANSFORM_GEND_masc

Данная модификация была сделана для того, чтобы уменьшить размер словаря и увеличить количество примеров для каждого правила в обучающем наборе данных. Таким образом достигается компромисс между обобщающей способностью и размером модели, так как с ростом количества правил увеличивается размер слоя классификации.

5. ЭКСПЕРИМЕНТЫ

В качестве предобученного кодировщика на основе архитектуры трансформер выбран Multilingual BERT. Обучение модели проводилось в течение двух этапов, каждый из которых длился 50 эпох. На первом этапе обучения размер батча равен 32, на втором – 16. В качестве оптимизатора параметров выбран Adam [25] с параметром $\mathrm{lr} = 10^{-5}$.

5.1. Примеры исправления предложений

В табл. 3 приведены примеры исправления ошибок моделью RuGECToR в предложениях, составленных человеком. Это результат применения соответствующих правил к входным токенам $\{x_1, x_2, \ldots, x_{n_i}\}$ в течение одной итерации. В результате работы модели слово “дораспределится” было исправлено в соответствии с правилами русского языка, хотя данное слово не было представлено в обучающем наборе данных.

5.2. Примеры итеративного исправления

В табл. 4 представлен пример работы итеративного исправления. Модель предсказывает правило $TRANSFORM_ПРЕ_ПРИ во время первой итерации и правило $APPEND_ли во время второй итерации. Хоть предложение уже после первой итерации является безошибочным, с частицей “ли” оно звучит лучше.

5.3. Примеры исправления реальных сочинений

В данном подразделе исследован результат работы модели RuGECToR на реальных сочинениях. Рассмотрены только те предложения,

Таблица 3. Примеры исправления предложений моделью RuGECToR

Корректирующее правило	Пример предложения
$KEEP	Я хочу играться
$TRANSFORM_ТЬСЯ/ТСЯ	Я хочу (игратся → играться)
$KEEP	Мы не успели дораспределиться
$TRANSFORM ТЬСЯ/ТСЯ	Мы не успели (дораспределится → дораспределиться)
$TRANSFORM_ПРЕ/ПРИ, $APPEND_ли	Можешь (ли) (приобразовать → преобразовать) это выражение?
$MERGE_SPACE	(Рас скажи → Расскажи) о себе
$TRANSFORM_GEND_femn	Она очень (красивый → красивая)
$MERGE_HYPHEN	(Красно зеленый → Красно-зеленый) цвет
$KEEP	Красный, зеленый цвета
$DELETE	Гектор (плохо → ∅) работает
$KEEP	Гектор отлично работает

Таблица 4. Пример итеративного исправления

#итерация	Исправление предложения	Позиция исправляемого токена
Исходное предложение	Можешь приобразовать это выражение?	0
Первая итерация	Можешь преобразовать это выражение?	2
Вторая итерация	Можешь ли преобразовать это выражение?	1

в которых модель обнаружила хотя бы одну ошибку. В табл. 5 приведены примеры таких предложений. Модель исправляет ошибки неидеально, но она обучалась на синтетически сгенерированных данных и тем не менее способна находить ошибки в реальных предложениях.

6. МЕТРИКИ КАЧЕСТВА

Сравнение качества работы моделей проводилось на синтетических и реальных данных. Для оценки качества использованы метрики, описанные в [11]:

$$R = \frac{\sum_{i=1}^{N} |g_i \cap e_i|}{\sum_{i=1}^{N} |g_i|}, \quad P = \frac{\sum_{i=1}^{N} |g_i \cap e_i|}{\sum_{i=1}^{N} |e_i|},$$

$$F_{0.5} = \frac{\left(1 + 0.5^2\right) \cdot R \cdot P}{R + 0.5^2 \cdot P},$$

где N – количество предложений; e_i – множество исправлений, предсказанных нашей моделью для

Таблица 5. Примеры исправления реальных сочинений

Исходное предложение	Исправленное предложение
Таким образом любовь к Родине крайне положительно влияет на человека дарит вдохновение в творчестве и **силы** в борьбе.	Таким образом любовь к Родине крайне положительно влияет на **жизнь** человека дарит вдохновение в творчестве и **сил** в борьбе.
В этом случае межнациональный конфликт подразумевает конфликт между этническими общностями, обычно проживающих поблизости государствах.	В этом случае межнациональный конфликт подразумевает конфликт между этническими общностями, обычно проживающих поблизости **в** государствах.
Автор повествует о бедной семье, в которой **пока** единственный ребенок – старший сын.	Автор повествует о бедной семье, в которой единственный ребенок – старший сын.
Этот случай показывает, что для родителей не может быть счастья большего, чем радость ребенка, а исполнение мечт и **вера** во что-то чудесное, не позволяет оксвернить несчастьям и ненастью светлую душу ребенка.	Этот случай показывает, что для родителей не может быть счастья большего, чем радость ребенка, а исполнение мечт и **веры** во что-то чудесное, не позволяет оксвернить несчастьям и ненастью светлую душу ребенка.

