ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 35 Выпуск 5

ISSN Online 2220-6426
Volume 35 Issue 5

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2023

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Редколлегия

Главный редактор - Аветисян Арутюн Ишханович, академик РАН, доктор физико-математических наук, профессор, ИСП РАН (Москва, Российская Федерация)

Заместитель главного редактора – Карпов Леонид Евгеньевич, д.т.н., ИСП РАН (Москва, Российская Федерация)

Члены редколлегии

Воронков Андрей Анатольевич, доктор физико-математических наук, профессор, Университет Манчестера (Манчестер, Великобритания)
Вирбицкайте Ирина Бонавентуровна, профессор, доктор физико-математических наук, Институт систем информатики им. академика А.П. Ершова СО РАН (Новосибирск, Россия)
Коннов Игорь Владимирович, кандидат физико-математических наук, Технический университет Вены (Вена, Австрия)
Ластовецкий Алексей Леонидович, доктор физико-математических наук, профессор, Университет Дублина (Дублин, Ирландия)
Ломазова Ирина Александровна, доктор физико-математических наук, профессор, Национальный исследовательский университет «Высшая школа экономики» (Москва, Российская Федерация)
Новиков Борис Асенович, доктор физико-математических наук, профессор, Санкт-Петербургский государственный университет (Санкт-Петербург, Россия)
Петренко Александр Федорович, доктор наук, Исследовательский институт Монреаля (Монреаль, Канада)
Черных Андрей, доктор физико-математических наук, профессор, Научно-исследовательский центр CICESE (Энсенада, Баха Калифорния, Мексика)
Шустер Ассаф, доктор физико-математических наук, профессор, Технион — Израильский технологический институтTechnion (Хайфа, Израиль)

Editorial Board

Editor-in-Chief - Arutyun I. Avetisyan, Academician of RAS, Dr. Sci. (Phys.–Math.), Professor, Ivannikov Institute for System Programming of the RAS (Moscow, Russian Federation)

Deputy Editor-in-Chief – Leonid E. Karpov, Dr. Sci. (Eng.), Ivannikov Institute for System Programming of the RAS (Moscow, Russian Federation)

Editorial Members

Igor Konnov, PhD (Phys.–Math.), Vienna University of Technology (Vienna, Austria)
Alexey Lastovetsky, Dr. Sci. (Phys.–Math.), Professor, UCD School of Computer Science and Informatics (Dublin, Ireland)
Irina A. Lomazova, Dr. Sci. (Phys.–Math.), Professor, National Research University Higher School of Economics (Moscow, Russian Federation)
Boris A. Novikov, Dr. Sci. (Phys.–Math.), Professor, St. Petersburg University (St. Petersburg, Russian Federation)
Alexandre F. Petrenko, PhD, Computer Research Institute of Montreal (Montreal, Canada)
Assaf Schuster, Ph.D., Professor, Technion - Israel Institute of Technology (Haifa, Israel)
Andrei Tchernykh, Dr. Sci., Professor, CICESE Research Centre (Ensenada, Baja California, Mexico).
Irina B. Virbitskaite, Dr. Sci. (Phys.–Math.), The A.P. Ershov Institute of Informatics Systems, Siberian Branch of the RAS (Novosibirsk, Russian Federation)
Andrey Voronkov, Dr. Sci. (Phys.–Math.), Professor, University of Manchester (Manchester, United Kingdom)

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Сайт: http://www.ispras.ru/proceedings/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: info-isp@ispras.ru
Web: http://www.ispras.ru/en/proceedings

© Институт Системного Программирования им. В.П. Иванникова РАН, 2023

Содержание

Table of Contents

DOI: 10.15514/ISPRAS-2023-35(5)-1

Результаты переработки уровней ролевого управления доступом и мандатного контроля целостности формальной модели управления доступом ОС Astra Linux

П.Н. Девянин, ORCID: 0000-0003-2561-794X <*pdevyanin@astralinux.ru*>
ООО «РусБИТех-Астра»,
117105, г. Москва, Варшавское ш., д. 26, стр.11

Аннотация. Механизм управления доступом является базовым для обеспечения безопасности системного программного обеспечения (ПО) такого, как операционная система (ОС) или система управления базами данных (СУБД). В качестве научной основы реализации такого механизма, а также в соответствии с требованиями нормативных документов отечественных регуляторов к сертифицированным средствам защиты информации должна разрабатываться соответствующая критериям ГОСТ Р 59453.1-2021 формальная модель управления доступом. Такой формальной моделью для сертифицированной по высшим классам защиты и уровням доверия ОС Astra Linux является мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель). С учетом внедрения в механизм управления доступом ОС Astra Linux новых элементов, с целью обеспечения более точного соответствия описания модели этому механизму, развитию научно обоснованных технологий и практик разработки и верификации формальных моделей МРОСЛ ДП-модель регулярно перерабатывается. В настоящее время завершена очередная такая переработка модели для двух уровней ее иерархического представления, соответствующих ролевому управлению доступом (представляющему традиционное для ОС семейства Linux дискреционное управление доступом) и мандатному контролю целостности, отражающая наиболее существенные изменения в ОС Astra Linux релиза 2023 года. В статье анализируются основные результаты этой переработки, в рамках которой: введены функции, задающие новые метки сущностей, изменены состав и описания де-юре правил преобразования состояний системы, административных и запрещающих ролей, скорректированы формулировки и заново доказаны несколько утверждений, а также внесены другие изменения в описание модели.

Ключевые слова: формальная модель управления доступом; МРОСЛ ДП-модель; ролевое управление доступом; мандатный контроль целостности; операционная система; Astra Linux.

Для цитирования: Девянин П.Н. Результаты переработки уровней ролевого управления доступом и мандатного контроля целостности формальной модели управления доступом ОС Astra Linux. Труды ИСП РАН, том 35, вып. 5, 2023, стр. 7–22. DOI: 10.15514/ISPRAS–2023–35 (5)–1.

The Results of Reworking the Levels of Role-Based Access Control and Mandatory Integrity Control of the Formal Model of Access Control in Astra Linux

P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>
RusBITech-Astra
26, Varshavskoe, Moscow, 117105, Russia.

Abstract. The access control mechanism is the basis for ensuring the security of system software (OS or DBMS). In accordance with the requirements of regulatory documents of domestic regulators for certified information security tools, as a scientific basis for the implementation of such a mechanism, a formal access control model that meets the GOST R 59453.1-2021 criteria should be developed. Such a formal model for the Astra Linux operating system certified for the highest protection classes and assurance levels is the mandatory entity-role model of access and information flows security control in OS of Linux family (MROSL DP-model). Taking into account the introduction of new elements into the access control mechanism of the Astra Linux and in order to ensure a more accurate correspondence of the model description to this mechanism, the development of scientifically based technologies and practices for the development and verification of formal models, the MROSL DP-model is regularly revised. Another such revision of the model now has been completed for two levels of its hierarchical representation, corresponding to role-based access control (representing discretionary access control, traditional for the OS of Linux family) and mandatory integrity control, reflecting the most significant changes in the Astra Linux release 2023. The article analyzes the main results of this revision, within which: functions that define new entity labels are introduced, the composition and descriptions of the de-jure rules for transforming system states, administrative and negative roles are changed, the wording is corrected and several statements are re-proved, and other changes are made in the model description.

Keywords: formal models of access control; MROSL DP-model; role-based access control; mandatory integrity control; operating system; Astra Linux.

For citation: Devyanin P.N. The results of reworking the levels of role-based access control and mandatory integrity control of the formal model of access control in Astra Linux. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 7-22 (in Russian). DOI: 10.15514/ISPRAS-2022-35(5)-1.

1. Введение

Системное программное обеспечение (ПО) такое, как операционные системы (ОС) или системы управления базами данных (СУБД), часто является основой программных средств защиты информации, одна из ключевых функций которых – это организация безопасного управления доступом субъектов доступа (процессов) к объектам доступа (сущностям, каталогам, файлам, сокетам, базам данных, таблицам, записям и др.). Для этого в такие средства включают механизм управления доступом, реализующий политики управления доступом [1]. В большинстве случаев (например, во многих ОС) это политика дискреционного управления доступом, а также (как, например, в ОС семейства Microsoft Windows) мандатного контроля целостности, реже – мандатного управления доступом. В СУБД, как правило, используется политика ролевого управления доступом.

В качестве научной основы реализации механизма управления доступом в соответствии с требованиями нормативных документов отечественных регуляторов к сертифицированным средствам защиты информации [2], в том числе к разработке для них безопасного ПО [3] должна использоваться формальная модель управления доступом [4, 5]. При этом такая модель должна соответствовать критериям, изложенным в ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения» [1], и быть верифицирована с применением инструментальных средств согласно рекомендациям ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом» [6].

Всем требованиям и критериям перечисленных национальных стандартов соответствует мандатная сущностно-ролевая ДП-модель управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модель) [5], на основе которой «Группой Астра» (в ее состав входит ООО «РусБИТех-Астра») в подсистеме безопасности PARSEC реализуется механизм управления доступом сертифицированной по высшим классам защиты и уровням доверия ОС Astra Linux (операционной системы специального назначения Astra Linux Special Edition) [7, 8]. В модели объединены мандатное управление доступом, мандатный контроль целостности и ролевое управление доступом, выражающее традиционное для ОС семейства Linux дискреционное управление доступом. При этом все три политики сочетаются в модели с учетом их свойств, не противореча друг другу.

Разработка, верификация и внедрение МРОСЛ ДП-модели в ОС Astra Linux ведется «Группой Астра» в рамках реализации единой методологии разработки безопасного системного ПО [9]. Для этого МРОСЛ ДП-модель формируется в двух нотациях: математической (аналогично классическим моделям) [5] и формализованной на языке формального метода Event-B [10]. Модель в первой нотации предоставляет возможность объективного анализа ее корректности любым специалистом в области информационной безопасности, владеющим языком математики. В формализованной нотации модель верифицируется с применением инструментальных средств дедуктивно (с помощью инструментального средства Rodin) и по методу проверки моделей (с помощью инструментального средства ProB) [11].

В обеих нотациях МРОСЛ ДП-модель имеет иерархическое представление, позволяющее описывать ее по уровням (слоям). При этом каждый нижний уровень модели представляет абстрактную систему, элементы которой не зависят от новых элементов, принадлежащих более высокому уровню модели, который, в свою очередь, наследует, а при необходимости корректирует или дополняет элементы нижнего уровня. В целом иерархическое представление модели состоит из восьми уровней – четырех уровней для моделирования управления доступом непосредственно в ОС (ролевого управления доступом, мандатного контроля целостности, мандатного управления доступом с информационными потоками по памяти, мандатного управления доступом с информационными потоками по времени) и четырех уровней для решения аналогичной задачи в штатной для ОС СУБД PostgreSQL, что обеспечивает согласованность механизмов управления доступом в ОС и СУБД. Такое разделение МРОСЛ ДП-модели на уровни соответствует подходу по ее реализации непосредственно в программном коде подсистемы безопасности PARSEC ОС Astra Linux.

Поскольку подсистема безопасности PARSEC регулярно модифицируется, то с целью обеспечения более точного соответствия описания модели этому механизму, развития научно обоснованных технологий и практик разработки и верификации формальных моделей МРОСЛ ДП-модель также регулярно перерабатывается. В настоящее время завершена очередная такая переработка модели для первых двух очень важных для реализации в ОС уровней ее иерархического представления: ролевого управления доступом и мандатного контроля целостности. Под влиянием выполненных в рамках модели теоретических исследований именно эти два вида управления доступом претерпели наиболее существенные изменения в ОС Astra Linux релиза 2023 г. Кроме того, был накоплен опыт верификации реализации модели в подсистеме безопасности PARSEC, также нашедший отражение в описании модели. В связи с этим в статье анализируются основные результаты соответствующей переработки модели в ее математической нотации, которая также отвечает рекомендациям, изложенным в разрабатываемом при участии специалистов «Группы Астра» проекте нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке».

Статья организована следующим образом. В следующем разделе рассматриваются основные элементы МРОСЛ ДП-модели, используемые для описания моделируемой системы – ОС Astra Linux. В разд. 3 анализируются изменения в описании состояний системы, включая

корректировки множеств административных и запрещающих ролей, введенные функции, задающие новые метки сущностей (объектов доступа, файлов или каталогов). В разд. 4 излагаются основные изменения в составе де-юре правил преобразования состояний системы и результаты корректировки условий безопасности системы в смысле мандатного контроля целостности. Заключение завершает статью, в нем подводятся итоги текущей переработки модели, а также рассматриваются дальнейшие направления этой работы.

2. Иерархическое представление МРОСЛ ДП-модели

Как уже было отмечено в предыдущем разделе, МРОСЛ ДП-модель имеет иерархическое представление, которое кроме удобства при переработке модели, обеспечения условий для ее развития и внедрения в механизмах защиты ОС Astra Linux, также позволяет использовать для верификации модели технику пошагового уточнения (refinement) Event-B [11-13].

Иерархическое представление модели, соответствующее механизму управления доступом непосредственно в ОС Astra Linux, состоит из следующих четырех уровней (рис. 1):

- Уровень 1.1 (базовый) – модель системы ролевого (дискреционного) управления доступом («корень» иерархического представления, не наследующий ни один из других уровней);
- Уровень 2.1 – модель системы ролевого управления доступом и мандатного контроля целостности;
- Уровень 3.1 – модель системы ролевого управления доступом, мандатного контроля целостности и мандатного управления доступом только с информационными потоками по памяти;
- Уровень 4.1 – модель системы ролевого управления доступом, мандатного контроля целостности и мандатного управления доступом с информационными потоками по памяти и по времени.

Для СУБД PostgreSQL описаны четыре аналогичных уровня (1.2, 2.2, 3.2 и 4.2), что кроме согласованности механизмов управления доступом в ОС и СУБД, также предоставляет возможность анализа безопасности информационных потоков (скрытых каналов) по памяти и по времени между сущностями ОС и СУБД [5]. При этом уровни СУБД наследуют и дополняют как предшествующие уровни для СУБД, так и соответствующие уровни для ОС (например, уровень 2.2 наследует и дополняет уровни 1.2 и 2.1).

Для описания состояний моделируемой системы на уровне 1.1 (ролевого управления доступом) используются следующие основные элементы:

- Множества: учетных записей пользователей, субъектов (процессов), сущностей – объектов (файлов) и контейнеров (каталогов), допустимых имен сущностей, ролей, запрещающих и административных ролей, видов прав доступа, видов доступа, прав доступа к сущностям или субъектам, административных прав доступа к ролям, запрещающим или административным ролям, доступов субъектов к сущностям, административных доступов субъектов к ролям, запрещающим или административным ролям;
- Функции: имен сущностей, ролей, запрещающих ролей или административных ролей, иерархии сущностей, ролей, запрещающих ролей или административных ролей, прав доступа ролей, запрещающих ролей или административных ролей к сущностям, административных прав доступа к ролям, запрещающим или административным ролям запрещающих или административных ролей, необходимого обладания запрещающей ролью, наличия права доступа у текущих ролей, запрещающих или административных ролей субъекта к сущности, роли, запрещающей роли, административной роли или субъекту, доступа субъекта к

сущностям в контейнерах (каталогах), потенциальной доступности авторизованной роли или административной роли учетной записи пользователя, вида метки сущности.

Рис. 1. Схема иерархического представления модели.
Fig. 1. Schema of hierarchical view of model.

Например, функция доступа субъекта к сущностям в контейнерах задается следующим образом (при этом используются обозначения: S — множество субъектов, E — множество сущностей, C — множество сущностей-контейнеров, H_E – функция иерархии сущностей, *check_right* – функция наличия права доступа у текущих ролей, запрещающих или административных ролей субъекта к сущности, роли, запрещающей роли, административной роли или субъекту):

execute_container: $S \times E \to \{true, false\}$ *– функция доступа субъекта к сущностям в контейнерах такая, что по определению для субъекта* $s \in S$ *и сущности* $e \in E$ *справедливо равенство execute_container (s, e) = true тогда и только тогда, когда существует последовательность сущностей* $e_1, \ldots, e_n \in E$, *где* $e = e_n$ *и либо* $n = 1$, *либо* $n \geq 2$ *и выполняются следующие условия:*

Условие 1. *Не существует сущности-контейнера* $e_0 \in C$ *такой, что* $e_1 \in H_E(e_0)$;

Условие 2. *Верно* $e_i \in H_E(e_{i-1})$, *где* $1 < i \leq n$;

Условие 3. *Верно* $check\_right(s, e_i, execute_r) = true$, *где* $1 \leq i < n$.

На уровне 2.1 (мандатного контроля целостности) для описания состояний моделируемой системы дополнительно используются следующие основные элементы:

- Множества: видов информационных потоков, информационных потоков, уровней целостности, учетных записей доверенных пользователей, учетных записей недоверенных пользователей, доверенных субъектов, недоверенных субъектов, сущностей-объектов, параметрически ассоциированных с учетными записями пользователей, сущностей-объектов, параметрически ассоциированных с субъектами, сущностей-объектов, функционально ассоциированных с субъектами, сущностей-объектов, параметрически ассоциированных с ролями, запрещающими ролями или административными ролями, сущностей-объектов, доступы на запись или чтение к которым не могут быть использованы для реализации информационных потоков по памяти;

- Функции: уровней целостности учетных записей пользователей, текущих уровней целостности субъектов, уровней целостности ролей, запрещающих ролей или административных ролей, уровней целостности сущностей, способа получения доступа на запись к сущностям-контейнерам, ролям, запрещающим ролям и административным ролям с учетом или без учета их уровня целостности, порядка получения доступа на чтение или использования права доступа на выполнение к сущностям, ролям, запрещающим ролям и административным ролям с учетом или без учета их уровня целостности, порядка активизации субъекта из сущности-объекта с учетом уровней целостности этого объекта, способа задания уровня целостности при создании в сущностях-контейнерах, ролях, запрещающих ролях и административных ролях с наследованием или без наследования их уровня целостности, де-факто владения субъектами.

Например, так задается решетка уровней целостности и функции уровней целостности (где дополнительно к ранее приведенным обозначениям используются следующие обозначения: *U* — множество учетных записей пользователей, *R, NR, AR* — множества ролей, запрещающих ролей и административных ролей, соответственно):

(LI, ≤) — решетка уровней целостности, при этом заданы минимальный i_low и максимальный i_high элементы решетки, соответственно;
i_u*: U → LI — функция, задающая для каждой учетной записи пользователя ее уровень целостности;*
i_e*: E → LI — функция, задающая уровень целостности для каждой сущности;*
i_r*: R ∪ NR ∪ AR → LI — функция, задающая для каждой роли, запрещающей роли или административной роли ее уровень целостности;*
i_s*: S → LI — функция, задающая для каждого субъекта его текущий уровень целостности.*

В МРОСЛ ДП-модели на уровнях 1.1 и 2.1 ее иерархического представления определены 35 де-юре правил преобразования состояний системы, предназначенных для формального описания (спецификации) следующих основных функций механизма ролевого (дискреционного) управления доступом и мандатного контроля целостности ОС Astra Linux:

- Создание, удаление, переименование, получение или изменение параметров учетных записей пользователей, субъектов, сущностей или «жестких» ссылок на них, ролей, запрещающих ролей, административных ролей;

- Получение доступов субъектов к сущностям, административных доступов к ролям, запрещающим ролям или административным ролям;

- Изменение прав доступа ролей, запрещающих ролей или административных ролей к сущностям, субъектам, ролям, запрещающим ролям или административным ролям;

- Изменение иерархий сущностей, ролей, запрещающих ролей или административных ролей;
- Управление множествами запрещающих ролей для ролей и административных ролей;
- Применение авторизованной роли или административной роли от имени учетной записи пользователя;
- Изменение уровней целостности учетных записей пользователей, сущностей, ролей, запрещающих ролей или административных ролей;
- Задание объектов, параметрически ассоциированных с учетными записями пользователей, ролями или административными ролями.

При этом для каждого де-юре правила задаются его параметры, условия и результаты применения. Например, де-юре правило *set_entity_owner (x, r, r', y)*, используемое для назначения субъектом x вместо роли r новой роли-владельца r' сущности y на уровне 2.1 модели задается согласно табл. 1 (где дополнительно к ранее приведенным обозначениям используются следующие обозначения: $write_a$, $read_a$ – доступы на запись и чтение, соответственно, own_r – право доступа владения, *AA* – множество административных доступов, *root_role* – специальная роль, соответствующая полномочиям суперпользователя *root* в ОС семейства Linux, *PA* – функция прав доступа ролей, H_E – функция иерархии сущностей, *direct* – функция вида метки сущности, *check_i_right* – функция наличия права доступа у текущих ролей, запрещающих или административных ролей субъекта к сущности, роли, запрещающей роли, административной роли или субъекту с учетом мандатного контроля целостности), в которой первая строка соответствует условиям и результатам применения правила, оставшимся без изменения с уровня 1.1 модели:

Табл. 1. Де-юре правило set_entity_owner(x, r, r', y)
Table. 1. De-jure rule set_entity_owner(x, r, r', y)

Условия	**Результаты применения**
$x \in S, r, r' \in R \cup AR, y \in E,$ $\{(x, r', write_a), (x, root\_role, read_a)\} \subseteq AA,$ $direct(y) = true,$ $[((x, r, write_a) \in AA, (y, own_r) \in PA(r))$ *или (для всех* $r'' \in R \cup AR$ *выполняется* $(y, own_r) \notin PA(r''))]$	$PA'(r) = PA(r) \setminus \{(y, own_r)\},$ $PA'(r') = PA(r') \cup \{(y, own_r)\},$ *если* $H_E(y) = \{y' \in H_E(y): direct(y') = false\},$ *то для всех* $y' < y$ *верно* $PA'(r) = PA(r) \setminus \{(y', own_r)\},$ $PA'(r') = PA(r') \cup \{(y', own_r)\}$
$check\_i\_right(x, y, own_r) = true,$ $execute\_i\_container(x, y) = true$	–

Для теоретического анализа условий безопасности в рамках МРОСЛ ДП-модели, начиная с уровня 2.1 ее иерархического представления, аналогично де-юре правилам определены 10 де-факто правил преобразования состояний системы. Они не требуют непосредственной реализации в ОС Astra Linux и предназначены для задания условий создания информационных потоков по памяти и по времени или получения одним субъектом возможности управления другим субъектом.