предложения s_i, а g_i – множество эталонных исправлений. Пересечение между g_i и e_i определяется как:

$$g_i \cap e_i = \{e \in e_i \mid \exists g \in g_i : e = g\}.$$

6.1. Синтетические данные

Для синтетического тестового набора данных сгенерировано 10 тыс. предложений с ошибками. Результаты на синтетическом наборе данных приведены в табл. 6. Видно, что метрики R и $F_{0.5}$ на втором этапе обучения ниже, чем на первом. Это можно объяснить двумя причинами:

1. 50% предложений в данных, использованных для второго этапа обучения, не содержали ошибок. Точность стала выше, но при этом количество охватываемых токенов уменьшилось.

2. На втором этапе обучения были использованы дополнительные предложения из литературных произведений, которые не использовались во время первого этапа обучения. Таким образом, была увеличена обобщающая способность модели за счет добавления данных из другого распределения.

6.2. Реальные данные

В качестве реальных тестовых данных использован набор данных RULEC. Результаты приведены в табл. 7. Модель достигает значения равного 22.2 с точки зрения метрики $F_{0.5}$. Это значение выше результата, демонстрируемого базовым подходом набора данных RULEC несмотря на то, что наша модель на нем не обучалась. Таким образом, модель обладает хорошей обобщающей способностью и не переобучается под конкретный набор данных. Следует также отметить, что на синтетических тестовых данных наша модель работает лучше, чем на реальных данных. Это вполне ожидаемо, так как синтетический обучающий и синтетический тестовый наборы данных имеют схожие распределения, в то время как RULEC сильно отличается от синтетического обучающего набора данных. Табл. 7 также показывает, что обобщающая способность модели на втором этапе обучения растет. На наборе данных RULEC качество работы модели после второго этапа значительно выше, чем после первого.

Таблица 6. Качество работы модели на синтетическом тестовом наборе данных

Модель	Этап обучения	P	R	$F_{0.5}$
RuGECToR	I	88.4	67.1	83.1
RuGECToR	II	88.5	65.1	82.5

Таблица 7. Сравнение качества работы моделей на наборе данных RULEC

Модель	Данные для обучения	P	R	$F_{0.5}$
Classifiers (learner)	RULEC	22.6	4.8	12.9
Classifiers (minimal sup.)	RULEC	38.0	7.5	21.0
MT	RULEC	30.6	2.9	10.6
RuGECToR	синтетические (I этап)	23.6	5.6	14.3
RuGECToR	синтетические (II этап)	**40.8**	**7.9**	**22.2**

7. ЗАКЛЮЧЕНИЕ

Проблема исправления грамматических ошибок хорошо изучена для английского языка, но недостаточно изучена для морфологически богатых языков из-за сложности исправления ошибок и малого количества размеченных данных. В данном исследовании представлена эффективная и интерпретируемая модель исправления грамматических ошибок для русского языка. Для этого составлен словарь с корректирующими правилами для русского языка и сгенерирован собственный синтетический набор данных, состоящий из предложений с ошибками. После этого изменен этап применения модели для использования этих правил. В качестве архитектуры модели взята наиболее эффективная Sequence Tagging модель для английского языка – GECToR. Наша модель достигла значений 82.5 на синтетических данных и 22.2 на реальных данных с точки зрения метрики $F_{0.5}$. Модель превосходит базовую модель для набора данных, на котором она не обучалась.

В дальнейшей работе мы хотим расширить словарь корректирующих правил и провести настройку параметров модели на других наборах данных. Также направлением дальнейших исследований является использование различных кодировщиков на основе архитектуры трансформер для русского языка и их ансамблирование.

СПИСОК ЛИТЕРАТУРЫ

1. *Rozovskaya A., Roth D.* Grammatical error correction: Machine translation and classifiers // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, 2016. P. 2205–2215.

2. *Yuan Z., Stahlberg F., Rei M., Byrne B., Yannakoudakis H.* Neural and FST-based approaches to grammatical error correction // Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. Florence, Italy: Association for Computational Linguistics, Aug. 2019. P. 228–239. [Online]. https://aclanthology.org/W19-4424
3. *Bryant C., Ng H.T.* How far are we from fully automatic high quality grammatical error correction? // ACL, 2015.
4. *Rajput D.* Review on recent developments in frequent itemset based document clustering, its research trends and applications // Int. J. Data Anal. Tech. Strateg. 2019. V. 11. P. 176–195.
5. *Flickinger D., Yu J.* Toward more precision in correction of grammatical errors // Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task. Sofia, Bulgaria: Association for Computational Linguistics, 2013. P. 68–73.
6. *Yuan X., Pham D., Davidson S., Yu Z.* ErAConD: Error annotated conversational dialog dataset for grammatical error correction // Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Seattle, United States: Association for Computational Linguistics, 2022. P. 76–84.
7. *Sie Yuen Lee J., Seneff S.* An analysis of grammatical errors in non-native speech in English // 2008 IEEE Spoken Language Technology Workshop, 2008. P. 89–92.
8. *Zhuravlev K., Rudakov K., Inyakin A., Kirsanov A., Lisitsa A., Nikitov G., Peskov N., Yaminov R., Chekhovich Y.* The system of recognition of intellectual text reuse "antiplagiat" // Mathematical methods of pattern recognition: 12th All-Russian conference: Collection of reports. MAKS Press, 2005. P. 329–332.
9. *Keck C.M.* How do university students attempt to avoid plagiarism? A grammatical analysis of undergraduate paraphrasing strategies // Writing & Pedagogy. 2010. V. 2. P. 193–222.
10. *Zhang W.E., Sheng Q.Z., Alhazmi A., Li C.* Adversarial attacks on deep learning models in natural language processing: A survey // ACM Trans. Intell. Syst. Technol., 2020. V. 11. № 3.
11. *Ng H.T., Wu S.M., Briscoe T., Hadiwinoto C., Susanto R.H., Bryant C.* The CoNLL-2014 shared task on grammatical error correction // Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014. P. 1–14. [Online]. https://aclanthology.org/W14–1701.
12. *Rozovskaya A., Roth D.* Grammar error correction in morphologically rich languages: The case of Russian // Transactions of the Association for Computational Linguistics. 2019. V. 7. P. 1–17.
13. *Rothe S., Mallinson J., Malmi E., Krause S., Severyn A.* A simple recipe for multilingual grammatical error correction // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Online: Association for Computational Linguistics. Aug. 2021. P. 702–707.
14. *Grundkiewicz R., Junczys-Dowmunt M., Heafield K.* Neural grammatical error correction systems with unsupervised pre-training on synthetic data // Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. Florence, Italy: Association for Computational Linguistics, 2019. P. 252–263.
15. *Omelianchuk K., Atrasevych V., Chernodub A., Skurzhanskyi O.* GECToR – grammatical error correction: Tag, not rewrite // Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications. Seattle, WA, USA. Online: Association for Computational Linguistics, 2020. P. 163–170.
16. *Malmi E., Krause S., Rothe S., Mirylenka D., Severyn A.* Encode, tag, realize: High-precision text editing // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019. P. 5054–5065.
17. *Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I.* Attention is all you need // Advances in Neural Information Processing Systems, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017. V. 30. Curran Associates, Inc.
18. *Korobov M.* Morphological analyzer and generator for russian and ukrainian languages // Analysis of Images, Social Networks and Texts, ser. Communications in Computer and Information Science, M. Khachay, N. Konstantinova, A. Panchenko, D. Ignatov, and V. Labunets, Eds. Springer International Publishing. 2015. V. 542. P. 320–332.
19. *Devlin J., Chang M.-W., Lee K., Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Volume 1 (Long and Short Papers), 2019. P. 4171–4186. Minneapolis, Minnesota. Association for Computational Linguistics.
20. *Vrandečić D., Krötzsch M.* Wikidata: A free collaborative knowledgebase // Communications of the ACM. Sep 2014. V. 57. № 10. P. 78–85.
21. Open source collection of school essays. https://www.kritika24.ru, accessed: 07.11.2022.
22. Open source collection of literary works. https://proza.ru, accessed: 07.11.2022.

23. *Trinh V.A., Rozovskaya A.* New dataset and strong baselines for the grammatical error correction of Russian // Findings of the Association for Computational Linguistics: ACL–IJCNLP 2021. Online: Association for Computational Linguistics, 2021. P. 4103–4111.
24. *Lyashevskaya O., Sharov S.* Frequency Dictionary of the Modern Russian Language (based on the materials of the National Corpus of the Russian Language) [in Russian]. M.: Azbukovnik, 2009.
25. *Kingma D.P., Ba J.* Adam: A method for stochastic optimization, 2017.

RUGECTOR: RULE-BASED NEURAL NETWORK MODEL FOR RUSSIAN LANGUAGE GRAMMATICAL ERROR CORRECTION

© 2024 I. A. Khabutdinov[a, b], A. V. Chashchin[b], A. V. Grabovoy[a, b], A. S. Kildyakov[b], Y. V. Chekhovich[b, c]

[a] *Moscow Institute of Physics and Technology (National Research University) Institutskiy per. 9, Dolgoprudny, Moscow Region, 141701 Russia*

[b] *Antiplagiat Company Varshavskoe highway 33, Moscow, 117105 Russia*

[c] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Vavilova st. 44-2, Moscow, 119333 Russia.*

Grammatical Error Correction is one of the core Natural Language Processing tasks. At the moment, the open source state-of-the-art Sequence Tagger for English is GECToR model. For the Russian language, this problem does not have solutions with the same good results due to the lack of labeled datasets, therefore we decided to contribute to the aforementioned task. In this research, we described the process of creating a synthetic dataset and training a model on it. We adapt GECToR architecture for Russian language and call it RuGECToR. We use this architecture because, unlike Sequence-to-Sequence approach, it is easy to interpret and does not require a lot of training data. Our goal was to train the model in such a way that it does not adapt to a specific sample, but generalizes the morphological properties of the language. The presented model achieves an $F_{0.5}$ of 82.5 on synthetic data and an $F_{0.5}$ of 22.2 on RULEC dataset which was not in the training set.

Keywords: natural language processing, grammatical error correction, sequence tagging, datasets creation, fine-tuning