Кроме того, в МРОСЛ ДП-модели приводятся определение безопасного начального состояния системы (состояния, в котором отсутствуют запрещенные информационные потоки по памяти или по времени, управление субъектами друг другом, а информационные потоки по памяти или доступы к сущностям, параметрически или функционально ассоциированным с субъектами, не нарушают правил мандатного контроля целостности) и определение трех смыслов нарушения безопасности системы:

- На уровне 2.1 – в смысле мандатного контроля целостности, позволяющее недоверенному субъекту с низким уровнем целостности захватить управление (де-факто владение) субъектом с более высоким уровнем целостности;
- На уровне 3.1 – в смысле Белла-ЛаПадулы, результатом которого является создание запрещенного информационного потока по памяти «сверху-вниз»;
- На уровне 4.1 – в смысле контроля информационных потоков по времени – создание запрещенного информационного потока по времени «сверху-вниз» между сущностями.

С использованием этих определений в модели сформулированы и обоснованы достаточные условия безопасности системы во всех трех смыслах. Например, для уровне 2.1 модели дается следующее определение безопасного начального состояния (где дополнительно к ранее приведенным обозначениям используются следующие обозначения: $write_m$ – информационный поток по памяти, N_S – множество недоверенных субъектов, O — множество сущностей-объектов, A – множество доступов субъектов к сущностям, F – множество информационных потоков, $[y]$, $]y[$ – множества сущностей-объектов, соответственно, функционально или параметрически ассоциированных с субъектом y, de_facto_own – функция де-факто владения субъектами):

Определение 1. Начальное состояние G_0 системы назовем безопасным, когда оно удовлетворяет следующим условиям:

Условие 1. Для каждых субъектов $x, y \in S_0$ таких, что $y \in de\_facto\_own_0(x)$, верно $i_{s0}(y) \le i_{s0}(x)$ (текущий уровень целостности субъекта, де-факто владеющего другим субъектом, не меньше текущего уровня целостности этого субъекта).

Условие 2. Для каждых недоверенного субъекта $x \in N_{S0}$, субъекта $y \in S_0$ и сущности-объекта $o \in O_0$ таких, что либо $(o \in [y]$ и $(x, o, write_m) \in F_0)$, либо $(o \in]y[$ и либо $(o, x, write_m) \in F_0$, либо $(x, o, read_a) \in A_0)$, верно неравенство $i_{s0}(y) \le i_{s0}(x)$ (недоверенный субъект может иметь информационные потоки по памяти с участием сущностей-объектов, функционально или параметрически ассоциированных с другим субъектом, или иметь к таким параметрически ассоциированным сущностям-объектам доступ на чтение, только если текущий уровень целостности первого субъекта не меньше текущего уровня целостности второго субъекта).

Условие 3. Для каждого информационного потока $(x, y, write_m) \in F_0$ справедливо $i_{x0}(x) \ge i_{y0}(y)$, где i_{x0} и i_{y0} соответствующие функции i_{e0} или i_{s0} (отсутствуют информационные потоки по памяти от субъектов или сущностей с меньшим уровнем целостности к субъектам или сущностям с большим или несравнимым уровнем целостности).

Далее в статье анализируются результаты переработки уровня 1.1 (ролевого управления доступом) и уровня 2.1 (мандатного контроля целостности) иерархического представления МРОСЛ ДП-модели.

3. Переработка описания состояний системы

Основное внимание при переработке описания состояний системы в рамках МРОСЛ ДП-модели было уделено определению функций, соответствующих в ОС Astra Linux новым меткам файлов и каталогов, позволяющим повысить удобство и гибкость применения мандатного контроля целостности. То есть эти функции были определены на уровне 2.1 модели, всего их четыре:

1. $IRELAX: C \cup R \cup NR \cup AR \to \{true, false\}$ — функция, задающая способ получения доступа на запись к сущностям-контейнерам, ролям, запрещающим ролям или административным ролям с учетом или без учета их уровня целостности;

2. *SSI:* $E \cup R \cup NR \cup AR \rightarrow \{true, false\}$ — функция, задающая порядок получения доступа на чтение или использования права доступа на выполнение к сущностям, ролям, запрещающим ролям или административным ролям с учетом или без учета их уровня целостности;
3. *SILEV:* $O \rightarrow \{true, false\}$ — функция, задающая порядок активизации субъекта из сущности-объекта с учетом уровня целостности этого объекта, уровня целостности родительского субъекта и уровня целостности учетной записи пользователя, от имени которой активизируется субъект;
4. *IINH:* $C \cup R \cup NR \cup AR \rightarrow \{true, false\}$ — функция, определяющая способ задания уровня целостности при создании в сущностях-контейнерах, ролях, запрещающих ролях или административных ролях с наследованием или без наследования их уровня целостности.

При этом в модели говорится, что если значение функции *IRELAX, SSI, SILEV* или *IINH* от сущности или какой-либо роли равно *true*, то эта сущность или роль помечена с помощью соответствующей функции.

Для всех видов ролей функция *IRELAX* задана только для обеспечения единообразия их атрибутов с сущностями-контейнерами (так как для ролей она по определению всегда равна *false*). При этом, если доступ на запись к сущности-контейнеру $c \in C$ разрешен субъекту только с текущим уровнем целостности, большим или равным уровня целостности этой сущности-контейнера, то по определению выполняется равенство *IRELAX(c) = false*, в противном случае выполняется равенство *IRELAX(c) = true*. В ОС такой помеченной с помощью функции *IRELAX* сущностью-контейнером является каталог «*/tmp*», доступ на запись к которому необходимо обеспечить процессам с любым текущим уровнем целостности.

При определении функции *SSI* задано, что если доступ на чтение или использование права доступа на выполнение к сущности или роли $e \in E \cup R \cup NR \cup AR$ разрешен субъекту только с текущим уровнем целостности, большим или равным уровня целостности этой сущности или роли, то по определению выполняется равенство *SSI(e) = true*, в противном случае выполняется равенство *SSI(e) = false*. Включение этой функции в модель связано с тем, что изначально мандатный контроль целостности накладывает ограничения на получение доступа на запись к сущностям, но не на чтение. Вместе с тем, в ОС существуют файлы или каталоги, доступ к которым на чтение (а к каталогам и на выполнение) необходимо разрешить только процессам с текущим уровнем целостности, не меньшим, чем у этих файлов или каталогов. Примерами этого являются некоторые файлы с высоким уровнем целостности из каталогов «*/etc*» и «*/dev*». Аналогично пометка некоторых ролей с помощью функции *SSI* позволяет запретить их получение в качестве текущих субъектами, обладающими текущим уровнем целостности, меньшим или несравнимым, чем у соответствующей роли (раньше такой запрет действовал на все роли). Таким образом, теперь даже обладающие высоким уровнем целостности административные роли можно делать (не помечая их с помощью функции *SSI*) текущими для недоверенных (с низким текущим уровнем целостности) субъектов, что позволит в будущем повысить гибкость ролевого управления доступом.

При активизации субъекта из сущности-объекта, помеченной с помощью функции *SILEV*, уровень целостности учетной записи пользователя, от имени которой он будет функционировать, должен быть не ниже уровня целостности объекта, и текущий уровень целостности субъекта устанавливается равным уровню целостности этого объекта. В ОС это позволяет процессам в сессии учетной записи пользователя с низким уровнем целостности запускать некоторые процессы, которым для выполнения их функций необходим высокий текущий уровень целостности. Примером использования такого помеченного с помощью функции *SILEV* объекта в ОС Astra Linux является исполняемый файл консольной утилиты

passwd, запускаемой для изменения пароля низкоцелостной учетной записи пользователя, образ которого хранится в обладающем высоким уровнем целостности файле «*/ets/shadow*».

Если при создании в сущности-контейнере или в роли любого вида $c \in C \cup R \cup NR \cup AR$ необходимо наследование их уровня целостности (с учетом возможной их пометки с помощью функции *IRELAX*), то по определению выполняется равенство *IINH(c) = true*, в противном случае, когда на создаваемые сущности или роли устанавливается минимальный уровень целостности (*i_low*), выполняется равенство *IINH(c) = false* (для ролей всех видов по определению задано, что все они помечены с помощью функции *IINH*). С использованием этой функции переопределен порядок наследования уровня целостности создаваемой сущности от уровня целостности родительской сущности-контейнера. По умолчанию наследования нет, то есть целостность создаваемой сущности устанавливается на минимальном уровне (*i_low*). Наследование осуществляется, когда родительская сущность-контейнер помечена с помощью функции *IINH*. При этом с учетом опыта реализации мандатного контроля целостности в ОС Astra Linux и с использованием функции *IRELAX* требуется:

- При создании в сущности-контейнере, не помеченной с помощью функции *IRELAX*, новой сущности она наследует уровень целостности от сущности-контейнера;
- Когда сущность-контейнер также помечена с помощью функции *IRELAX*, уровень целостности новой сущности устанавливается равным наибольшей нижней границе (в решетке уровней целостности) значений уровней целостности сущности-контейнера и текущего уровня целостности осуществляющего создание субъекта.

Для удобства использования функции *IINH* в ОС пометка с помощью этой функции наследуется на создаваемые подкаталоги от родительских каталогов, что также отражено в модели.

Кроме включения в МРОСЛ ДП-модель при ее переработке новых функций на уровне 1.1 реализована возможность назначения запрещающим ролям не только прав доступа к сущностям или субъектам, а по сути, запрещающих административных прав доступа к другим ролям или административным ролям, что делает запрещающие роли по назначению близкими к административным ролям. При этом во избежание разного рода взаимных запретов запрещающие роли не могут иметь административных прав доступа к другим запрещающим ролям, специальным административным ролям и ряду других ролей. Кроме того, запрещающие роли могут иметь к ролям или административным ролям только административные права доступа на чтение или запись.

Также существенные изменения при переработке модели коснулись состава ролей. На уровне 1.1 модели для администрирования ролей, запрещающих ролей и административных ролей добавлены специальные административные роли, соответственно, *cap_role, admin_cap_role, negative_cap_role*, назначение которых близко к имеющейся в ОС привилегии *PARSEC_CAP_CAP* (она позволяет давать или отзывать другие привилегии). При этом добавлена специальная административная роль *root_role*, предназначенная для моделирования ситуации использования штатных для ОС семейства Linux полномочий суперпользователя *root*, функционирующим от имени которого процессам разрешено игнорировать основное для этих ОС дискреционное управление доступом. Этой роли на уровне 2.1 модели назначается минимальный уровень целостности (*i_low*), так как изменять дискреционные права доступа (получать к *root_role* доступ на запись) в ОС Astra Linux можно и на низком уровне целостности. Всем остальным специальным административным ролям назначается максимальный уровень целостности (*i_high*), что по-прежнему дает возможность их администрирования только доверенным (обладающим максимальным уровнем целостности) субъектам. Однако только предназначенная для администрирования административных ролей административная роль *admin_cap_role* помечается с помощью

функции *SSI*, что обеспечивает возможность ее получения как текущей только доверенными субъектами.

На уровне 2.1 МРОСЛ ДП-модели для придания большего соответствия технологиям, используемым в ОС Astra Linux, добавляются еще три специальные административные роли. Первая из них – *chmac_role* соответствует привилегии *PARSEC_CAP_CHMAC* и позволяет обладающему ею как текущей субъекту понижать уровни целостности сущностей, когда они не выше текущего уровня целостности этого субъекта. Вторая – *inherit_integrity_role* соответствует привилегии *PARSEC_CAP_INHERIT_INTEGRITY* и обеспечивает субъекту возможность создавать в сущности-контейнере (каталоге) новые сущности с наследованием уровня целостности от этой сущности-контейнера, как если бы он была помечена с помощью функции IINH. Третья – *setmac_role* соответствует привилегии *PARSEC_CAP_SETMAC* и разрешает субъекту активизировать нового субъекта с текущим уровнем целостности меньшим, чем у активизирующего субъекта (по умолчанию их текущие уровни целостности должны быть равными), что будет полезно в ОС Astra Linux для реализации механизма "песочниц" для функционирования недоверенного ПО на низких уровнях целостности.

С учетом опыта исследования ОС Astra Linux по сравнению с предыдущими представлениями модели при ее текущей доработке на уровне 2.1 сделано уточнение, что ассоциированными с учетными записями пользователей, субъектами или ролями могут быть только сущности-объекты (раньше это были любые сущности). Это связано с тем, что до сих пор не было выявлено случаев, когда именно сущности-контейнеры (каталоги) без содержащихся в них сущностей-объектов (файлов) являлись бы ассоциированными параметрически или функционально с учетными записями пользователей, субъектами или ролями. Такой подход позволил уточнить, а в ряде случаев упростить формулировки некоторых положений модели. При этом в определении абсолютной функциональной корректности удалось отказаться от информационных потоков по памяти (их сложно анализировать в реальной ОС) и свести к доступу субъекта на чтение к сущности, т.к. только такой доступ субъекта с большим текущим уровнем целостности к сущности с меньшим уровнем целостности создает потенциальную угрозу возникновения при участии этого субъекта информационного потока по памяти к обладающей высоким уровнем целостности сущности-объекту, функционально ассоциированной с другим субъектом. Однако для параметрической корректности аналогичный результат пока получить не удалось, т.к. для нее существенны в общем случае не запрещаемые мандатным контролем целостности информационные потоки по памяти от обладающих высоким уровнем целостности сущностей-объектов, параметрически ассоциированных с субъектами, к сущностям с низким уровнем целостности.

Внесенные изменения в описания состояний системы в рамках МРОСЛ ДП-модели были учтены при переработке описаний и изменении состава де-юре правил преобразования состояний системы, а также при корректировке условий безопасности системы в смысле мандатного контроля целостности и, как следствие, повторном доказательстве их достаточности. Все перечисленное анализируется в следующем разделе.

4. Изменения в составе де-юре правил преобразования состояний системы и корректировка условий безопасности

В первую очередь изменения в составе и описаниях де-юре правил преобразования состояний системы были направлены на обеспечение их большего соответствия реализованным в подсистеме безопасности PARSEC ОС Astra Linux обрабатывающим системные вызовы функциям. Относящиеся к чисто ролевому управлению доступом де-юре правила менялись меньше, чтобы не увеличивать и так большой объем правок модели (это оставлено на будущее, когда ролевое управление доступом будет непосредственно реализовываться в ОС). В результате уже на уровне 1.1 модели были осуществлены следующие основные изменения:

- Правила *access_write(x, y)* и *access_read(x, y)*, дающие возможность субъекту *x* получить, соответственно, доступ на запись или чтение к сущности или роли *y*, заменены одним соответствующим системному вызову *open* правилом *take_access (x, y, {α_{oj}: 1 ≤ j ≤ k})*, где α_{oj} – предоставляемые доступы на чтение или запись;
- Правила *grant_rights (x, r, y, {α_{rj}: 1 ≤ j ≤ k})* и *remove_rights (x, r, y, {α_{rj}: 1 ≤ j ≤ k})*, позволявшие субъекту *x* дать или забрать у роли *r* права доступа *{α_{rj}: 1 ≤ j ≤ k}* к сущности *y*, заменены одним соответствующим системному вызову *chmod* правилом *set_rights (x, r, y, {α_{rj}: 1 ≤ j ≤ k}, plus, t)*, в зависимости от булева параметра *plus*, добавляющим или удаляющим права доступа у роли *r*. Кроме того, в это правило добавлен булевский параметр *t*, который в случае, когда *y* является сущностью-контейнером (каталогом), указывает, является ли она разделяемой (в ОС такой каталог имеет атрибут *sticky bit*) или нет, что позволило отказаться от отдельного устанавливающего параметр *t* де-юре правила *set_container_attr (x, y, t)*;
- Правила *add_negative_owner (x, nr, y)* и *remove_negative_owner (x, nr, y)*, использованные ранее для назначения субъектом x запрещающей роли-владельца nr сущности y, заменены одним правилом *set_negative_owner (x, nr, y, plus)*, что позволяет моделировать планируемое в будущем развитие функционала системного вызова *chown*.

Также на уровне 1.1 модели путем добавления нового де-юре правила *apply_user_role(x, u, r)* реализована возможность применения субъектом *x* авторизованной роли или административной роли r учетной записи некоторого пользователя u. Такое допустимо в ОС, когда некоторому ее процессу (например, средству виртуализации Libvirt) поступает запрос (в модели этому соответствует использование роли), который надо выполнить от имени некоторой учетной записи пользователя. При этом в правиле проверяется, что субъект *x* может применить роль *r* только при существовании последовательности авторизованных административных (кроме, возможно, последней роли в последовательности) ролей учетной записи пользователя *u*, начинающейся с ее индивидуальной административной роли (для каждой учетной записи пользователя задается такая административная роль), и заканчивающейся *r*, где каждая авторизованная административная роль в последовательности обладает административным правом доступа на чтение к последующей роли, и во множествах запрещающих ролей для ролей последовательности нет ни одной запрещающей роли, обладающей административным правом доступа на чтение хотя бы к одной из ролей последовательности. При этом на уровне 2.1 дополнительно проверяется, во-первых, отсутствие запретов на получения доступов на чтение к ролям последовательности с учетом их пометки с помощью функции *SSI* и отсутствие у этих ролей параметрически ассоциированных объектов. Во-вторых, субъект *x* должен иметь текущий уровень целостности не ниже уровня целостности учетной записи пользователя *u* и доступ на чтение или запись ко всем объектам, параметрически с ней ассоциированных.

Следующей существенной правкой описаний де-юре правил преобразование состояний стало добавление на уровне 2.1 модели в позволяющее менять параметры ролей всех видов правило *set_role_labels (x, r, ri, ssi, ro)* возможности изменения их уровней целостности. Аналогично сущностям понижение уровня целостности роли, запрещающей роли или административной роли разрешено только субъекту, обладающему текущим уровнем целостности не ниже уровня целостности этой роли и обладающему текущей специальной административной ролью *root_role* и текущей специальной административной ролью *cap_role*, *negative_cap_role* или *admin_cap_role*, соответственно. При этом для повышения или назначения несравнимого уровня целостности этой роли или ее пометки с помощью функции *SSI* (кроме запрещающих ролей) дополнительно требуется, чтобы субъект был доверенным (с максимальным текущим уровнем целостности *i_high*) и обладал текущей специальной административной ролью *admin_cap_role*.

Кроме того, на уровне 2.1 описания условий применения де-юре правил, определяющих порядок администрирования уровней целостности сущностей и значений от них функций *IRELAX, SSI, SILEV* и *IINH*, приближены к тому, что для этого сделано или планируется сделать в ОС Astra Linux. Так изменение значений функций *IINH, SSI* или *IRELAX* от сущности (функций *IINH* и *IRELAX* только для сущности-контейнера) разрешено субъекту с текущим уровнем целостности не ниже уровня целостности этой сущности. Понижение уровня целостности сущности разрешено только субъекту, обладающему текущим уровнем целостности не ниже уровня целостности этой сущности и обладающему текущими специальными административными ролями *root_role* и *chmac_role* (как аналогом привилегии *PARSEC_CAP_CHMAC*). При этом для повышения или назначения несравнимого уровня целостности этой сущности или изменения ее параметров, задаваемых функцией *SILEV,* дополнительно требуется, чтобы субъект был доверенным и обладал текущей специальной административной ролью *admin_cap_role*.

Приведенные в предыдущем и текущем разделах изменения в описаниях состояний и де-юре правил их преобразования потребовали на уровне 2.1 МРОСЛ ДП-модели скорректировать формулировки теоремы об условиях безопасности системы в смысле мандатного контроля целостности, дополнив их требованием пометки с помощью функции *SSI* сущностей, параметрически ассоциированных с субъектами (в этих формулировках условий дополнительно к ранее приведенным обозначениям используются следующие обозначения: $G_i \vdash_{opi} G_{i+1}$ – переход системы из состояния G_i в состояние G_{i+1} в результате применения де-юре или де-факто правила op_i, *af_correct* – функция, задающая для каждого субъекта множество пар вида субъект и сущность, относительно которых первый субъект абсолютно функционально корректен, *ap_correct* – функция, задающая для каждого субъекта множество пар вида субъект и сущность или субъект, относительно которых первый субъект абсолютно параметрически корректен).

Определение 2. Пусть G_0 — безопасное начальное состояние системы, и существует траектория без кооперации доверенных и недоверенных субъектов $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$ (траектория, на которой доверенные субъекты не инициируют выполнение де-юре правил преобразования состояний), где $N \geq 1$. Будем говорить, что в состоянии G_N произошло нарушение безопасности системы в смысле мандатного контроля целостности, когда существуют недоверенный субъект $x \in N_{SN}$ и субъект $y \in de\_facto\_own_N(x)$ такие, что не верно неравенство $i_s(y) \leq i_s(x)$, и это условие не выполняется в состояниях G_i траектории, где $0 \leq i < N$. Назовём систему безопасной в смысле мандатного контроля целостности, когда в ней невозможно соответствующее нарушение безопасности.

Теорема 1. *Пусть G_0 — безопасное начальное состояние системы (в смысле определения 1). Пусть на всех траекториях системы без кооперации доверенных или недоверенных субъектов $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$, где $N \geq 0$, и в каждом состоянии G_N для каждого субъекта $s \in S_N$ и сущности $e \in E_N$ выполняются следующие условия:*

<u>Условие 1</u>. Если $o \in [s]$, то выполняется условие $i_{sN}(s) \leq i_{eN}(o)$ (уровень целостности объекта не ниже текущего уровня целостности субъекта, с которым он функционально ассоциирован).

<u>Условие 2</u>. Если $o \in]s[$, то $i_{sN}(s) \leq i_{eN}(o)$ и либо $SSI_N(o) = true$, либо для каждой роли или административной роли $r \in R_N \cup AR_N$ такой, что $(o, read_r) \in PA_N(r)$, выполняется условие $i_{eN}(o) \leq i_{rN}(r)$ и $SSI_N(r) = true$ (уровень целостности объекта не ниже текущего уровня целостности субъекта, с которым он параметрически ассоциирован, и, кроме того, объект либо помечен с помощью функции SSI, либо любая обладающая к нему правом доступа на чтение роль или административная роль имеет уровень целостности не ниже, чем у объекта, и помечена с помощью функции SSI).

<u>Условие 3</u>. Для всех субъектов $s \in S_N$ таких, что $i\_low < i_{sN}(s)$, выполняются условия $\{s' \in S_N \mid i\_low < i_{sN}(s') \leq i_{sN}(s)\} \times E_N \subset af\_correct_N(s)$, $\{s' \in S_N \mid i\_low < i_{sN}(s') \leq i_{sN}(s)\} \times (E_N \cup S_N) \subset$

$ap\_correct_N$ (s) (каждый субъект с текущим уровнем целостности выше минимального абсолютно функционально и параметрически корректен относительно субъектов с не большим, чем у него, и большим минимального текущим уровнем целостности, и любых сущностей).

Тогда на этих траекториях система безопасна в смысле мандатного контроля целостности (в смысле определения 2).

Следует отметить, что в определении 2 рассматриваются только траектории без кооперации доверенных и недоверенных субъектов, на которых доверенные субъекты не инициируют выполнение де-юре правил (на них они могут инициировать выполнение только некоторых де-факто правил). То есть, как это принято при разработке формальных моделей управления доступом, предполагается, что проверка безопасности системы должна осуществляться после того, как доверенные субъекты выполнили свои задачи по администрированию системы.

С практической точки зрения реализация требований пометки сущностей с помощью функции *SSI* (условие 2 теоремы 1) гораздо удобнее при администрировании ОС Astra Linux, чем только использованные ранее ограничения на предоставление к таким сущностям прав доступа на чтение ролям или административным ролям с уровнем целостности не ниже, чем у этих сущностей. При этом было осуществлено повторное доказательство данной теоремы, с учетом всех внесенных в модель корректировок.

5. Заключение и направления дальнейшей переработки модели

В настоящей статье изложены и проанализированы основные результаты очередной переработки описания МРОСЛ ДП-модели, а, именно, двух уровней ее иерархического представления, соответствующих ролевому управлению доступом и мандатному контролю целостности. При этом акцентировалось внимание на мотивированность этой переработки модели необходимостью отражения в ее описании изменений, планируемых или уже внесенных в подсистему безопасности PARSEC ОС Astra Linux релиза 2023 года.

Осуществляемая переработка модели является частью реализуемой «Группой Астра» методологии разработки безопасного системного ПО. Кроме того, поскольку с участием специалистов «Группы Астра» в настоящее время готовится проект нового национального стандарта ГОСТ Р «Защита информации. Формальная модель управления доступом. Часть 3. Рекомендации по разработке», которому, как и его предыдущим частям ГОСТ Р 59453.1-2021 и ГОСТ Р 59453.2-2021, полностью соответствует МРОСЛ ДП-модель, то настоящую ее переработку можно также считать частью процесса апробации этого проекта национального стандарта.

В будущем планируется завершить переработку других уровней иерархического представления МРОСЛ ДП-модели, соответствующих мандатному управлению доступом и штатной для ОС Astra Linux СУБД PostgreSQL.

Список литературы / References

[1]. ГОСТ Р 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения». М.: Стандартинформ. 16 с. / GOST R 59453.1-2021 «Information protection. Formal access control model. Part 1. General principles», 2021 (in Russian).

[2]. Выписка из Требований по безопасности информации, утвержденных приказом ФСТЭК России от 2 июня 2020 г. N 76. Доступно по ссылке: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, 07.12.2023 / Excerpts from Requirements for information security approved by FSTEK Russia order #76 of 2nd June 2020. Available at: https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/trebovaniya-po-bezopasnosti-informatsii-utverzhdeny-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, accessed 07.12.2023. (in Russian).

[3]. Информационное сообщение ФСТЭК России от 10.02.2021 № 240/24/647. Доступно по ссылке: https://fstec.ru/dokumenty/vse-dokumenty/informatsionnye-i-analiticheskie-materialy/informatsionnoe-

soobshchenie-fstek-rossii-ot-10-fevralya-2021-g-n-240-24-647, 07.12.2023 / Informational message of FSTEK Russia of 10th February 2021 #240/24/647. Available at: https://fstec.ru/dokumenty/vse-dokumenty/informatsionnye-i-analiticheskie-materialy/informatsionnoe-soobshchenie-fstek-rossii-ot-10-fevralya-2021-g-n-240-24-647, accessed 07.12.2023. (in Russian).

[4]. Bishop M. Computer Security: Art and Science, 2nd edition. Pearson Education Inc., 2018, 1440 p.

[5]. Девянин П. Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учебное пособие для вузов. 3-е изд., перераб. и доп. М.: Горячая линия – Телеком, 2020. 352 с.: ил. / P.N. Devyanin. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2013, 338 p. (in Russian).

[6]. ГОСТ Р 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом». М.: Стандартинформ. 12 с./ GOST R 59453.2-2021 «Information protection. Formal access control model. Part 2. Recommendations on verification of formal access control model», 2021 (in Russian).

[7]. Операционная система специального назначения Astra Linux Special Edition. Доступно по ссылке: https://astralinux.ru/software-services/os/, 07.12.2023. / Astra Linux Special Edition operating system. Available at: https://astralinux.ru/software-services/os/, accessed 07.12.2023.

[8]. Девянин П. Н., Тележников В. Ю., Третьяков С. В. Основы безопасности операционной системы Astra Linux Special Edition. Управление доступом. Учебное пособие. М., Горячая линия – Телеком, 2022, 148 стр. / Devyanin P. N., Telezhnikov V. Y., Tret'yakov S. V. Astra Linux Special Edition security basics. Access control. Hotline-Telecom, 2022, 148 p. (in Russian).

[9]. Девянин П. Н., Хорошилов А. В., Тележников В. Ю. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем. Труды ИСП РАН, том 33, вып. 5, 2021, стр. 25-40 / Devyanin P. N., Telezhnikov V. Y., Khoroshilov V. V. Building a methodology for secure system software development on the example of operating systems. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 25-40 (in Russian).

[10]. Abrial J.-R., Butler M., Hallerstede S. et al. Rodin: An open toolset for modelling and reasoning in Event-B // International Journal on Software Tools for Technology Transfer. 2010. Vol. 12, no. 6, pp. 447-466.

[11]. Девянин П. Н., Леонова М. А. Приемы по доработке описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения ее автоматизированной верификации с применением инструментов Rodin и ProB // Прикладная дискретная математика. 2021. № 52. С. 83-96. / P. N. Devyanin, M. A. Leonova, "The techniques of formalization of OS Astra Linux Special Edition access control model using Event-B formal method for verification using Rodin and ProB", Prikl. Diskr. Mat., 2021, no. 52, pp. 83–96 (In Russian).

[12]. Abrial J.-R., Hallerstede S. Refinement, decomposition, and instantiation of discrete models: Application to Event-B // Fundamenta Informaticae, Volume 77, Issue 1-2, 2007, pp. 1-28.

[13]. Девянин П. Н., Ефремов Д. В., Кулямин В. В., Петренко А. К., Хорошилов А. В., Щепетков И. В. Моделирование и верификация политик безопасности управления доступом в операционных системах. М.: Горячая линия – Телеком, 2019. 214 с.: ил./ P. N. Devyanin, D. V. Efremov, V. V. Kuliamin, A. K. Petrenko, A. V. Khoroshilov. Modeling and verification of access control access policies in operating systems. Hotline-Telecom, 2019, 214 p. (in Russian).

Информация об авторах / Information about authors

Петр Николаевич ДЕВЯНИН – член-корреспондент Академии криптографии России, доктор технических наук, профессор, научный руководитель ООО "РусБИТех-Астра" («Группа Астра»). Область интересов: теория информационной безопасности, формальные модели безопасности компьютерных систем, разработка безопасного программного обеспечения, операционные системы семейства Linux.

Petr Nikolaevich DEVYANIN – Dr. Sci. (Tech.), corresponding member of Russian Academy of Cryptography, professor, scientific director in RusBITech-Astra (Astra Linux). Field of Interest: information security theory, formal security models of computer systems, secure software development, operating systems of Linux family.

DOI: 10.15514/ISPRAS-2023-35(5)-2

CC BY

Исследование возможности идентификации веб-сайтов, посещаемых пользователем, на основе HTTP/2 трафика

[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*
[1,2] *И.А.Степанов,* ORCID: 0009-0003-1964-5001 *<stepanov.ia@phystech.edu>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*
[2] *Московский физико-технический институт, 141700, Россия, Московская область, г. Долгопрудный, Институтский пер., 9.*
[3] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20.*
[4] *Московский государственный университет имени М.В. Ломоносова 119991, Россия, г. Москва, Ленинские горы, д. 1.*

Аннотация. Конфиденциальность является важным свойством безопасности при обмене данными по сети. Для её реализации используется семейство протоколов SSL/TLS, которые, однако, в полной мере не скрывают ни посещаемого сайта, ни действий пользователя. Помимо конфиденциальности приватность также играет значимую роль для пользователей сети. Для обеспечения дополнительной приватности были реализованы некоторые программные решения, такие как Tor и I2P. В качестве меры приватности соответствующих решений может использоваться их устойчивость к специализированному классу атак. Одной из атак является Website Fingerprinting, позволяющая по трафику, отправляемому и получаемому известным пользователем, определять, какие именно сайты он посещал. Website Fingerprinting — это задача классификации, где объектом является посещение пользователем веб-сайта, а классом сам веб-сайт. В данной статье исследуется атака Website Fingerprinting для HTTP/2 трафика. В работе присутствует описание и вычисление популярных признаков, используемых при классификации трафика, и оценивается их применимость к задаче Website Fingerprinting. Для реализации атаки Website Fingerprinting строится несколько классификаторов, среди которых выбирается алгоритм, дающий лучший результат на собранном наборе данных. Точность лучшего классификатора составляет 97.8% в определённых допущениях. Кроме того, в работе присутствует оценка и анализ некоторых ограничений реального мира, влияющих на точность классификации.

Ключевые слова: Website Fingerprinting; HTTP/2; машинное обучение.

Для цитирования: Гетьман А.И., Степанов И.А. Исследование возможности идентификации веб-сайтов, посещаемых пользователем, на основе HTTP/2 трафика. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 23–36. DOI: 10.15514/ISPRAS–2023–35(5)–2.

Investigation of the Possibility of Identifying Websites Visited by the User Based on Http/2 Traffic

[1,2,3,4] *A.I. Getman*, ORCID: 0000-0002-6562-9008 <*ever@ispras.ru*>
[1,2] *I.A.Stepanov*, ORCID: 0009-0003-1964-5001 <*stepanov.ia@phystech.edu*>

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Moscow Institute of Physics and Technology (National Research University)*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.
[3] *National Research University «Higher School of Economics»*
20, Myasnitskaya ulitsa, Moscow 101978, Russia.
[4] *Lomonosov Moscow State University*
1, Leninskie Gory, Moscow, 119991, Russia.

Abstract. Confidentiality is an important security feature when exchanging data over a network. To implement it, a family of SSL/TLS protocols is used, which, however, do not fully hide either the visited site or the user's actions. In addition to privacy, privacy also plays a significant role for network users. To provide additional privacy, some software solutions have been implemented, such as Tor and I2P. As a measure of the privacy of the relevant solutions, their resistance to a specialized class of attacks can be used. One of the attacks is Website Fingerprinting, which allows the traffic sent and received by a known user to determine which sites he visited. Website Fingerprinting is a classification task, where the object is the user's visit to the website, and the class is the website itself. This article examines the Website Fingerprinting attack for HTTP/2 traffic. The paper contains a description and calculation of popular features used in traffic classification, and assesses their applicability to the Website Fingerprinting task. To implement the Website Fingerprinting attack, several classifiers are built, among which an algorithm is selected that gives the best result on the collected dataset. The accuracy of the best classifier is 97.8% under certain assumptions. In addition, there is an assessment and analysis of some real-world constraints affecting the accuracy of classification.

Keywords: Website Fingerprinting; HTTP/2; Machine learning.

For citation: Getman A.I., Stepanov I.A. Investigation of the possibility of identifying websites visited by the user based on HTTP/2 traffic. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 23-36 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-2.

1. Введение

С появлением Всемирной паутины и протокола HTTP, работающего поверх TCP/IP, встал вопрос о конфиденциальности и целостности данных TCP протокола. В 1995 году появился протокол SSL 2.0, который пытался решить эту проблему. Чуть позже в 1999 появился TLS, который и по сей день используется для обеспечения защищённой передачи между узлами во Всемирной паутине.

Однако семейство протоколов SSL/TLS не может в полной мере обеспечить конфиденциальность, так как по косвенным признакам можно примерно восстановить отправляемые запросы без доступа к их содержимому. Помимо конфиденциальности приватность также играет значимую роль для пользователей сети. Для обеспечения ещё большей приватности был создан математический аппарат mixed-сетей и луковой маршрутизации, реализованный в некоторых программных решениях. Для оценки того, насколько хорошо достигается приватность mixed-сетями и луковой маршрутизацией был разработан ряд атак, одной из которых является Website Fingerprinting. Теоретически Website Fingerprinting позволяет по перехваченному трафику, который пользователь отправляет или получает, определить, какие веб-сайты посещал пользователь. С точки зрения машинного обучения Website Fingerprinting – это задача классификации, где объектом является посещение пользователем веб-сайта, а классом сам веб-сайт.

Атакующий, отследив веб-сайты, посещаемые пользователем, может узнать некоторую конфиденциальную информацию о нём: уровень материального благополучия, состояния здоровья и так далее. С другой стороны, высокий уровень конфиденциальности может создавать проблемы провайдерам и системам безопасности, затрудняя защиту от DDoS-атак, фильтрацию небезопасных ресурсов, организацию безопасного интернета и родительского контроля. Таким образом, задача исследования уровня приватности, предоставляемого современными протоколами, и возможности идентификации посещаемых ресурсов без анализа содержимого передаваемых пакетов, является актуальной на данный момент.

Первые работы, посвящённые задачи Website Fingerprinting, использовали для классификации лишь размеры веб-объектов HTML-файла. В случае HTTP/1.1 трафика у атакующего есть больше информации, так как HTTP-запросы обрабатываются последовательно, в отличие от HTTP/2 трафика. Поэтому, в случае HTTP/1.1 трафика помимо размеров и временных меток пакетов, могут быть получены размеры веб-объектов, принадлежащих конкретной веб-странице. Однако в последнее время большинство веб-сайтов используют HTTP/2 вместо HTTP/1.1. Идентификация страниц HTTP/2 трафика является более сложной задачей, так как в этом случае происходит передача нескольких асинхронных HTTP-запросов по одному TCP-соединению. Поэтому в этом случае выделить размеры объектов уже не так просто.

Кроме того, не стоит забывать о различных ограничениях реального мира, которые могут уменьшать точность работы классификатора. В некоторых работах было показано, что время между обучением и использованием классификатора, версия браузера, наличие нескольких вкладок, открытый мир, наличие кэша могут негативно влиять на способность предсказания классификатором веб-сайтов, которые посещал пользователь.

Стоит отметить, что доля работ, посвященных атаке вида Website Fingerprinting для HTTP/2 трафика, невелика. Возникает вопрос о том, могут ли признаки и алгоритмы, используемые при решении данной задачи для других протоколов (HTTP/1.1) и других технологий (VPN, Tor), быть эффективны для HTTP/2 трафика.

Данная работа посвящена исследованию данной проблемы: эффективности атаки вида Website Fingerprinting к современному веб-трафику на основе протокола HTTP/2. Основными результатами работы являются:

- изучение эффективности различных признаков HTTP/2 трафика для атаки Website Fingerprinting
- сбор набора данных для построения классификации веб-сайтов
- построение различных классификаторов и выбор среди них оптимального
- оценка влияния различных ограничений реального мира на точность работы оптимального классификатора

В разделе 2 представлены и описаны различные термины и понятия, касающиеся задачи Website Fingerprinting. Затем в разделе 3 представлен обзор существующих решений. В разделе 4 содержится описание построения различных классификаторов и приведены результаты работы наиболее эффективных из них. Оценки ограничений реального мира описаны в разделе 5. Наконец, в разделе 6 подводятся итоги и обозначаются дальнейшие направления развития данной работы.

2. Описание задачи

В этом разделе представлены термины и понятия, использующиеся в задаче Website Fingerprinting.

2.1 Модель атаки

На рис. 1 представлена модель атаки Website Fingerprinting. Атакующий может только записывать сетевые пакеты, но при этом не может их задерживать, изменять и расшифровывать. Используя собранную информацию, атакующий может попытаться классифицировать веб-сайты, которые посещал пользователь.

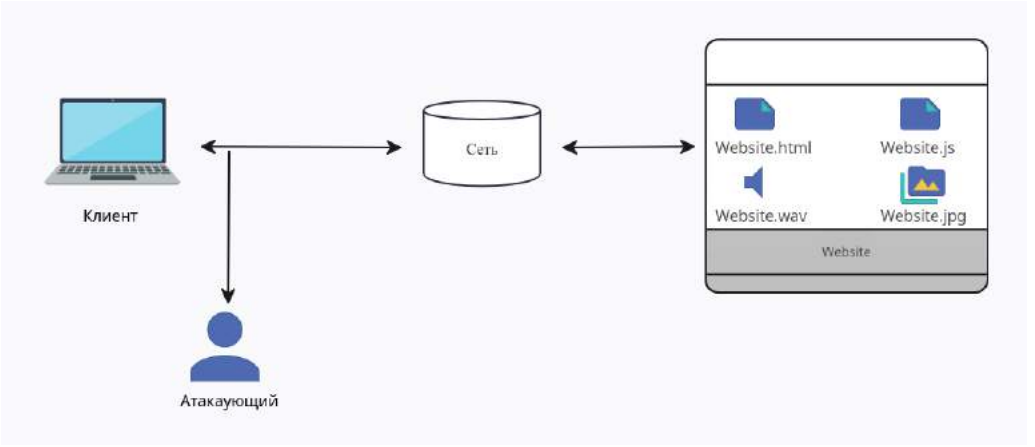


Рис. 1. Модель атаки.
Fig. 1. Attack model.

В данной работе используется термин веб-взаимодействие. Под этим термином мы понимаем набор пакетов, соответствующий посещению пользователем конкретного веб-сайта.

2.2 Метрики

В задаче Website Fingerprinting используемыми метриками являются: accuracy, recall, precision. Стоит понимать, что числовые значения данных метрик для атак Website Fingerprinting сильно зависят от набора данных, на котором обучалась и тестировалась модель машинного обучения. Поэтому, сравнивать эффективность атак Website Fingerprinting, построенных и протестированных на различных наборах данных, между собой, основываясь только на числовых значениях данных метрик, не стоит.

2.3 Сценарий открытого и закрытого мира

В сценарии закрытого мира присутствуют N классов. При обучении у атакующего есть экземпляры каждого из N классов. При тестировании классификатора предполагается, что встречаются веб-сайты только из известного набора (набора из N классов). Наиболее популярной и логичной в данном сценарии является метрика accuracy, так как чаще всего классы сбалансированы. Теперь рассмотрим сценарий открытого мира. Сначала дадим два определения:

Отслеживаемый набор – набор экземпляров веб-сайтов, о наличие которых знает атакующий при обучении и тестировании. Кроме того, при обучении атакующий знает класс, к которому принадлежит каждый экземпляр отслеживаемого набора.

Не отслеживаемый набор – набор экземпляров веб-сайтов, о наличии которых не знает атакующий при тестировании. Кроме того, при обучении атакующий не знает класс (веб-сайт), к которому принадлежит каждый экземпляр не отслеживаемого набора.

Главное отличие сценария закрытого мира от сценария открытого мира заключается в наличии не отслеживаемого набора во втором случае. При этом, классификация в данном сцена-

рии бывает двух видов. Первый: классификация трафика на отслеживаемый и не отслеживаемый набор, второй: классификация идентичная классификации в сценарии закрытого мира, но не на N классов, а на N + 1 класс, где вводится дополнительный класс, соответствующий не отслеживаемому набору.

2.4 Классификация веб-сайтов или веб-страниц

В задаче Website Fingerprinting существует некоторая разница между веб-сайтом и веб-страницей. Под веб-сайтами подразумеваются индексные или фоновые страницы веб-сайтов и классификатор настроен именно на их классификацию. Классификация же веб-страниц более сложный процесс, так как в данном случае у классификатора есть меньше информации для определения конкретной веб-страницы. Кроме того, всё усложняется при классификации веб-страниц одного и того же веб-сайта. В данном случае посещение различных веб-страниц будет иметь схожее "поведение" с точки зрения задачи Website Fingerprinting.

2.5 Отличие HTTP/1.1 от HTTP/2

Поведение протоколов HTTP/2 и HTTP/1.1 отличается друг от друга, что может приводить к тому, что методы для классификации веб-сайтов HTTP/1.1 трафика могут быть неэффективны для классификации HTTP/2 трафика. Теперь рассмотрим подробнее эти отличия.

В HTTP/2 присутствует мультиплексирование в отличие от HTTP/1.1. Мультиплексирование означает передачу нескольких асинхронных HTTP-запросов по одному TCP-соединению, в отличие от HTTP/1.1, где HTTP-запросы обрабатываются последовательно. Это свойство является принципиальным с точки зрения задачи Website Fingerprinting. В HTTP/1.1 каждый объект страницы, такой как файлы HTML, CSS, JS, JPEG и т.д., загружается последовательно в рамках отдельного TCP-соединения. В HTTP/2 же объекты загружаются параллельно, что затрудняет их выделение. Поэтому алгоритмы, основанные на коэффициенте Жаккара и размерах объектов HTML-файла, могут быть не эффективны в случае HTTP/2 трафика.

Server Push. В HTTP/2 присутствует технология Server Push, которая позволяет отправить объект клиенту, не дожидаясь запроса от него. Однако на практике эта технология используется нечасто.

Сжатие заголовков. При передаче данных часть передаваемой информации, а именно заголовки, повторяется. Для решения данной проблемы в HTTP/2 вводят сжатие заголовков с помощью алгоритма Хаффмана.

2.6 Признаки

В задаче Website Fingerprinting наиболее популярными признаками являются: направления пакетов, временные интервалы, размеры пакетов, накопленная длина пакетов. В данной работе подробно исследуется, как те или иные признаки эффективны для классификации. Представим наше веб-взаимодействие массивом пакетов F:

$$F = (p_1, \dots p_n)$$

Направления пакетов. Пусть веб-взаимодействие представлено массивом пакетов F, где $p_i = 1$ представляет пакет нисходящей линии связи (от сервера), а $p_i = -1$ представляет пакет восходящей линии связи (к серверу). Тогда массив F будет массивом признаков направления пакетов.

Размеры пакетов. Пусть веб-взаимодействие представлено массивом пакетов F, где p_i – это размер пакета с учётом направления. Тогда массив F будет массивом признаков размеров пакетов.

Временные интервалы. Пусть веб-взаимодействие представлено массивом пакетов F. Пусть t_i - время отправки пакета с номером i. Введём временной интервал между двумя пакетами:

$$d_i = t_{i+1} - t_i$$

Тогда массивом признаков временных интервалов будет следующий массив:

$$D = (d_1, \dots d_{n-1})$$

Однако признаки, связанные с временными характеристиками, могут быть сильно связаны с некоторыми свойствами сети: скорость интернета, браузер и так далее, что может сильно мешать при классификации веб-сайтов.

Сумма байт до первого встречного пакета. Пусть веб-взаимодействие представлено массивом пакетов F, где p_i – это размер пакета с учётом направления. Введём a_i – сумма подряд идущих элементов одного знака массива F до первого элемента с другим знаком. Тогда массивом данных признаков будет следующий массив:

$$A(F) = (a_1, \dots a_k)$$

Накопленная длина пакетов. Пусть веб-взаимодействие представлено массивом пакетов F, где $p_i > 0$, если пакет идёт от сервера и $p_i = 0$, если к серверу. Пусть A:

$$A(F) = (a_1, \dots a_n) : a_1 = p_1, a_i = p_i + p_{i-1}$$

Тогда массив A будет накопленной суммой пакетов.

2.7 Коэффициент Жаккара.

Пусть есть два множества A и B. Коэффициентом Жаккара для множеств A и B называется отношение:

$$J = \frac{c}{a+b-c}, \text{ где}$$

- c – количество элементов, общих для множества A и B
- a и b – количество элементов множества A и B соответственно

Данный коэффициент для алгоритма классификации был популярен в ранних работах, посвящённых Website Fingerprinting. Однако, как отмечалось ранее, с появлением новых протоколов «выделение размеров объектов» HTML страницы веб-сайта стало более проблематично, что заметно уменьшило популярность коэффициента Жаккара.

3. Обзор литературы

В этом разделе рассмотрены наиболее популярные работы и методы, посвящённые задаче Website Fingerprinting. В конце раздела представлена сравнительная таблица различных работ по данной теме.

3.1 Раннее развитие WF

Первые Website Fingerprinting атаки пытались определять URL-адреса, который пользователь посещает через зашифрованные SSL-соединения. Так в 1998 [1] была предложена атака, которая на основе объёма передаваемых данных (число байт) определяла веб-сайт, так как в случае HTTP/1.0 размер HTML-файла веб-сайта был главным признаком для классификации данного веб-сайта.

Так как в случае HTTP/1.1 трафика файлы передаются по TCP-соединению последовательно, существует возможность определить размеры отдельных объектов веб-сайта. Используя это в [2], авторы предложили классифицировать веб-сайты не по общему числу переданных байт, а по числу и размерам отдельных объектов веб-сайта. Используя эти же признаки и коэффициент Жаккара, авторы в [3] показали, что Website Fingerprinting атака может достигать приемлемой точности для значительного числа веб-сайтов в определённых допущениях.

Позже появились работы для классификации не только SSL трафика, но и VPN трафика. Так в [4] Liberatore и др. классифицируют веб-сайты VPN трафика, используя только размеры пакетов, интервалы между пакетами и коэффициент Жаккара.

3.2 WF на классических алгоритмах ML

С развитием ML алгоритмов постепенно стали появляться статьи, посвящённые Website Fingerprinting атакам на основе алгоритмов машинного обучения. Так в [5] D.Herrmann и др., используя в качестве признаков частоты распределений размеров ip-пакетов, а в качестве ML алгоритма алгоритм Наивного байеса, достигают точности в 97% для 775 различных сайтов. Помимо SSL и VPN трафика, начиная с определённого момента, стали появляться статьи, посвящённые Tor трафику. К примеру, в [6] авторы, используя направления, размеры и временные интервалы трафика, а в качестве алгоритма SVM, достигли точности 55% для закрытого мира. Кроме того, в данной работе проведено исследование сценария открытого мира.

В последствии появились работы [7], посвящённые не только построению атак, но и изучению влияния различных факторов, таких как сценарий открытого или закрытого мира, время с момента сбора трафика, наличие нескольких вкладок у пользователя, на точность атаки. В [8] авторы предложили алгоритм, основанный на расстоянии Махаланобиса и классифицирующий трафик с двумя вкладками. Точность классификации первой страницы составила 75,9%, точность классификации второй - 40,5%. При этом, интервал задержки между двумя страницами составил 2 секунды.

В 2016 A. Panchenko и др. [9], используя в качестве классификатора SVM, исследовали задачу Website Fingerprinting для открытого и закрытого мира и показали, что с увеличением числа экземпляров открытого мира точность классификатора значительно падает. При этом, в случае закрытого мира точность составила более 90% процентов. Алгоритм классификации с данным набором признаков, основанных на числе и размере ячеек в сети Tor, получил в литературе название CUMUL.

Одними из первых кто показал, что в качестве признаков для классификации могут выступать лишь направления пакетов были Avdoshin S. и др. [10]. В своей работе авторы использовали алгоритм на основе SVM, а исследуемая выборка состояла из 7 веб-сайтов. При этом, точность составляла чуть больше 70%.

С другой стороны, существуют работы, посвящённые анализу трафика с использованием информации только о временных метках. К примеру, в [11], используя в качестве алгоритма метод ближайших соседей, авторы классифицируют трафик без знания о начале и конце исследуемого веб-взаимодействия.

3.3 Эпоха глубокого обучения

С развитием глубокого обучения стало появляться большое число статей, посвящённых классификации веб-сайтов с помощью нейронных сетей. В [12] авторы, классифицируют веб-сайты, используя в качестве алгоритма сверточную нейронную сеть. Работа характерна детальным подбором оптимальных гиперпараметров сверточной нейронной сети, большим числом экземпляров набора данных, исследованием как открытого, так и закрытого мира. Стоит отметить, что в качестве признаков алгоритма выступали лишь направления пакетов (ячеек Tor) трафика. Rimmer и др. [13] в качестве алгоритма классификации использовали не только сверточную нейронную сеть, но и автоэнкодер и сеть долгой краткосрочной памяти.

Полученные результаты в различных работах говорят о том, что для классификации веб-сайтов, содержащих быстро меняющийся контент, наиболее подходят признаки, основанные лишь на размерах, направлениях и временных интервалах пакетов, в то время как признаки, созданные вручную (средний размер пакета, максимальный размер пакета и др.), могут быть неэффективны.

Работы, посвящённые атаке Website Fingerprinting для HTTP/2 трафика, стали появляться лишь в последнее время. Так в [14] авторы используют в качестве признаков накопленную длину пакетов, а в качестве алгоритма метод ближайших соседей.

В 2020 году появилась работа [15] классификации веб-сайтов на основе анализа HTTP/2 трафика. При этом, алгоритм основан на уже забытом коэффициенте Жаккара, который достаточно долго не использовался для оценки близости. Хотя точность классификации не высока относительно других алгоритмов 62,21%, однако это можно объяснить огромным числом классов 55 212. Главным достоинством работы является классификация именно HTTP/2 трафика, так как во многих других работах он почти не встречается.

В табл. 1 приведено сравнение по признакам и алгоритмам различных работ, которые упоминались выше.

Число работ, в которых встречается классификации веб-сайтов HTTP/2 трафика невелико, а количество тех, где классифицируется только HTTP/2 трафик (без HTTP/1.1) ещё меньше. Однако в последнее время доля HTTP/2 трафика заметно выросла. В данной работе исследуется атака Website Fingerprinting имено для HTTP/2 трафика.

Табл. 1. Сравнительная таблица работ
Table 1. Comparative table of works

№	Год	Трафик	Признаки	Алгоритм
1	1998	HTTP/1.0	размер html-файла	-
2	2002	HTTP/1.1	размер объектов веб-страницы	коэф. Жаккара
3	2002	HTTP/1.1	размер объектов веб-страницы	коэф. Жаккара
4	2006	VPN	размеры пакетов и временные интервалы	коэф. Жаккара
5	2009	HTTP(s)/1.1	размеры пакетов	наив. Байес
6	2011	Tor	размеры пакетов	SVM
7	2014	Tor	размеры пакетов	SVM решающее дерево наив. Байес
8	2015	HTTP(s)/1.1	размеры пакетов и временные интервалы	SVM
9	2016	HTTP(s)/1.1	размеры пакетов	SVM
10	2016	Tor	направления пакетов	SVM
11	2016	HTTP(s)/1.1 Tor	временные интервалы	другой алгоритм
12	2017	Tor	направления пакетов	сверточная нейронная сеть автоэнкодер LSTM
13	2018	Tor	направления пакетов	сверточная нейронная сеть
14	2019	HTTP(s)/2	накопленная сумма пакетов	K-NN
15	2020	HTTP(s)/2	размеры пакетов	коэф. Жаккара

4. Построение классификатора

В этом разделе будет описан процесс сбора данных и построения оптимального классификатора, то есть выбор признаков, алгоритма и гиперпараметров алгоритма, дающих лучший результат на собранном наборе данных.

Отметим, что в данной работе решается задача закрытого мира для классификации веб-сайтов HTTP/2 трафика. Кроме того, предполагается, что клиент посещает в выбранный момент времени лишь одну веб-страницу, и что нам известны время начала и конца конкретного веб-взаимодействия. Также предполагается, что клиент использует ту же версию браузера, что и атакующий. Эти допущения встречаются во многих работах и заметно упрощают задачу, но

при этом делают её менее реалистичной. В разделе 5 рассмотрены эти допущения и то, как они влияют на точность классификации.

4.1 Сбор набора данных

Для снятия трафика и сбора данных использовался Wireshark для снятия трасс и Selenium для автоматизации процесса. Сбор набора данных осуществлялся следующим образом:

Шаг 1. Открытие браузера.
Шаг 2. Открытие сайта из списка сайтов автоматическим способом.
Шаг 3. Ожидание загрузки веб-сайта.
Шаг 4. Закрытие веб-сайта, сохранение времени начала и конца сессии.
Шаг 5. Повторение 2-4 шага N раз.
Шаг 6. Фильтрация трафика, удаление пакетов, не относящихся к исследуемым веб-сайтам.
Шаг 7. Сохранение отфильтрованных pcap-файлов.
Шаг 8. Разбиение pcap-файлов на веб-взаимодействия, соответствующие посещению пользователем (клиентом) индексной страницы веб-сайта, с помощью информации о начале и конце веб-взаимодействия.

В конечном итоге было собрано 9405 веб-взаимодействий для 15 популярных веб-сайтов. При этом число экземпляров веб-взаимодействий для каждого веб-сайта одинаково. Таким образом, можно говорить о сбалансированности нашей выборки относительно классов.

4.2 Вычисление признаков

Всего было исследовано 5 различных признаков для обучения: направления пакетов, размеры пакетов, временные интервалы, сумма байт до первого встречного пакета, накопленная длина пакетов. На рис. 2 представлена схема вычисления признаков.

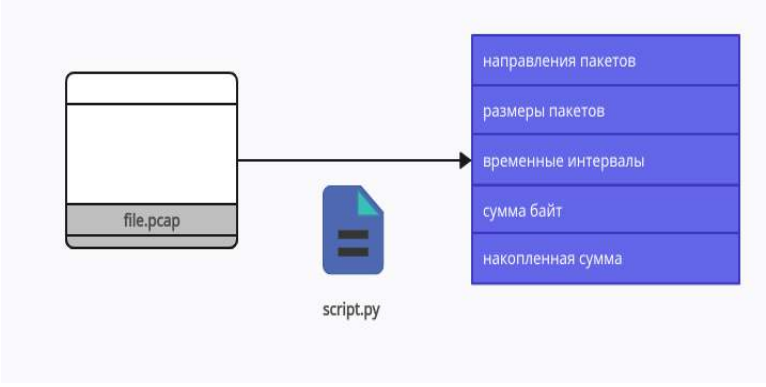


Рис. 2. Вычисление признаков.
Fig. 2. Calculating features.

4.3 Построение оптимального классификатора

Исследовались следующие алгоритмы:

- Решающее дерево
- Случайный лес
- Бустинг
- Коэффициент Жаккара

4.3.1 Решающее дерево

В качестве решающего дерева использовалось решающее дерево из scikit-learn. Точность рассчитывалась с помощью кросс-валидации с разбиением на 10 фолдов. При этом, исследовались следующие гиперпараметры алгоритма: глубина решающего дерева (depth).

На рис. 3 представлена точность работы алгоритма от глубины дерева для различных признаков.

Результаты показывают, что лучший результат при использовании решающего дерева даёт признак накопленная сумма.

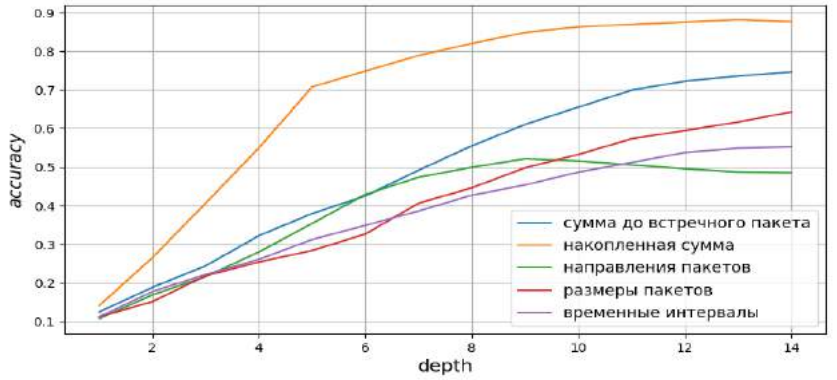


Рис. 3. Точность решающего дерева.
Fig. 3. The accuracy of the decision tree.

4.3.2 Случайный лес

В качестве случайного леса использовался случайный лес из scikit-learn. Точность рассчитывалась с помощью кросс-валидации с разбиением на 10 фолдов. При этом, исследовались следующие гиперпараметры алгоритма: число деревьев (n_estimators). На рис. 4 представлена точность работы алгоритма от числа деревьев для различных признаков.

График показывает, что лучший результат при использовании случайного леса даёт признак накопленная сумма.

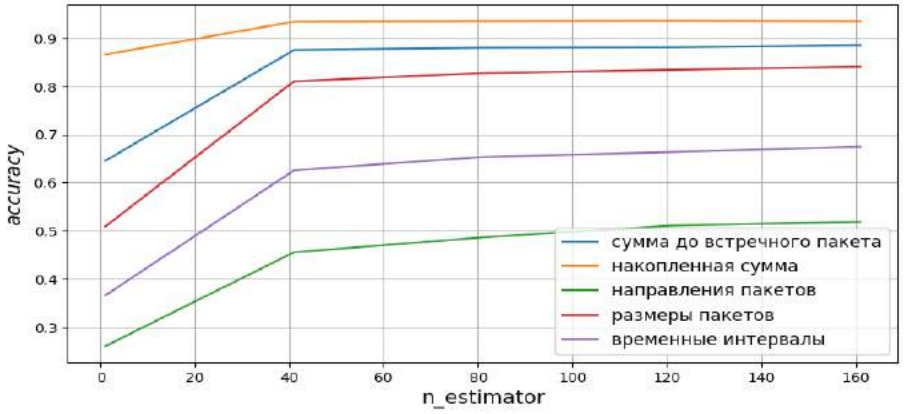


Рис. 4. Точность случайного леса.
Fig. 4. Random Forest Accuracy.

4.3.3 Бустинг

В качестве бустинга использовался CatBoostClassifier из catboost. Точность рассчитывалась с помощью кросс-валидации с разбиением на 10 фолдов. При этом, исследовались следующие

гиперпараметры алгоритма: число деревьев (iterations), глубина дерева (depth). На рис. 5 представлена точность работы алгоритма от числа деревьев для различных признаков.

График показывает, что лучший результат при использовании бустинга даёт признак накопленная сумма и при достижении 200 деревьев точность увеличивается незначительно. На рис. 6 представлена точность работы алгоритма от глубины деревьев для различных признаков.

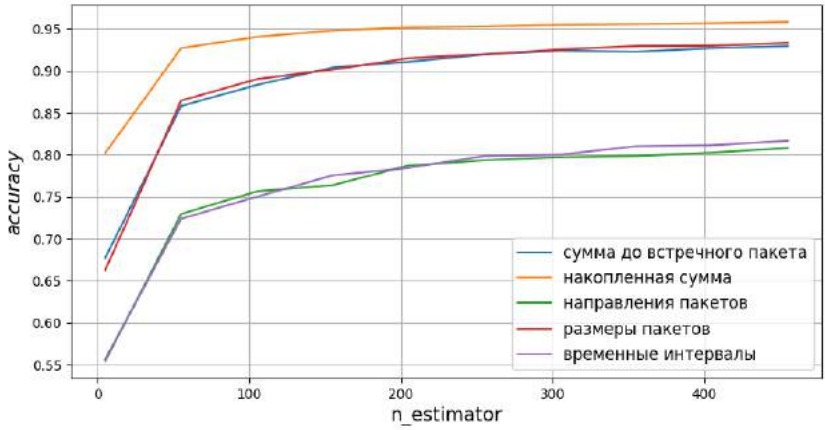


Рис. 5. Точность бустинга (число деревьев).
Fig. 5. Boost Accuracy.

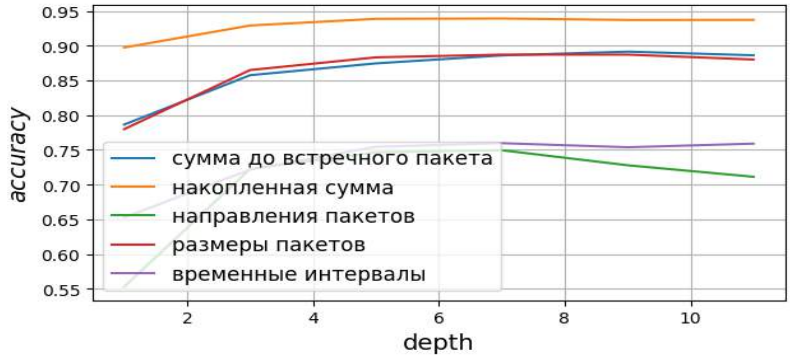


Рис. 6. Точность бустинга (глубина деревьев).
Fig. 6. Boost Accuracy.

4.3.4 Коэффициент Жаккара

Алгоритм, основанный на коэффициенте Жаккара для классификации веб-сайтов, выглядит следующим образом:

Шаг 1. Пусть дано n известных веб-сайтов (1 ... n) для каждого из которых найдены размеры его объектов.

Шаг 2. Пусть дан неизвестный веб-сайт - x, который нужно соотнести с одним из известных, и даны размеры его объектов.

Шаг 3. Рассчитываем коэффициент Жаккара для следующих пар *(1;x) ... (n;x).*

Шаг 4. Максимальное значение коэффициента Жаккара будет указывать на нужный веб-сайт.

Однако стоит понимать, что точность может сильно зависеть от того, какие веб-сайты будут установлены в качестве известных. Поэтому нужно попробовать в качестве известных все комбинации веб-сайтов и выбрать те, что дают лучший результат.

4.4 Итоги

Результаты показывают, что накопленная сумма пакетов даёт лучший результат для всех алгоритмов. В табл. 2 представлены результаты работ различных алгоритмов на собранном наборе данных. Таким образом, лучший результат на собранном наборе данных составил 97.8%.

Табл. 2. Результаты работы алгоритмов
Table 2. Results of the algorithms

№	Алгоритм	Оптимальные гиперпараметры	Лучший признак	Точность
1	Решающее дерево	max_depth = 11	Накопленная сумма	0.891
2	Случайный лес	n_estimators = 40 max_depth = 11	Накопленная сумма	0.925
3	CatBoostClassifier	iterations = 400 depth = 5 learning_rate = 0.2	Накопленная сумма	0.978
4	Коэффициент Жаккара	-	Накопленная сумма	0.907

5. Оценка влияния ограничений реального мира на точность классификации

В данном разделе будут рассмотрены различные ограничения реального мира и оценено их влияние на точность классификации.

5.1 Браузер

Оценим, как версия браузера пользователя влияет на точность классификации. Для этого обучим классификатор CatBoostClassifier на трафике, собранном с помощью браузера1. Протестируем же классификатор на трафике, собранном с помощью браузера2, который принципиально отличается от браузера1. Таким образом моделируется ситуация, когда браузеры атакующего и клиента не совпадают. В табл. 3 представлена точность работы алгоритма, обученном на наборе данных браузера1. Результаты показывают, что алгоритм имеет сильную зависимость от браузера клиента.

Табл. 3. Влияние браузера на точность классификации
Table 3. Browser influence on classification accuracy

	Тестовый набор данных	Точность
1	Браузер 1	0.975
2	Браузер 2	0.340

5.2 Классификация нескольких вкладок

Как уже было сказано, в данной работе предполагалось, что в определённый момент времени клиент имеет только одну открытую вкладку. Однако это предположение заметно отклоняет задачу от реальной. Смоделируем ситуацию, когда у клиента открыто несколько вкладок. Для этого соберём наборы данных, в которых у пользователя фоном открыты одна, две и три вкладки соответственно. Затем протестируем данные наборы данных на классификаторе, построенном в разделе 4. Результаты представлены в табл. 4.

Результаты показывают, что метод имеет сильную зависимость от числа вкладок, открытых у пользователя.

Табл. 4. Влияние нескольких вкладок на точность классификации
Table 4. The effect of multiple tabs on classification accuracy

	Тестовый набор данных (число вкладок)	Точность
1	Одна вкладка	0.975
2	Одна вкладка и одна фоновая	0.480
3	Одна вкладка и две фоновые	0.330
4	Одна вкладка и три фоновые	0.275

6. Заключение

В данной работе представлено исследование классификации веб-сайтов HTTP/2 трафика, посещаемых пользователем. В предположениях, сформулированных выше, было построено несколько классификаторов на популярных алгоритмах машинного обучения. Полученные результаты говорят о теоретической возможности классифицировать веб-сайты, посещаемые пользователем.

Однако стоит понимать, что в условиях реального мира данная классификация может иметь очень невысокую предсказательную способность, что было показано в разделе 5.

В будущей работе планируется значительно увеличить набор данных для классификации, а также более детально исследовать влияние ограничений реального мира на точность работы классификации.

Кроме того, планируется исследование возможности классификации веб-сайтов без знания о начале и конце веб-взаимодействия, что как было сказано ранее, является сильным допущением.

Список литературы / References

[1]. Mistry S. Traffic Analysis of SSL-Encrypted Web Browsing //http://bmrc. berkeley. edu/people/shailen/Classes/SecurityFall98/paper. ps. – 1998.
[2]. Hintz A. Fingerprinting websites using traffic analysis //International workshop on privacy enhancing technologies. – Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. – С. 171-178.
[3]. Sun, Q., Simon, D. R., Wang, Y. M., Russell, W., Padmanabhan, V. N., & Qiu, L. (2002, May). Statistical identification of encrypted web browsing traffic. In Proceedings 2002 IEEE Symposium on Security and Privacy (pp. 19-30). IEEE.
[4]. Liberatore, M., & Levine, B. N. (2006, October). Inferring the source of encrypted HTTP connections. In Proceedings of the 13th ACM conference on Computer and communications security (pp. 255-263).
[5]. Herrmann, D., Wendolsky, R., & Federrath, H. (2009, November). Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In Proceedings of the 2009 ACM workshop on Cloud computing security (pp. 31-42).
[6]. Panchenko, A., Niessen, L., Zinnen, A., & Engel, T. (2011, October). Website fingerprinting in onion routing based anonymization networks. In Proceedings of the 10th annual ACM workshop on Privacy in the electronic society (pp. 103-114).
[7]. Juarez, M., Afroz, S., Acar, G., Diaz, C., & Greenstadt, R. (2014, November). A critical evaluation of website fingerprinting attacks. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 263-274).
[8]. Gu, X., Yang, M., & Luo, J. (2015, May). A novel website fingerprinting attack against multi-tab browsing behavior. In 2015 IEEE 19th international conference on computer supported cooperative work in design (CSCWD) (pp. 234-239). IEEE.
[9]. Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., & Wehrle, K. (2016, February). Website Fingerprinting at Internet Scale. In NDSS.
[10]. Avdoshin, S. M., & Lazarenko, A. V. (2016). Deep web users deanonimization system. Труды Института системного программирования РАН, 28(3), 21-34.
[11]. Feghhi, S., & Leith, D. J. (2016). A web traffic analysis attack using only timing information. IEEE Transactions on Information Forensics and Security, 11(8), 1747-1759.
[12]. Sirinam, P., Imani, M., Juarez, M., & Wright, M. (2018, October). Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 1928-1943).

[13]. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., & Joosen, W. (2017). Automated website fingerprinting through deep learning. arXiv preprint arXiv:1708.06376.

[14]. Shen, M., Liu, Y., Chen, S., Zhu, L., & Zhang, Y. (2019, May). Webpage fingerprinting using only packet length information. In ICC 2019-2019 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

[15]. Ghiëtte, V., & Doerr, C. (2020, June). Scaling website fingerprinting. In 2020 IFIP Networking Conference (Networking) (pp. 199-207). IEEE.

Информация об авторах / Information about authors

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент ВШЭ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.

Aleksandr Igorevich GETMAN – PhD in physical and mathematical sciences, senior researcher at ISP RAS, associate professor at HSE. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.

Иван Александрович СТЕПАНОВ – студент МФТИ. Сфера научных интересов: анализ сетевого трафика, машинное обучение.

Ivan Alexandrovich STEPANOV is a student at MIPT. Research interests: network traffic analysis, machine learning.

DOI: 10.15514/ISPRAS-2023-35(5)-3

CC BY

Организация конфиденциальных запросов к облаку

[1] *Н.П. Варновский*, ORCID: 0000-0002-2363-0254 <*otd13isp@gmail.com*>
[2] *С.А. Мартишин,* ORCID: 0000-0001-5437-4049 <*mart@ispras.ru*>
[2] *М.В. Храпченко,* ORCID: 0000-0002-5147-5132 <*khrap@ispras.ru*>
[2] *А.В. Шокуров,* ORCID: 0000-0002-6801-7728 <*shok@ispras.ru*>

[1] *Институт проблем информационной безопасности МГУ имени М.В.Ломоносова. 119192, г. Москва, Мичуринский проспект, 1, офис 10.*
[2] *Институт системного программирования РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

Аннотация. В статье исследуется известная криптографическая задача получения клиентом данных из базы, размещенной на сервере, таким образом, чтобы никто из имеющих доступ к серверу, кроме самого клиента, не смог получить информацию о содержании этого запроса. Задача, известная как PIR (Private Information Retrieval), была сформулирована в информационно-теоретической постановке в 1995 году Шором, Голдрайхом, Кушелевицем и Суданом. Предложена модель облачных вычислений, включающая облако, центр аутентификации, пользователя, клиентов, доверенное лицо (дилера), активного противника, работающего по протоколу, на облаке. Предполагается, что у атакующей стороны имеется возможность создания фальшивых клиентов для формирования неограниченного числа запросов. Предложен алгоритм размещения базы данных на облаке и алгоритм запроса требуемого бита. Применяется инъективное преобразование номеров битов, представленных в l-ичной системе счисления словами длины d, в слова без повторяющихся цифр той же длины с алфавитом из $\hat{l}$ цифр, то есть $\{\mathbf{0},\dots,\boldsymbol{l}-\mathbf{1}\}^{\boldsymbol{d}} \to \{\mathbf{0},\dots,\hat{\boldsymbol{l}}-\mathbf{1}\}^{\boldsymbol{d}}$, что позволяет уменьшить вероятность угадывания противником номера бита. Приведены оценки коммуникационной сложности и вероятности раскрытия запрашиваемого бита с учетом выполненного преобразования.

Ключевые слова: базы данных; облачные вычисления; PIR.

Для цитирования: Варновский Н.П., Мартишин С.А., Храпченко М.В., Шокуров А.В. Организация конфиденциальных запросов к облаку. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 37–54. DOI: 10.15514/ISPRAS–2023–35(5)–3.

Благодарности: Работа выполнена при финансовой поддержке Российской Федерации в лице Минобрнауки России (соглашение № 075-15-2020-788) и ИСП РАН. Авторы выражают особую благодарность Лазареву Д. О. за ценные замечания в процессе работы над статьей.

About Cloud Request Protection

[1] *N.P. Varnovskiy* ORCID: 0000-0002-2363-0254 <*otd13isp@gmail.com*>
[2] *S.A. Martishin* ORCID: 0000-0001-5437-4049 <*mart@ispras.ru*>
[2] *M.V. Khrapchenko* ORCID: 0000-0002-5147-5132 <*khrap@ispras.ru*>
[2] *A.V. Shokurov* ORCID: 0000-0002-6801-7728 <*shok@ispras.ru*>

[1] *Information Security Institute of Moscow State Lomonosov University, Office 10, 1, Michurinskiy prospect, 119192, Moscow, RUSSIA.*
[2] *Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

Abstract. The article examines the well-known cryptographic problem of obtaining data from a database by a client so that no one with access to the server except the client himself could obtain information about this request. This problem known as PIR (Private Information Retrieval) was formulated in 1995 by Chor, Goldreich, Kushilevitz and Sudan in the information-theoretic setting. A model of cloud computing is proposed. It includes a cloud, an authentication center, a user, clients, trusted dealer, an active adversary executing the protocol in the cloud. The attacking side has the ability to create fake clients to generate an unlimited number of requests. An algorithm for the organization and database distribution on the cloud and an algorithm for obtaining the required bit were proposed. An injective transformation of bit numbers represented in the *l*-ary number system by words of length *d* into words without repeating digits of the same length with an alphabet of $\hat{\boldsymbol{l}}$ digits is used, i.e. a transformation $\{\mathbf{0}, \dots, \boldsymbol{l}-\mathbf{1}\}^{\boldsymbol{d}} \to \{\mathbf{0}, \dots, \hat{\boldsymbol{l}}-\mathbf{1}\}^{\boldsymbol{d}}$ was constructed. This transformation reduces the probability of disclosure of the requested bit number. The communication complexity and probability of revealing required bit were estimated, taking into account the performed transformation.

Keywords: database; cloud computing; PIR.

For citation: Varnovskiy N.P., Martishin S.A., Khrapchenko M.V., Shokurov A.V. About cloud request protection. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 37-54 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-3.

Acknowledgements. This work was funded by Russian Federation represented by the Ministry of Science and Higher Education (grant number 075-15-2020-788) and ISP RAS. The authors are especially grateful to D.O. Lazarev for valuable comments during the work on the article.

1. Введение

Задача (PIR – Private Information Retrieval) в информационно-теоретической постановке была сформулирована в 1995 году Шором, Голдрайхом, Кушелевицем и Суданом [1]. Задача заключается в получении информации из базы данных с открытым доступом, размещенной на облаке, таким образом, чтобы никто, кроме клиента, обращающегося с запросом, не обладал сведениями о запрашиваемой информации при наличии активного противника, работающего по протоколу через создаваемых фальшивых клиентов и за счет этого получающего больше информации.

Классическая постановка задачи PIR:

- имеется база данных – бинарная строка $X = (x_1, \dots, x_n)$ длины n, хранящаяся на сервере в облаке;
- клиент хочет получить один бит информации x_i из базы данных X так, чтобы сервер не смог определить, с какой позиции i был запрошен бит.

В работах [1-2] было показано, что при размещении базы данных на единственном сервере с пассивным противником, конфиденциальность запроса может быть обеспечена только тогда, когда клиент запрашивает базу данных целиком. Однако такой подход на практике потребует значительных ресурсов (времени скачивания информации, памяти для ее загрузки и пр.), поэтому исследования были сосредоточены на поиске решений, в которых коммуникационная сложность протокола должна быть существенно меньше размера базы

данных. Под коммуникационной сложностью протокола понимают общее количество бит, которыми обмениваются участники протокола за время его работы.

В работах [1-2] была предложена модель реплицированной базы данных с размещением нескольких копий строки $X = (x_1, \dots, x_n)$ на разных серверах. Предполагается, что клиент имеет связь с каждым из этих серверов, но сами серверы не имеют связи друг с другом. Также предполагается, что противник может наблюдать любой из серверов, на которых размещена база данных, причем, возможно, разные серверы во время выполнения разных сеансов протокола. Работы [3-4] были сосредоточены на изучении методов уменьшения коммуникационной сложности, в работах [5-8] были исследованы возможности построения схемы PIR на одном сервере при помощи различных техник, в том числе гомоморфного шифрования и оценки коммуникационной сложности предложенной схемы.

Схемы реализации PIR исследовались и для облачных вычислений. Например, в [9] предложена реализация аналогичная построению систем RAID (Redundant Array of Inexpensive Disks), где каждый сервер хранит только часть базы данных, а серверы могут быть размещены на разных платформах и иметь разных провайдеров. Предполагается, что часть из них не вступает в сговор. Серверы могут быть опрошены параллельно.

Ранее авторами в [10] была рассмотрена задача, в которой в отличие от классического PIR реплицированная база данных хранится на облаке. В этом случае ни владелец данных (пользователь), ни клиент (имеющий официальный доступ для получения информации из базы данных) не могут контролировать облачную коммуникационную среду. Таким образом, в облачных информационных системах нельзя обеспечить изолированность копий распределенной базы данных друг от друга. Кроме того, говоря о хранении и использовании конфиденциальной информации, требуется знать, с каким противником, активным или пассивным, приходится иметь дело.

Активный противник в облаке, который способен вмешиваться в ход выполнения криптографического протокола, может быть выявлен достаточно быстро путем полного анализа результатов однократного выполнения криптографического протокола. Однако активный противник может работать по протоколу и проводить атаки не нарушая протокол.

Пассивный противник получает некоторую информацию о процессе выполнения криптографического протокола путем наблюдения, но не вмешивается в процесс выполнения протокола. Пассивный противник не может быть обнаружен даже при исчерпывающем анализе результатов многократного выполнения криптографического протокола.

Поэтому, если базы данных хранятся на облачных серверах, то условия задачи PIR существенно изменяются, и ранее известные методы ее решения оказываются неприемлемыми. Чтобы найти решение задачи PIR в облачной вычислительной среде, предлагается рассмотреть иную модель взаимодействия участников протокола с базой данных.

Ниже будут рассмотрена модель, основанная на представлении номера бита в системе счисления, зависящей от числа битов и количества копий базы данных на облаке. Предложены алгоритмы представления номера бита в новой системе счисления, формирования запроса к облаку в виде множества шифротекстов, обработка ответа облака и получение значения искомого бита. Даны оценки коммуникационной сложности алгоритмов и вероятности угадывания противником номера запрошенного бита.

Вводится инъективное преобразование номеров битов, представленных в l-ичной системе счисления словами длины d, в слова без повторяющихся цифр той же длины с алфавитом из $\hat{l}$ цифр, то есть $\{0, \dots, l-1\}^d \to \{0, \dots, \hat{l}-1\}^d$. Это позволяет уменьшить вероятность угадывания противником номера бита. Приведены оценки коммуникационной сложности и вероятности раскрытия запрашиваемого бита с учетом выполненного преобразования.

2. Модель вычислений, основные определения и постановка задачи

2.1 Состав модели

Модель организации хранения и запросов к данным, предложенная в данной работе, имеет ряд отличий по сравнению с моделью, предложенной в [10]. В модель введен центр аутентификации пользователей и клиентов, через который посредством стандартных протоколов осуществляется их аутентификация. Центр аутентификации связан с дилером каналом связи, использующим стандартный криптографический протокол для обмена зашифрованными данными. Кроме того, вместо защищенных каналов [10] для связи между дилером и пользователями/клиентами достаточно иметь каналы связи, использующие стандартный криптографический протокол обмена зашифрованными данными.

Состав модели:

Облако. Состоит из нескольких серверов, на каждом из которых хранится копия одной и той же базы данных. Эти копии отличаются друг от друга при использовании различных ключей шифрования. Облачные серверы способны выполнять любые вычислительные операции, присущие системам управления базами данных. Серверы соединены посредством незащищенных каналов связи друг с другом и дилером. По этим каналам облачные серверы обмениваются по запросу информацией (получают и передают) с дилером. Облачные серверы и все примыкающие к ним каналы связи доступны для стороннего наблюдателя (противника).

Пользователь. Хранит данные на облаке. Предполагается, что на облаке хранится k копий баз данных. Для загрузки данных пользователь обращается к дилеру по каналу связи, использующему стандартный криптографический протокол конфиденциальной передачи информации.

Клиенты. Запрашивают некоторую информацию из базы данных. Обращаются для выполнения запроса к дилеру. С дилером соединены каналами связи, использующими стандартный криптографический протокол конфиденциальной передачи информации.

Центр аутентификации. Аутентифицирует пользователя и клиентов.

Дилер. Находится вне облака, противнику недоступен. Канал связи с облаком является незащищенным. Объем памяти дилера для постоянного хранения данных пренебрежимо мал по сравнению с n. Функции дилера при работе с пользователем

- вырабатывает совместно с пользователем секретный ключ для шифрования базы данных;
- получает от пользователя данные для размещения на облаке в зашифрованном виде.

Функции дилера при работе с клиентом:

- публикует открытые ключи шифрования.

При формировании запроса к облаку дилер:

- осуществляет обмен данными с облаком в процессе размещения данных на облаке и в процессе обработки запроса;
- выполняет расшифрование запроса;
- производит сортировку шифротекстов.

В дальнейшем для простоты будем считать, что база данных загружается одним пользователем. Предполагается, что после загрузки база данных не может быть изменена. Либо изменена полностью. Также будем рассматривать случай запроса одного бита.

На начальном этапе пользователь обращается в центр аутентификации. Центр проводит аутентификацию пользователя. Если полномочия пользователя подтверждены, то пользователь может передавать информацию дилеру и производить загрузку базы данных на облако.

2.2 Основные определения

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

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

Атака:

- Атака пассивного противника на облаке заключается в наблюдении за запросом от дилера и ответом облака дилеру. Эта информация используется для сбора статистики и анализа.
- Атака активного противника (атака с известными открытыми запросами) заключается в создании фальшивых клиентов и управление ими. При помощи фальшивых клиентов активный противник может сформировать произвольное число запросов, что позволит в коалиции с пассивным противником на облаке собрать и проанализировать информацию для всей базы данных.

2.3 Постановка задачи и основные обозначения

Имеется база данных – бинарная строка $X = (x_1, \dots, x_n)$ длины n.

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

Будем размещать на облаке k копий баз данных ($k \geq 4$). Пусть $k = 2^d$, где $d \geq 2$. Без потери общности предполагается, что $n = l^d$, то есть, $d = log_2 k$ и $l = \sqrt[d]{n}$. Пусть $L_p = \frac{n}{l}$.

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

Пусть x – элемент группы Z_2, а K, K_{enc}, K_{dec} - ключи шифрования и расшифрования, alg - алгоритм шифрования или расшифрования, $h \in \{1, \dots, k\}$ – номер соответствующей копии базы данных. $DEnc(x, K, alg, h)$ - функция однозначного шифрования. $REnc(x, K_{enc}, alg, h)$ и $RDec(x, K_{dec}, alg, h)$ – функции вероятностного шифрования и расшифрования.

Пусть $Len(K)$ – длина шифротекста в битах при шифровании номера бита БД в облаке односторонней функцией. Такая функция необратима, то есть при ее использовании задача восстановления номера бита по шифротексту является вычислительно сложной.

Пусть $Len(K_{enc})$ – длина шифротекста в битах при вероятностном шифровании значений битов БД в облаке. Пусть $Len(K, K_{enc})$ – сумма длин $Len(K)$ и $Len(K_{enc})$.

3. Алгоритм преобразования слов в слова без повторяющихся букв

3.1 Постановка задачи

Представим номер бита как d разрядов в l-ичной системе счисления. Выбор l и d основывается на известной модели [1,2], где номер i бита x_i представлен в l-ичной записи: $i = a_0^{(i)} + a_1^{(i)} l + \dots + a_{d-1}^{(i)} l^{d-1}$. Цифры l-ичной записи представляют собой кортежи $\left(a_0^{(i)}, \dots, a_{d-1}^{(i)}\right)$ длины d. Эти d элементов кортежа можно интерпретировать как точку с целочисленными координатами в кубе размерности d и длиной стороны l, где $l=\sqrt[d]{n}$. На каждом месте в строке стоит цифра в l-ичной системе счисления от 0 до l-1. Такое множество

точек куба размерности d можно рассматривать как множество слов длины d с алфавитом $(0,\dots,l-1)$.

Обозначим через $N_d(l)$ множество слов длины d с алфавитом $(0,\dots,l-1)$, т. е. множество $\{0,\dots,l-1\}^d$. Элементы множества $N_d(l)$ можно рассматривать как номера чисел от 0 до l^d-1, то есть l-ичную запись d-разрядного числа.

Обозначим через $\widehat{N}_d(l)$ – множество слов длины d с алфавитом $(0,\dots,l+d-2)$, где все буквы в слове различны. Элементы множества $\widehat{N}_d(l)$ можно рассматривать как подмножество номеров битов от 0 до $(l+d-1)^d-1$, где все цифры числа различны.

Построим отображение f: $N_d(l)\to\widehat{N}_d(l)$, то есть преобразуем слова множества $N_d(l)$ в слова множества $\widehat{N}_d(l)$, все буквы в которых различны.

Ниже будет доказано, что отображение f является инъективным.

Значение функции $f(a)$ для каждого $a\in N_d(l)$ будет определяться матрицей $A(a)$ размера $d\times 3$. Зафиксировав элемент a, будем эту матрицу обозначать через A.

3.2 Первый этап построения матрицы A

Сопоставим каждому элементу $a=(a_0,\dots,a_{d-1})$ (d-разрядной l-ичной записи числа) из $N_d(l)$ матрицу A размера $d\times 3$ с помощью следующей процедуры. На первом этапе строим матрицу A_0, размера $d\times 2$, первым столбцом которой будет последовательность $0,1,\dots,d-1$. Второй столбец заполним последовательно цифрами $a_0,\dots,a_{d-1}$ в l-ичном представлении числа a. Далее упорядочим строки полученной матрицы A_0 по неубыванию элементов второго столбца. При наличии последовательностей совпадающих элементов во втором столбце, содержащие эти элементы строки упорядочиваем по возрастанию элементов первого столбца. Получаем матрицу A_1.

3.3 Второй этап построения матрицы A - алгоритм формирования третьего столбца матрицы A

На вход алгоритма подается матрица A_1. На выходе алгоритма получаем матрицу A размера $d\times 3$ (листинг 1).

```
// вводим счетчик c, начальное значение c = l − 1
c ← l − 1
// вводим индикатор σ, начальное значение σ = −1
σ ← −1
// последовательно перебираем строки матрицы A1
for i ← 1 to d do
// для i-ой строки матрицы A1 выбираем элемент a_{i,2}, и
// сравниваем со значением σ.
// если σ и a_{i,2} различны, то положим σ = a_{i,2} и a_{i,3} = a_{i,2}
      if σ ≠ a_{i,2} then
          σ ← a_{i,2}
          a_{i,3} ← a_{i,2}
// если σ = a_{i,2}, то присваиваем a_{i,3} текущее значение счетчика c
      else
          a_{i,3} ← c
      c ← c + 1
// упорядочиваем строки матрицы A1 по возрастанию значений первого
// столбца и получаем матрицу A
A ← Sort(A1, 1, Ascending)
// конец алгоритма
```

Листинг 1. Алгоритм формирования третьего столбца матрицы A.

Listing 1 Algorithm for forming the third column of matrix A.

Третий столбец матрицы A будет искомой последовательностью в $\widehat{N}_d(l)$, соответствующий элементу a.

Лемма 1. Приведенный выше алгоритм переводит l-ичную d-разрядную запись числа в $(l + d - 1)$-ичную d-разрядную запись, в которой кратность каждой цифры равна единице.

Доказательство.

Во втором столбце стоят упорядоченные по неубыванию цифры. Если кратность цифры равна единице, то $a_{i,3} = a_{i,2}$ и эти цифры в третьем столбце различны: цифры $a_{i,2}$ находятся в диапазоне $[0, l-1]$.

Для тех цифр, кратность которых больше единицы первый раз цифра заменяется, как если бы ее кратность была равна единице. Начиная со второго вхождения происходит замена последующих вхождений этих цифр, в зависимости от номера строки. Производится замена $a_{i,3} = c$, счетчик c увеличивается на единицу при переходе на следующую строку. То есть для каждой строки c различно. Поскольку $c \in [l, l + d - 2]$, то не совпадает с цифрами из $N_d(l)$, принадлежащих диапазону $[0, l-1]$ ■.

Лемма 2. В матрице A всегда выполняется одно из двух соотношений:

$a_{i,3} = a_{i,2}$, если $a_{i,2} \neq a_{i-1,2}$

или

$a_{i,3} = l + i - 2$, если $a_{i,2} = a_{i-1,2}$.

Доказательство.

На первом этапе алгоритма определим $c = l - 1$. В конце каждого шага алгоритма счетчик c изменяется по формуле $c = c + 1$.

Если рассматриваем первую строку или при переборе строк матрицы A цифра $a_{i,2}$ встречается первый раз, то есть $a_{i,2} \neq a_{i-1,2}$, то в соответствии с алгоритмом $a_{i,3} = a_{i,2}$.

Если цифра $a_{i,2}$ встречалась ранее во втором столбце, то есть $a_{i,2} = a_{i-1,2}$, то в соответствии с алгоритмом $a_{i,3} = c$.

Теорема 1. Отображение f: $N_d(l) \to \widehat{N}_d(l)$ является инъективным.

Доказательство.

Возьмем два числа a' и a''. Построим для этих чисел матрицы A' и A'' согласно алгоритму. Пусть c_i - значение величины c при формировании третьего столбца i-й строки матрицы A.

Последовательно сравниваем строки матриц A' и A'' и ищем первое несовпадение элементов строк. Пусть они отличаются во втором столбце. Так как предыдущие строки совпадали, то $\sigma' = \sigma'' = \sigma$ и $c' = c'' = l + i - 2$ для матриц A' и A''. Поскольку $a'_{i,2} \neq a''_{i,2}$, без нарушения общности можно предполагать, что $a'_{i,2} < a''_{i,2}$. Поскольку вторые столбцы матриц A' и A'' отсортированы по неубыванию, по лемме 2 возможны два варианта:

1. $a'_{i,2} > \sigma$. Тогда $a''_{i,2} > \sigma$. Тогда $a'_{i,3} = a'_{i,2}$, а $a''_{i,3} = a''_{i,2}$. Следовательно, так как матрица A'' отсортирована по неубыванию, элемент $a'_{i,2} < a''_{k,2}, k \in [i, d]$ и в силу этого неравенства цифра $a'_{i,2}$ из матрицы A' отсутствует в третьем столбце матрицы A'', следовательно, образы элементов a' и a'' различаются.
2. $a'_{i,2} = \sigma$. Тогда $a'_{i,3} = c = l + i - 2$. По алгоритму цифра $l + i - 2 = c_i$ может стоять только в i-ой строке матрицы A'' в третьем столбце. В силу того, что $a'_{i,2} < a''_{i,2}$, а элементы строк были равны до i-ой строки, то в i-ой строке матрицы A'' во втором столбце стоит цифра $a''_{i,2}$ (в i-ой строке будет $a''_{i,3} = a''_{i,2}$, а $a''_{i,2} \in [0, l-1]$). То есть цифра $l + i - 2$ отсутствует в записи второго числа.

Пусть теперь при первом несовпадении строк элементы вторых столбцов матриц совпадают, а первые различаются. В этом случае $a'_{i,3} = a''_{i,3}$, но эти цифры находятся в разных разрядах образов элементов a' и a'', и, следовательно, согласно лемме 1 не могут совпадать ■.

В последующих алгоритмах матрица A использоваться не будет.

4. Загрузка данных на облако

4.1 Инициализация массива $\hat{X}$

Напомним, что $n = l^d$ и $\hat{l} = l + d - 1$.

Обозначим: $\hat{n} = \hat{l}^d$.

Ранее было определено преобразование f такое, что

$$f\colon N_d(l) \to \widehat{N}_d(l).$$

Определим массив $\hat{X} = (\hat{x}_1, \dots, \hat{x}_{\hat{n}})$.

Таким образом каждому номеру бита в l-ичной системе счисления будет соответствовать номер в $(l + d - 1)$-ичной системе счисления (точка в кубе размерности d со стороной $\hat{l}$). Цифры записи номера бита из $N_d(l)$ в $\widehat{N}_d(l)$ будут представлены различными целыми числами от 0 до $\hat{l} - 1$.

Заметим, что $\widehat{N}_d(l) \subset N_d(\hat{l})$, поэтому не для всех элементов $N_d(\hat{l})$ существует прообраз в множестве $\widehat{N}_d(l)$.

Массив $(x_1, \dots, x_n)$ это биты. Тогда массив $\hat{X}$ также массив битов.

В массиве $\hat{X}$ заполним только элементы, для которых в множестве $N_d(\hat{l})$ существует прообраз в множестве $\widehat{N}_d(l)$ по следующей формуле:

$\hat{x}_{f(i)} = x_i$, $(i = 1, \dots, n)$.

На остальных местах могут стоять случайные значения битов.

Заметим, что нам не требуется, чтобы существовало обратное преобразование $f^{-1}\colon N_d(\hat{l}) \to N_d(l)$, поскольку из облака возвращается значение бита, а не его номер.

4.2 Вспомогательные построения, необходимые для работы алгоритма запроса бита - алгоритм построения матрицы G

Построим матрицу G, состоящую из $\hat{n}$ строк и 2 столбцов ($i = 1,..,\hat{n},\ j = 1,2$).

$$G = \begin{pmatrix} 1 & g_{1,2} \\ \vdots & \vdots \\ \hat{n} & g_{\hat{n},2} \end{pmatrix}$$

Первый столбец этой матрицы – последовательность первых $\hat{n}$ натуральных чисел $[1, \hat{n}]$. Второй столбец матрицы G – элементы $\hat{X} = (\hat{x}_1, \dots, \hat{x}_{\hat{n}})$.

4.3 Алгоритм загрузки данных на облако

Обозначим через $G_{enc}(h)$ $(h \in \{1, \dots, k\})$ матрицу, хранящую зашифрованные элементы матрицы G. В первом столбце в i строке матрицы $G_{enc}(h)$ хранятся зашифрованные значения $g_{i,1} - 1$ матрицы G. Во втором столбце в i строке матрицы $G_{enc}(h)$ хранятся зашифрованные значения $g_{i,2}$ матрицы G. Матрицы $G_{enc}(h)$ формируются для k копий базы данных.

Алгоритм загрузки данных на облако аналогичен алгоритму, приведенному в [10].

На вход алгоритма подается бинарный массив $X = (x_1, \dots, x_n)$. В результате работы алгоритма на облако загружается k зашифрованных копий баз данных.

Шаг 1. Пользователь передает дилеру полученный массив номеров битов и значений, т.е. массив $X = (x_1, \dots, x_n)$. Дилер аутентифицирует пользователя и принимает данные.

Шаг 2. Дилер формирует матрицу $G = \|g_{i,j}\|$, где $f(i) = 1,..,\hat{n},\ j = 1,2$.

Шаг 3. Дилер генерирует ключи $K(h)$, $K_{enc}(h)$ и $K_{dec}(h)$, где $h \in \{1, \dots, k\}$.

Ключ $K(h)$ (реализующий одностороннюю функцию) предназначен для шифрования первого столбца матрицы G.

Открытый $K_{enc}(h)$ и секретный $K_{dec}(h)$ ключи предназначены для вероятностного шифрования/расшифрования значений столбцов матрицы G.

Шаг 4. Дилер шифрует ключом $K(h)$ первый столбец матрицы G $DEnc(f(i), K(h), alg, h)$, где $f(i) = 1, .., \hat{n}, h \in \{1, \dots, k\}$ и получает столбец шифротекстов, соответствующих номерам строк матрицы G.

Дилер шифрует ключом $K_{enc}(h)$ второй столбец матрицы G: $REnc\left(g_{ij}, K_{enc}(h), alg, h\right)$ где $f(i) = 1, .., \hat{n},\ h \in \{1, \dots, k\}$ и получает шифротексты, соответствующие элементам второго столбца матрицы G для каждой копии базы данных.

Дилер получает матрицу $G_{enc}(h)$, где $h \in \{1, \dots, k\}$.

Шаг 5. Матрицу $G_{enc}(h)$, где $h \in \{1, \dots, k\}$, дилер сортирует по первому столбцу.

Шаг 6. Дилер загружает матрицу $G_{enc}(h)$, где $h \in \{1, \dots, k\}$ на облако.

Шаг 7. Шаги 4-6 дилер выполняет в цикле для каждой копии базы данных. По окончании перебора k копий базы данных происходит выход из алгоритма.

Конец алгоритма.

Число шифротекстов, переданных каждой из k копий базы данных равно $2 \cdot \hat{n}$.

Заметим, что поскольку в дальнейшем расшифровка первого столбца матрицы G не выполняется, в качестве алгоритма шифрования можно использовать одностороннюю функцию $DEnc(x, K_{enc}, alg, h)$, где K_{enc} является аргументом односторонней функции. Для возможности поиска по базе данных односторонняя функция должна быть однозначной, то есть без коллизий (различным открытым текстам соответствуют различные шифротексты).

Для шифрования второго столбца матрицы G используется вероятностное шифрование. Дилер применяет вероятностное шифрование/расшифрование значения. Вероятностное шифрование позволяет для исходных значений 0 или 1 получить различные шифротексты в случае одинаковых исходных значений.

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

В приведенном ниже алгоритме дилер должен выполнять Шаги 4-6 для каждой копии базы данных последовательно. Иначе ему придется хранить в памяти все k копий базы данных, что приводит к большим расходам памяти.

5. Алгоритм выполнения запроса клиента дилером

Напомним, что имеется отображение:

$$f\colon N_d(l) \rightarrow \widehat{N}_d(l).$$

Также напомним, что элементы множества $N_d(l)$ будем также интерпретировать как l-ичную запись d-разрядного числа. Тогда множество $N_d(l)$ представляет все целые числа от 0 до $l^d - 1$.

После преобразования f в представлении номера бита в $\hat{l}$-ичной системе счисления у нас все цифры различны. Следовательно, можно получить $d!$ различных перестановок из различных цифр для преобразованного числа из $N_d(l)$. Каждая такая перестановка будет соответствовать единственному числу, так как все цифры различны.

Пусть $\{\sigma_1, \dots \sigma_{d!}\}$ - множество элементов симметрической группы S_d, причем σ_1 является тождественной перестановкой, то есть σ_1 - единица. σ_i, где $i > 1$ порождаются идентично при помощи генератора перестановок и σ_i не требуют хранения.

Для каждого элемента $\sigma_i \epsilon S_d$ определено отображение

$$h_\sigma: N_d(\hat{l}) \to N_d(\hat{l}).$$

по формуле

$$h_{\sigma_j}(a_0, \dots, a_{d-1}) = (\sigma_j(a_o), \dots, \sigma_j(a_{d-1})), \text{ где } a_i \epsilon N_d(\hat{l}).$$

Согласно такому определению, множество $N_d(\hat{l})$ преобразуется в себя. Ввиду интерпретации элементов множества $N_d(\hat{l})$ как целых чисел из отрезка $[0, \hat{l}^d - 1]$ можно определить $\hat{l} \times d!$-матрицу $M = \| m_{i,j} \|$, где

$$m_{i,1} = i - 1, i = 1, .., \hat{n},$$
$$m_{i,j} = h_{\sigma_j}(i-1), i = 1, .., \hat{n}, \ j = 2, \dots, d!$$

Матрица M также не хранится в памяти, поскольку однозначно генерируется при помощи перестановок σ и параметра i.

5.1 Использование множества номеров в запросе для сокрытия номера *i*

Пусть клиент интересуется элементом $x_i \in X$. По построению номеру i соответствует строка матрицы M с номером $g(i) = f(i-1)$, где $g(i) \in [1, \hat{n}]$.

Для сокрытия при запросе искомого номера бита, вместо одного запрашиваемого зашифрованного номера будем передавать множество зашифрованных номеров. Выбираем $f(i)$ строку матрицы M ($m_{g(i),1} = g(i)$).

Разобьем отрезок $[1, \hat{n}]$ на $\hat{l}$ интервалов по $\hat{L}_p = \frac{\hat{n}}{\hat{l}}$ элементов в каждом интервале.

Строка матрицы M состоит из $d!$ элементов. Для каждого элемента выберем по одному числу y в каждом из оставшихся $\hat{l} - 1$ интервалов (по построению числа все будут различные). Таким образом каждому числу $m_{g(i),j}$ $(j = 1, .., d!)$ в $g(i)$-й строке матрицы M поставим в соответствие $\hat{l} - 1$ чисел, таким образом, чтобы эти числа лежали в различных интервалах и не попадали в интервал, где лежит $m_{g(i),j}$.

Поскольку отрезок $[1, \hat{n}]$ был разбит на $\hat{l}$ интервалов по $\hat{L}_p = \frac{\hat{n}}{\hat{l}}$ элементов в каждом интервале, то любое число, принадлежащее $[1, \hat{n}]$ попадет в один из интервалов. Необходимо найти номер интервала w, в который попало число $m_{f(i),j}$.

Определяется номер интервала, в который попало число $m_{g(i),j}$ при разбиении отрезка $[1, \hat{n}]$ на $\hat{l}$ интервалов:

$$w = \left\lfloor \frac{m_{g(i),j} - 1}{\hat{L}_p} \right\rfloor + 1.$$

Заметим, что $w \in \{1, \dots, \hat{l}\}$.

Приведем $m_{g(i),j}$ по модулю $\hat{L}_p$. Обозначим через $r = m_{g(i),j} \ mod \hat{L}_p$ результат приведения.

Положим

$$m'_{g(i),j} = \begin{cases} r, \text{если } r \neq 0 \\ \hat{L}_p, \text{если } r = 0 \end{cases}$$

порядковый номер в интервале с номером w.

Сформируем вектор-столбец $\boldsymbol{b}$, состоящий из различных случайных натуральных чисел b_m значения которых лежат в диапазоне $[1, \hat{L}_p]$, $m \in \{1, \dots, \hat{l}\}$. Числа b_m генерируются датчиком псевдослучайных чисел. На вход датчика *PRNG(db,m)* подается номер интервала m, для которого вычисляется b_m. Вектор-столбец $\boldsymbol{b}$ восстанавливается каждый раз одинаково. Элементы вектора-столбца в общем случае могут совпадать.

Хранить вектор-столбец $\boldsymbol{b}$ нет необходимости, поскольку при одинаковом начальном значении параметра m датчик срабатывает одинаково. Вектор $\boldsymbol{b}$ противнику неизвестен.

Пусть $a = \left(m'_{g(i),j} - b_w\right) mod \hat{L}_p$. Число a противнику неизвестно.

Для всех интервалов (где $m \in [1, \hat{l}]$) найдем $\hat{l}$ элементов y с учетом a по формуле:

$$c = \begin{cases} (a + b_m) mod L_p, & \text{если } c \neq 0 \\ L_p, & \text{если } c = 0 \end{cases}$$

$$y = L_p \cdot (m - 1) + c$$

Обозначим через множество $Set_{g(i),j}$ числа y для элемента $m_{g(i),j}$.

Обозначим через множество $Set_{g(i),j}$ числа y для элемента $m_{g(i),j}$. По построению множество $Set_{g(i),j}$ включает в себя элемент $m_{g(i),j}$ и $\left|Set_{g(i),j}\right| = \hat{l}$.

Пусть $Set_{g(i)} = \cup Set_{g(i),j}$, где $j = 1, .., d!$. Следовательно, мощность множества $\left|Set_{g(i)}\right|$ равна $\hat{l} \cdot d!$.

Число a и вектор $\boldsymbol{b}$ позволяют осуществлять сдвиг в каждом интервале. Сдвиг нужен для того, чтобы не дать противнику возможность определить состав множества $Set_{g(i)}$ за один шаг.

Алгоритм формирования множества $Set_{g(i)}$

На вход алгоритма (листинг 2) подается: $\hat{L}_p$, $\hat{l}$, i, d. На выходе множество $Set_{g(i)}$.

```
g(i) ← i
// формируем строку матрицы M с номером f(i)
// построим множества Set_{g(i),j} для каждого элемента строки матрицы M с
// номером g(i)
for j← 1 to d! do
Set_{g(i),j} ← ∅
        // найдем номер интервала
        w ← ⌊(m_{g(i),j} − 1)/L̂_p⌋ + 1
        // приведем m_{g(i),j} по модулю L̂_p
        r ← m_{g(i),j} mod L̂_p
        // найдем порядковый номер числа в интервале с номером w
        m'_{g(i),j} = { r, если r ≠ 0
                      { L̂_p, если r = 0
        // при помощи датчика псевдослучайных чисел PRNG(db,1)
        // генерируется b_w
        // найдем число a
        a = (m'_{g(i),j} − b_w) mod L̂_p
        for m ← 1 to l̂ do
                // при помощи датчика псевдослучайных чисел PRNG(db,m)
                // генерируется b_m
                c ← (a + b_m) mod L_p
                if c = 0 then
                        c ← L_p
                y ← L_p · (m − 1) + c
                Set_{g(i),j} ← Set_{g(i),j} ∪ y
```

Листинг 2. Алгоритм формирования множества $Set_{g(i)}$

Listing 2. $Set_{g(i)}$ formation algorithm.

5.2 Описание алгоритма построения множеств $Set_{f(i)}$ дилером

Дилер шифрует множество $Set_{g(i)}$ и отправляет запрос облаку, состоящий из $\hat{l} \cdot d!$ шифротекстов. Дилер получает ответ от облака как множество шифротекстов значений мощностью $\hat{l} \cdot d!$

На вход алгоритма подается: $\hat{L}_p$, $\hat{l}$, i, d. На выходе алгоритм формирует множество $Set_{g(i)}$.

Шаг 1. Дилер получает номер запрашиваемого бита i, преобразует его в $g(i)$ и строит строку матрицы M из $d!$ элементов.

Шаг 2. Дилер находит номер интервала w, в который попал номер запрашиваемого бита $m_{g(i),j}$ и его порядковый номер в этом интервале $m'_{g(i),j}$. Для вектора-столбца $\boldsymbol{b}$ дилер при помощи датчика псевдослучайных чисел *PRNG(db,w)* находит элемент b_w.

Шаг 3. Дилер находит число a для номера интервала w, куда попал номер $m_{g(i),j}$: $a = \left(m'_{g(i),j} - b_w\right) mod \hat{L}_p$.

Шаг 4. Для всех интервалов (где $m \in \left[1, \hat{l}\right]$) дилер находит $\hat{l}$ элементов y по формулам:

$$c = \begin{cases} (a + b_m) mod L_p, & \text{если } c \neq 0 \\ L_p, & \text{если } c = 0 \end{cases}$$

$$y = L_p \cdot (m - 1) + c$$

$$Set_{g(i),j} \cup y$$

Шаги 2-4 выполняются для каждого элемента построенной строки матрицы M.

Шаг 5. Дилер строит множество: $Set_{g(i)} = \cup\, Set_{g(i),j}$, где $j = 1, \dots, d!$ ■

5.3 Запрос клиентом *i*-го бита. Предварительные замечания

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

При запросе клиентом i-го бита дилер строит множество $Set_{g(i)}$, которое содержит элементы $g(i)$-й строки матрицы M. Напомним, что в $g(i)$-й строке элемент $m_{g(i),1}$ является искомым номером. Множество $Set_{g(i)}$ содержит $\hat{l} \cdot d!$, элементов, чтобы скрыть от противника информацию о запрашиваемом бите.

Напомним, что физически в облаке хранятся матрицы $G_{enc}(h)$, $h \in \{1, \dots, k\}$. Строка матрицы $G_{enc}(h)$ состоит из шифротекста номера строки и шифротекста значения.

По аналогии с [10] для каждого элемента множества $Set_{g(i)}$ случайно и равномерно выбирается номер копии базы данных. Все элементы множества $Set_{g(i)}$ для которых выбрана копия базы данных h, обозначим через множества $Set^h_{g(i)}$, где $h \in \{1, \dots, k\}$. Очевидно, что множества $Set^h_{g(i)}$ не пересекаются между собой, поскольку все элементы множества $Set_{g(i)}$ различны и каждому элементу ставится в соответствие ровно один номер h. Объединение множеств $Set^h_{g(i)}$ содержит все элементы множества $Set_{g(i)}$, то есть $Set_{g(i)} = \cup\, Set^h_{g(i)}$, где $h \in \{1, \dots, k\}$.

Далее каждое множество $Set^h_{g(i)}$ шифруется ключом $K(h)$, полученные шифротексты сортируются для каждого множества $Set^h_{g(i)}$ отдельно. В результате получим множества $Set^{enc(h)}_{g(i)}$ – зашифрованные соответствующим ключом $K(h)$ и затем отсортированные.

Номер искомого бит попадает в одно из множеств $Set^{enc(h)}_{g(i)}$. Дилер запоминает номер h и соответствующий номер шифротекста в перестановке. Это соответствие необходимо запомнить и хранить на время выполнения запроса, чтобы дилер мог восстановить значение запрашиваемого бита.

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

Первый элемент вектора-строки $\boldsymbol{s}_{num}$ содержит искомый номер элемента i_{cnv}.

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

Третий элемент вектора-строки $\boldsymbol{s}_{num}$ содержит номер шифротекста в перестановке элементов множества $Set_{g(i)}^{enc(h)}$. Таким образом, s_3 является функцией от трех параметров. Эта функция устанавливает соответствие между исходным номером $f(i)$ и номером шифротекста в перестановке:

$$s_3 = F\left(s_1, s_2, Set_{g(i)}^{enc(s_2)}\right).$$

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

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

На k копий базы данных дилер отправляет в общей сложности $\hat{l} \cdot d!$ шифротекстов для элементов множества: $Set_{g(i)}$.

После выполнения запроса дилер при помощи вектора-строки $\boldsymbol{s}_{num}$ определяет значение искомого элемента для строки матрицы M из копии базы данных. После расшифрования шифротекста, дилер получает значение запрашиваемого бита и отправляет его клиенту.

5.4 Подготовка запроса к облаку

На входе алгоритм получает номер запрашиваемого бита i. На выходе дилер получает зашифрованные и отсортированные множества $Set_{g(i)}^{enc(h)}$ и вектор-строку $\boldsymbol{s}_{num}$.

Шаг 1. Клиент запрашивает у дилера значение бита с номером i.

Шаг 2. Дилер восстанавливает $f(i)$-ю строку матрицы M и генерирует множество $Set_{g(i)}$.

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

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

Шаг 5. Дилер выполняет разбиение множества $Set_{g(i)}$ на непересекающиеся подмножества $Set_{g(i)}^{h}$ путем случайного и равномерного выбора номера копии базы данных для каждого элемента множества $Set_{g(i)}$.

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

Шаг 6. Дилер шифрует элементы множеств $Set_{g(i)}^{h}$ в соответствии с выбранной копией базы данных ключом $K(h), h \in \{1, \dots, k\}$, получает шифротексты и сортирует их для каждой копии базы данных (множества $Set_{g(i)}^{enc(h)}$).

Шаг 7. Дилер заносит в третий элемент вектора-строки $\boldsymbol{s}_{num}$ номер шифротекста соответствующего элементу s_1 в перестановке множества $Set_{g(i)}^{enc(s_2)}$ ■.

5.5 Выполнение запроса к облаку и ответ облака

На вход алгоритма подаются зашифрованные и отсортированные множества $Set_{g(i)}^{enc(h)}$. На выходе дилер получает шифротексты значений битов в том же порядке, в котором передавались шифротексты, соответствующие номерам битов.

Шаг 1. Для каждой копии базы данных дилер отправляет на облако отсортированные шифротексты номеров битов для множеств $Set_{g(i)}^{enc(h)}$.

Шаг 2. Для запрошенных шифротекстов номеров битов облако возвращает дилеру шифротексты значений битов из второго столбца матрицы $G_{enc}(h)$, где $h \in \{1, \dots, k\}$).

Порядок возвращаемых шифротекстов соответствует исходной сортировке шифротекстов для множеств $Set_{g(i)}^{enc(h)}$ ■.

5.6 Обработка ответа облака

На входе дилер получает второй столбец матриц $G_{enc}(h)$ для шифротекстов множеств $Set_{g(i)}^{enc(h)}$. На выходе значение i-го бита.

Шаг 1. Дилер при помощи вектора-строки $\boldsymbol{s}_{num}$ выбирает зашифрованное значения бита. Остальные шифротексты отбрасываются.

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

Шаг 3. Дилер отправляет значение i-го бита клиенту ■.

6. Оценка требуемой памяти и сложности алгоритма

6.1 Объем информации, который необходимо хранить дилеру

В процессе работы на время загрузки базы данных дилер генерирует k ключей $K_{enc}(h)$. Эти ключи служат для вероятностного шифрования значений битов. После загрузки ни ключи, ни шифротексты дилер не хранит.

На протяжении всего времени существования базы данных дилер хранит информацию для этой базы данных объема n:

- значение для инициализации датчика случайных чисел *PRNG (db, i)*;
- k ключей $K(h), h \in \{1, \dots, k\}$ для шифрования односторонней функцией;
- k ключей $K_{dec}(h)$, где $h \in \{1, \dots, k\}$ для вероятностного расшифрования значений битов.

Заметим, что на практике число n битов в базе данных традиционно предполагается 2^{30} - 2^{40}, а число копий баз данных k не превышает 16. Таким образом, k значительно меньше n, а хранение k ключей для шифрования номеров битов и k ключей для расшифрования значений битов занимает $2 \cdot k \cdot l_{key}$, где l_{key} - длина ключа. Длина ключа на практике чаще всего ограничена 256 байтами [11]. Следовательно, дилер хранит объем информации значительно меньше n.

6.2 Характеристики предложенной схемы

Напомним, что $Len(K)$ – длина шифротекста в битах при шифровании номера бита БД в облаке односторонней функцией, $Len(K_{enc})$ – длина шифротекста в битах при вероятностном шифровании значений битов БД в облаке и $Len(K, K_{enc})$ – сумма длин $Len(K)$ и $Len(K_{enc})$.

Предложенная схема организации базы данных размера n и запроса к ней удовлетворяет следующим свойствам. На время загрузки базы данных дилер генерирует генерирует k ключей $K_{enc}(h)$. Эти ключи служат для вероятностного шифрования значений битов. После загрузки ни ключи, ни шифротексты дилер не хранит.

Как было сказано ранее, на протяжении всего времени существования базы данных объема n дилер хранит значение для инициализации датчика случайных чисел *PRNG(db,i)*, k ключей $K(h), h \in \{1, \dots, k\}$ для шифрования односторонней функцией и k ключей $K_{dec}(h)$, где $h \in \{1, \dots, k\}$ для вероятностного расшифрования значений битов.

Объем хранимой информации много меньше n.

Напомним, что $Len(K)$ – длина шифротекста в битах при шифровании номера бита БД в облаке односторонней функцией, $Len(K_{enc})$ – длина шифротекста в битах при вероятностном шифровании значений битов БД в облаке и $Len(K, K_{enc})$ – сумма длин $Len(K)$ и $Len(K_{enc})$.

Предложенная схема организации базы данных размера n и запроса к ней удовлетворяет следующим свойствам. На время загрузки базы данных дилер генерирует:

- k ключей $K_{enc}(h)$ для вероятностного шифрования значений битов, которые после окончания загрузки не хранятся.

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

6.3 Основные результаты

Под коммуникационной сложностью для предложенной схемы будем понимать общее количество пересылаемых битов, необходимых для обмена информацией между дилером и облаком. То есть сумму числа битов, отправляемых дилером на облако и числа битов, получаемых от облака дилером, необходимых для нахождения дилером значения бита, запрашиваемого у дилера.

Утверждение 1. Коммуникационная сложность схемы получения значения номера бита дилером без раскрытия его номера равна $\hat{l} \cdot d! \cdot Len(K, K_{enc})$ битов.

Доказательство. Для запроса значения бита дилер посылает копиям базы данных $\hat{l} \cdot d!$ (где $\hat{l} = \sqrt[d]{\hat{n}}$) шифротекстов номеров битов длиной $Len(K)$. Облако отвечает $\hat{l} \cdot d!$ шифротекстами значений битов длиной $Len(K_{enc})$ ■.

Проанализируем вероятность угадывания противником номера запрашиваемого бита.

Если запрошены два номера бита, то они могут принадлежать либо одному, либо разным множествам $Set_{g(i)}$. Противник может увидеть, выполнен ли запрос к одному из множеств $Set_{g(i)}$, на которые разбита база данных или к разным. Если запросы выполнены к одному множеству, то они неразличимы. Таким образом, если противник совершает n или более запросов, он не узнает номер бита. Максимум, какую информацию противник может получить - такое подмножество множества $Set_{g(i)}$, что при запросе к каждому элементу из подмножества, на облаке осуществляется доступ ко всем битам из множества $Set_{g(i)}$ и только к ним. Что будет означать, что противник не знает, а только угадывает, какой именно бит из данного множества был выбран ■.

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

Доказательство.

Множество $Set_{g(i),j}$ ($Set_{g(i),j} \subset Set_{g(i)}$) порождается одинаково для всех $\hat{l}$ чисел, в него входящих. То есть существует одинаковое множество для $\hat{l}$ различных номеров битов. Таким образом вероятность угадывания номера из множества $Set_{g(i),j}$ равна $\frac{1}{\hat{l}}$, так как $Set_{g(i),j}$ имеет мощность $\hat{l}$. Так как $Set_{g(i)} = \cup Set_{g(i),j}$ где $j = 1, \dots, d!$, вероятность угадывания номера i будет не более $\frac{1}{\hat{l} \cdot d!}$ ■.

Сделав n или более запросов, противник получит информацию о числе реальных битов в каждом множестве $Set_{g(i)}$. Число этих битов будет меньше или равно $\hat{l}$. Таким образом, противник понимает, что запрошенный клиентом номер находится среди истинных номеров битов. И вероятность того, что клиент запрашивал конкретный номер реального бита для всех реальных номеров битов из $Set_{g(i)}$ одинаковая.

Для того, чтобы оценить вероятность угадывания конкретного номера бита из множества $Set_{g(i)}$, необходимо посчитать среднее число истинных битов, попадающих в множество $Set_{g(i)}$.

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

Пусть P_r - вероятность получения в выборке мощности $\hat{l}$ ровно r элементов, соответствующих элементам множества X. Тогда P_r находится по формуле [12]:

$$P_r = \frac{C_n^r \cdot C_{\hat{n}-n}^{\hat{l}-r}}{C_{\hat{n}}^{\hat{l}}}.$$

Тогда среднее число истинных битов, попадающих в множество $Set_{g(i)}$, равно:

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

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

Доказательство.

Выполнив *n* запросов, противник определяет элементы, входящие в каждое множество $Set_{g(i)}$ (в предположении, что $k < ld$). Если фиктивные клиенты будут выполнять любое количество запросов большее n, противник все равно не получит никакой дополнительной информации.

Количество реальных бит, которые попали в каждый интервал в среднем равно N. Так как $Set_{g(i)} = \cup\, Set_{g(i),j}$ где $j = 1, \dots, d!$, вероятность угадывания номера *i* в среднем будет не более $\frac{1}{N \cdot d!}$ ■.

Так как значение k по сравнению с n мало, хранение k пар ключей для шифрования/расшифрования значений битов не требует много памяти и занимает $2 \cdot k \cdot l_{key}$.

Докажем, что предложенный протокол удовлетворяет условию лаконичности.

Теорема 2. Для предложенной схемы организации базы данных размера *n*:

- Коммуникационная сложность запроса равна $\hat{l} \cdot d! \cdot Len(K, K_{enc})$ битов.
- При однократной атаке с известным открытым запросом номера бита *i* и предположении о наличии пассивного противника на облаке вероятность угадывания противником номера бита не более $\frac{1}{\hat{l} \cdot d!}$.
- При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником в среднем не более $\frac{1}{N \cdot d!}$.

Доказательство. Из Утверждения 1 следует, что коммуникационная сложность запроса равна $\hat{l} \cdot d! \cdot Len(K, K_{enc})$ битов.

Из Утверждения 2 следует, что при однократном запросе номера бита вероятность угадывания номера равна $\frac{1}{\hat{l} \cdot d!}$.

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

7. Заключение

Целью этого исследования было уменьшение вероятности угадывания номера бита при незначительном увеличении коммуникационной сложности. В результате проведенного исследования было показано, что использование другого основания системы счисления и инъективного отображения для представления номера бита позволяет получить следующие результаты.

Так как число копий баз данных мало (на практике обычно $k \leq 16$), а дилер хранит не более $O(k)$ бит информации, то дилер в состоянии обслуживать запросы к значительному числу баз данных. В этом случае имеется возможность эффективно использовать облако как место хранения значительного объема информации.

В ранее предложенном алгоритме коммуникационная сложность составляет $l{\cdot}d{\cdot}(Len(K)+2\ Len(K_{enc}))$ бит. А вероятность угадывания номера - $\frac{1}{l}$. Так как $\hat{l} = l + d - 1$, а d мало (практике d обычно не превышает 4 для 16 копий баз данных), то можно считать, что l и $\hat{l}$ отличаются незначительно. Если предположить, что $Len(K)$ и $Len(K_{enc})$ мало различаются, то сравнивая результаты предыдущего и нового алгоритма, получаем увеличение коммуникационной сложности в $\frac{2}{3}(d-1)!$ То есть при $d = 4$ получим увеличение коммуникационной сложности в 4 раза. Вероятность угадывания уменьшается в $d!$ раз, то есть в 24 раза при $d = 4$.

При $d \geq 2$ вероятность угадывания уменьшается быстрее, чем растет коммуникационная сложность.

Таким образом, предложенные методы для рассматриваемой схемы организации базы данных просты в реализации и снижают вероятность угадывания номера и значения бита.

Список литературы / References

[1]. Chor B., Goldreich O., Kushilevitz E., Sudan M. Private Information Retrieval, in IEEE Annual Symposium on Foundations of Computer Science, 1995, pp. 41–50.

[2]. Chor B., Goldreich O., Kushilevitz E., Sudan M. Private Information Retrieval, Journal of the ACM, Vol. 45, No. 6, November 1998, pp. 965–982.

[3]. Gasarch W. A survey on private information retrieval, Bulletin of the EATCS, 2004 pp. 72-107

[4]. Yekhanin S. Locally Decodable Codes and Private Information Retrieval Schemes, Springer Heidelberg Dordrecht London New York, ISSN 1619-7100, ISBN 978-3-642-14357-1 e-ISBN 978-3-642-14358-8, DOI 10.1007/978-3-642-14358-8 2010, p.82

[5]. Kushilevitz E., Ostrovsky R. Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In Proc. of the 38st IEEE Sym. on Found. of Comp. Sci., pages 364 373, 1997.

[6]. Kushilevitz E., Ostrovsky R. One-Way Trapdoor Permutations Are Sufficient for Non-trivial Single-Server Private Information Retrieval. In EUROCRYPT00, 2000.

[7]. Ostrovsky R., Skeith III W. E. A Survey of Single-Database Private Information Retrieval: Techniques and Applications". Proceedings of the 10th International Conference on Practice and Theory in Public-Key Cryptography. Springer-Verlag. pp. 393–411.

[8]. Aguilar-Melchor C., Barrier J., Fousse L. XPIR: Private Information Retrieval for Everyone, Proceedings on Privacy Enhancing Technologies 2016(2), pp. 155-174, DOI:10.1515/popets-2016-0010.

[9]. Demmler D., Herzberg A., Schneider T. RAID-PIR: Practical multi-server PIR CCSW '14: Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, November 2014 Pages 45–56б https://doi.org/10.1145/2664168.2664181, DOI:10.1145/2664168.2664181.

[10]. Мартишин С.А., Храпченко М.В., Шокуров А.В. Организация безопасного запроса к базе данных на облаке. Труды Института системного программирования РАН. Том 34, № 3, 2022г., с. 173-188, ISSN 2079-8156 (Print), ISSN 2220-6426 (Online).

[11]. Wahid M.N.A., Ali A., Esparham B., Marwan M. A Comparison of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention. Journal of Computer Science Applications and Information Technology, 2018, 3(2); 1-7.

[12]. Ширяев А. Н. Вероятность – 1, Москва, Из-во МЦНМО, 2021, изд. 7, стереот., 552 с. ISBN 978-5-4439-1557-9.

Информация об авторах / Information about authors

Николай Павлович ВАРНОВСКИЙ – старший научный сотрудник Института проблем информационной безопасности МГУ имени М.В.Ломоносова. Сфера научных интересов: математика и информационная безопасность, теория сложности.

Nikolay Pavlovich VARNOVSKIY – researcher of Mathematical Studies in Information Security Section of Information Security Institute of Moscow State Lomonosov University. His research interests are mathematics, Information Security and Cryptography, complexity theory.

Сергей Анатольевич МАРТИШИН – кандидат физико-математических наук, научный сотрудник отдела теоретической информатики Института системного программирования им. В.П. Иванникова РАН. Его научные интересы включают параллельные алгоритмы, базы данных, облачные вычисления.

Sergey Anatolievich MARTISHIN – Cand. Sci. (Phys.-Math.), researcher of the Department of Theoretical Computer Science of Ivannikov Institute for System Programming of the RAS. His research interests include parallel algorithms, databases, cloud computing.

Марина Валерьевна ХРАПЧЕНКО – научный сотрудник отдела теоретической информатики Института системного программирования им. В.П. Иванникова РАН. Её научные интересы включают параллельные алгоритмы, базы данных, облачные вычисления,.

Marina Valerievna KHRAPCHENKO – researcher of the Department of Theoretical Computer Science of Ivannikov Institute for System Programming of the RAS. Her research interests include parallel algorithms, databases, cloud computing.

Александр Владимирович ШОКУРОВ – кандидат физико-математических наук, доцент, заведующий отделом теоретической информатики Института системного программирования им. В.П. Иванникова РАН с 2019 года. Сфера научных интересов: алгебраические структуры в полях Галуа, базисы Гребнера, модулярная арифметика, нейрокомпьютерные технологии, цифровая обработка сигналов, криптографические методы защиты информации.

Alexander Vladimirovich SHOKUROV – Cand. Sci. (Phys.-Math.), Professor, Head of the Department of Theoretical Computer Science of Ivannikov Institute for System Programming of the RAS since 2019. Research interests: algebraic structures in the Galois fields, modular arithmetic, neurocomputer technologies, Grobner bases, digital signal processing, cryptographic methods for protecting information.

DOI: 10.15514/ISPRAS-2023-35(5)-4

Метод мутации сложноструктурированных входных данных при фаззинг-тестировании JavaScript интерпретаторов

Н.С. Ерохина, ORCID: 0000-0002-4878-0865 <*ens@secdev.space*>

Академия ФСО России,
302034, Россия, г. Орёл, ул. Приборостроительная, д. 35.

Аннотация. Фаззинг-тестирование JavaScript интерпретаторов является одним из наиболее сложных направлений в тестировании веб-браузера, ввиду сложности генерации его входных данных. Интерпретаторы обрабатывают JavaScript код на веб-странице и требуют постоянной поддержки новых стандартов языка и усложнения своей архитектуры. Наиболее распространенные сегодня фаззеры не способны эффективно мутировать сложноструктурированные входные данные при фаззинг-тестировании. Генерация JavaScript кода с нуля не позволяет инкапсулировать необходимую семантику, а текущие мутаторы быстро разрушают синтаксис и семантику языка входных данных. В данной статье представлена новая стратегия мутации, сохраняющая синтаксис и семантику входных данных за счет модификации AST-деревьев фрагментов JavaScript кода. Данный метод позволяет эффективно генерировать разнообразные и корректные входные данные, которые могут привести к выявлению ошибок и уязвимостей в интерпретаторах JavaScript. Данный метод может быть использован для повышения безопасности веб-браузеров и обеспечения надежности интерпретации JavaScript кода.

Ключевые слова: мутации сложноструктурированных данных; интерпретатор JavaScript; уязвимости программного обеспечения; фаззинг-тестирование.

Для цитирования: Ерохина Н. С. Метод мутации сложноструктурированных входных данных при фаззинг-тестировании JavaScript интерпретаторов. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 55–66. DOI: 10.15514/ISPRAS–2023–35(5)–4.

Method for Mutation of Complexly Structured Input Data during Fuzzing of Javascript Engines

N.S. Erokhina, ORCID: 0000-0002-4878-0865 <*ens@secdev.space*>

Academy of the Federal Guard Service of Russian Federation,
35, Priborostroitelnaya st., Orel, 302034, Russia.

Abstract. Fuzzing of JavaScript engines is one of the most difficult areas in web-browser testing due to the complexity of input data generating. JavaScript engines process JavaScript code on a web page and require constant support for new language standards and increasing complexity in their architecture. The most common fuzzers today are not able to effectively mutate complexly structured input data during fuzzing. Generating JavaScript code from scratch does not allow encapsulating the necessary semantics, and current mutators quickly destroy the syntax and semantics of the input data language. This article presents a new mutation strategy that preserves the syntax and semantics of the input data by modifying the AST of JavaScript code fragments. This method allows you to efficiently generate diverse and correct input data, which can lead to the identification of errors and vulnerabilities in JavaScript engines. This method can be used to improve the security of web browsers and ensure reliable interpretation of JavaScript code.

Keywords: mutations of complex structured data; JavaScript engine; software vulnerabilities; fuzzing.

For citation: Erokhina N.S. Method for mutation of complexly structured input data during fuzzing of JavaScript engines. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 55-66 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-4.

1. Введение

Высокая сложность современного программного обеспечения, обусловленная большим объемом исходного кода, иногда достигающим нескольких миллионов строк, а также наличие уязвимостей и ошибок в нем, является фундаментальной проблемой, вызванной текущим состоянием развития информационных технологий. Веб-браузер является ярким примером, демонстрирующим данный факт. На рис. 1 представлена поверхность атаки веб-браузера.

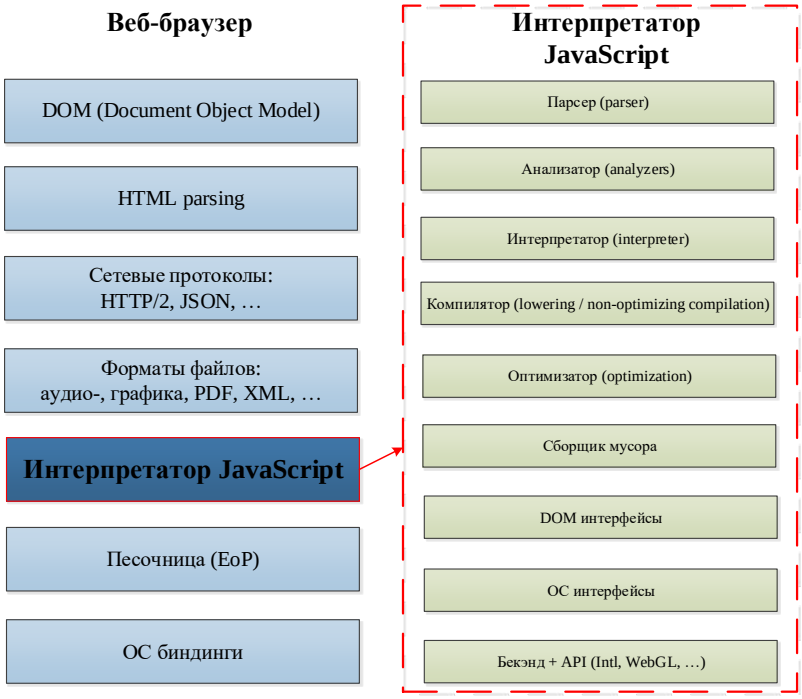


Рис.1. Поверхность атаки веб-браузера.
Fig. 1. Web-browser attack surface.

Среди всей архитектуры веб-браузера, поиск уязвимостей в JavaScript интерпретаторах является наиболее сложной, потому востребованной задачей в тестировании веб-браузеров. Появление в 90-х годах 20 века динамических веб-приложений стало отправной точной популярности языка JavaScript. По данным рейтинга W3Techs[1] на октябрь 2023 года 98,8% всех веб-сайтов используют язык JavaScript. Любой современный веб-браузер поддерживает язык динамической разработки JavaScript без использования дополнительного программного обеспечения (ПО) с помощью встроенного JavaScript интерпретатора. Стремительный рост числа новых интернет-технологий требует постоянного усложнения и наращивания кодовой базы интерпретатора веб-браузера. Данный факт часто приводит к ошибкам, что негативно сказывается на безопасности веб-браузера в целом и активизирует деятельность авторов вредоносных программ. Согласно Национальной базе данных уязвимостей (NVD2), 43% всех

[1] https://w3techs.com/technologies/details/cp-javascript.
[2] https://nvd.nist.gov/.

уязвимостей, обнаруженных в веб-браузерах Microsoft Edge и Google Chrome, были уязвимостями интерпретатора JavaScript.

2. Актуальные проблемы фаззинг-тестирования интерпретаторов JavaScript

Современные руководящие документы по безопасной разработке предлагают широкий спектр методов экспертного, статического и динамического анализа[3]. Наиболее распространенные из них: межпроцедурный контекстно-чувствительный анализ, анализ бинарного кода, а также байт-кода, формальная верификация, отслеживание помеченных данных и сканирование уязвимостей, а также направленный анализ участков кода и моделирование атак. Каждый из них имеет свои достоинства и недостатки, однако фаззинг-тестирование применительно к задаче выявления уязвимостей сложного программного обеспечения, такого как JavaScript-интерпретаторы имеет ряд преимуществ:

- высокая степень автоматизации, что позволяет проводить тестирование большого объема кода за обозримое время;
- может быть использован для тестирования любой системы, которая имеет входные данные;
- возможность обнаружения новых и неизвестных ранее уязвимостей;
- автоматизированное тестирование может выявить те ошибки, которые не удалось найти методом ручного тестирования, за счёт большего покрытия кода;
- автоматизированное тестирование позволяет человеку участвовать лишь на этапе работы с результатами;
- позволяет собрать общее представление о защищенности тестируемого кода.

Фаззинг становится все более популярным методом проверки функциональности программного обеспечения и поиска уязвимостей безопасности. Он успешно применяется для тестирования различных приложений, начиная от механизмов рендеринга и процессоров изображений и заканчивая компиляторами и интерпретаторами.

Программная реализация, которой проводится фаззинг-тестирование, называется фаззером. В различных источниках классификация фаззеров может отличаться, чаще всего она производится по следующим параметрам: по информации о целевой программе, которая будет подвергнута фаззингу; по наличию обратной связи с тестируемой программой; по операциям, которые будут совершаться над входными данными, а по методам генерации данных [1]. В области тестирования JavaScript интерпретаторов проведено множество исследований, среди них: генеративные фаззеры создают новые тестовые случаи с нуля на основе заранее определенной грамматики, такие как jsfunfuzz[4] и Fuzzilli [2], или путем их создания из синтезируемых блоков кода, разобранных на большой корпус [3]. Мутационные фаззеры генерируют новые тестовые примеры на начальных входных данных для тестирования. Например, LangFuzz [4] разбивает программы в большом корпусе на небольшие фрагменты кода, рекомбинирует их с исходным вводом и генерирует новые тестовые примеры; Skyfire [5], Nautilus [6] и Superion [7] мутируют каждую программу индивидуально с помощью сегментов, полученных от других программ в корпусе, или с их правилом мутации.

По количеству информации о целевой программе фаззеры подразделяются на: фаззеры «белого ящика», фаззеры «серого ящика» (например, [2[2], 6-7]) а также фаззеры «черного

[3] Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении (утверждена приказом ФСТЭК России в декабре 2020 года).

[4] https://github.com/MozillaSecurity/funfuzz.

ящика» (например, [3-5]). Среди распространенных фаззеров JavaScript интерпретаторов нет представителей «белого ящика» ввиду сложности и объемности кода интерпретаторов.

В последние годы фаззинг «серого ящика» на основе покрытия зарекомендовал себя как один из наиболее эффективных методов поиска ошибок безопасности на практике. Фаззинг-тестирование методом «серого ящика» предполагает внедрение инструментов в тестируемую программу для получения обратной связи для управления ходом тестирования. Главное преимущество метода в том, что фаззер получает информацию не только о выводе и аварийном завершении программы, но и о ходе исполнения программы. В общем случае информация может быть любой, но обычно используют такую метрику, как покрытие кода программы.

AFL – это канонический пример фаззера, который в 2013 году дал толчок в массовому использованию фаззинга с обратной связью, и обнаруживший тысячи громких уязвимостей. Его базовая идея заключается в сборе покрытия ветвей при каждом исполнении, а цель – максимизация покрытия. Сейчас первоначальный AFL уже не используется, но от него образовалось множество проектов. Самый популярный и быстроразвивающийся из них – это AFLPlusPlus (AFL++) [8]. Он вбирает в себя новые техники и постоянно расширяет возможности исследователей. AFL++ известен своей высокой производительностью и эффективностью в обнаружении уязвимостей и ошибок в программном обеспечении. AFL++ обрабатывает целевую программу в два этапа: инструментирование целевой программы и фаззинг инструментированной программы. На этапе инструментирования в точки ветвления тестируемой программы внедряются маяки для считывания покрытия ветвей вместе с количеством попаданий в них.

Однако, применение фаззинга с обратной связью к интерпретаторам JavaScript нетривиально. AFL++ эффективен при работе с бинарными данными, а не с текстовыми и тем более не с жестко структурированными входными данными, такими как JavaScript код. AFL приходится тратить много времени на борьбу с корректностью синтаксиса, находя при этом только ошибки синтаксического анализа. Универсальные алгоритмы обрезки и мутации данных, встроенные в фаззер AFL++, работают на битовом уровне представления JavaScript кода, что разрушают тонкую семантику или условия, закодированные в коде. Из чего следует, что большая часть предложенных некорректных мутированных входных данных с высокой вероятностью будет мешать обнаруживать новые пути в коде. Тестовые примеры, сгенерированные текущими методами, могут ссылаться на переменные, которые не определены в текущем контексте выполнения. Вследствие чего, программа может завершиться из-за синтаксической или семантической ошибки уже во время выполнения. На рис. 2 представлена разница операций замены и вставки значений встроенным в AFL++ способом и на основе AST.

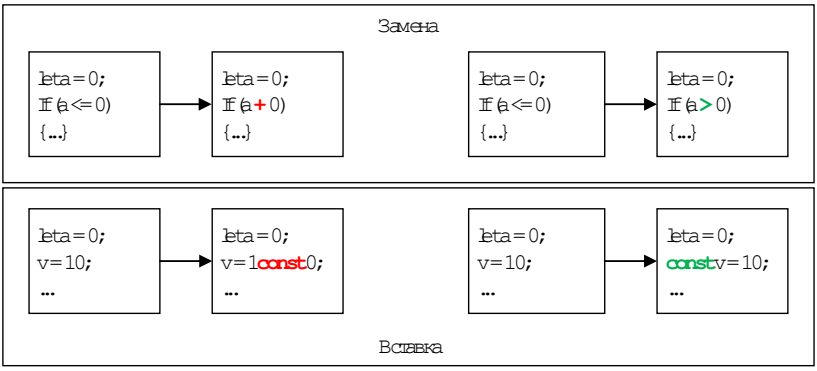


Рис. 2. Пример мутации AFL++ (а) и мутации на основе AST-дерева (б).
Fig. 2. Example of AFL++ mutation (a) and mutations based on AST (b).

Не менее важно также ответить на вопросы: как генерировать тестовые примеры, охватывающие больше программных путей, и как определить мутацию, которая направит тестирование на новые, еще не исследованные пути в программе. Повышение покрытия кода означает повышение охвата состояний выполнения программы и повышение качества тестирования. Известно, что увеличение покрытия кода на 1% увеличивает процент обнаруженных ошибок на 0,92%[5]. Однако, большинство тестовых примеров охватывают только некоторое, ограниченное число путей, в то время как большая часть кода не достигается. Распределение ошибок в программах чаще не сбалансированно, то есть, например, 80% ошибок могут находиться в 20% программного кода, что приводит к тому, что многие фаззеры тратят много времени на тестирование неуязвимых путей.

Большинству фаззеров, основанных на покрытии, включая AFL++, требуется набор входных данных, которые они используют в качестве основы для запуска процесса фаззинга. Корпус хорошего качества имеет решающее значение для производительности и эффективности фаззера. Получить такой высококачественный корпус непросто: если язык, принимаемый целевым приложением, широко используется, одним из подходов является сканирование общедоступных примеров из Интернета или из общедоступных репозиториев кода. Однако эти примеры, скорее всего, будут смещаться в сторону очень часто используемых частей грамматики, в которых используются хорошо протестированные части целевого ПО. Естественно, исследователи безопасности часто хотят протестировать функции, которые редко используются или были представлены совсем недавно и, которые с большей вероятностью приведут к ошибкам в целевом ПО. Собрать корпус для тестирования этих функций явно сложнее, а написание примеров вручную обходится очень дорого. Вследствие вышеизложенных проблем, результаты фаззинг-тестирования интерпретаторов сильно уступают результатам, достигнутым в других областях [9].

Дополнительной сложностью является выбор данных, инкапсулирующих в себе необходимую семантику, позволяющую более эффективно выявлять уязвимости в тестируемом коде. В случае с JavaScript интерпретаторами подходящими данными являются файлы регрессионных тестов браузеров, написанные специалистами вручную и нацеленные на проверку функциональности веб-браузеров. В исследовании [10] была выявлена корреляция между файлами регрессионных тестов и фрагментами кода, вызывающими уязвимости, и наглядно продемонстрирована эффективность их использования.

Эффективность фаззинг-тестирования JavaScript интерпретаторов может быть повышена за счет использования новых алгоритмов обрезки и мутации, которые нацелены на сохранение синтаксиса JavaScript языка и семантики регрессионных тестов. Проведенный анализ литературы позволяет утверждать, что, на сегодняшний день, разработка алгоритмов эффективной обрезки и мутации сложноструктурированного кода JavaScript-интерпретаторов является достаточно сложным и востребованным процессом, с точки зрения информационной безопасности [6-11].

3. Метод мутаций сложноструктурированных данных

Многими исследованиями наглядно продемонстрирована эффективность представления и дальнейшей обработки сложноструктурированных данных в виде абстрактного синтаксического дерева [7-10]. Абстрактное синтаксическое дерево или AST (от англ. Abstract syntax tree) – это конечное, помеченное, ориентированное дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья – с соответствующими операндами [12]. Каждый входной JavaScript файл преобразуется в AST-дерево в соответствии со спецификацией языка ECMAScript.

[5] C. Miller, Fuzz by number: More data about fuzzing than you ever wanted to know // in Proceedings of the CanSecWest – 2008

В отличие от токенов, используемых при мутациях на основе словаря, AST фактически моделируют тестовые входные данные как объекты с именованными свойствами, четко соблюдая структуру кода. Таким образом, мутация на уровне AST обеспечивают подходящую степень детализации для фаззера, а также позволяет сохранять синтаксис и семантику JavaScript языка.

Стратегия обрезка реализуется путем итеративного удаления каждого поддерева в AST входных данных и наблюдения за различиями в покрытии. Алгоритм пытается обрезать AST-дерево, так чтобы сохранить исходное покрытие. Обрезанные входные данные имеют преимущество более короткого времени выполнения и меньшего набора потенциальных мутаций во время дальнейшей обработки. Обрезка поддеревьев направлена на то, чтобы сделать их как можно короче, при этом вызывая новые пути в коде. На рис. 3 показан пример разработанной стратегии обрезки входных данных на языке JavaScript, где полная строка (выделена перечеркиванием) обрезается без внесения каких-либо различий в покрытии. С помощью встроенной стратегии обрезки AFL++ практически невозможно отсечь такое полное утверждение.

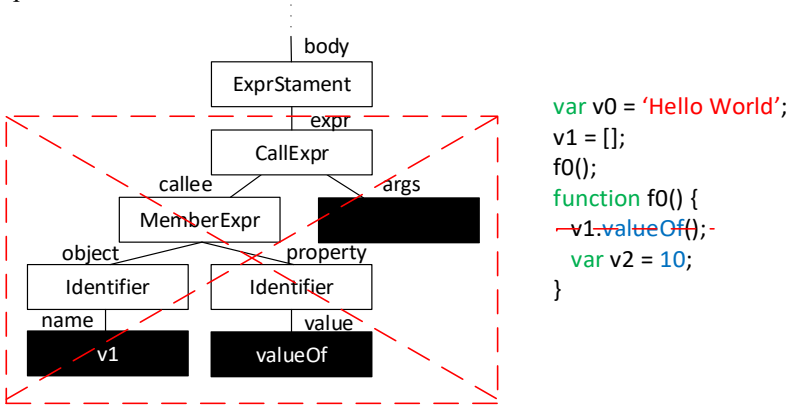


Рис. 3. Пример обрезки файла на основе AST-дерева .
Fig. 3. Example of file trimming based on AST.

Процедура разработанной стратегии представлена на рис. 4 псевдокодом алгоритма. Случайно выбранный входной файл преобразуется в AST-дерево. Далее алгоритм проходит по каждому узлу в дереве и пытается обрезать поддерево, для которого данный узел является корневым. Если обрезка проходит успешно и итоговое покрытие тестируемого кода тестом не ухудшается – обрезанный входной файл добавляется в очередь фаззера. Минимизация входных данных направлена на удаление избыточности в данных, мешающей фаззеру наиболее эффективно производить мутации.

После того, как AST-дерево было обрезано, используется несколько методов мутации для создания новых входных данных для фаззера (рис. 5). Разработанная стратегия мутации AST-дерева JavaScript кода включает в себя последовательное применение алгоритмов: мутаций узла AST (рис. 6), случайных мутаций, мутаций объединения (рис. 7), а также AFL++ мутаций. Данная стратегия изменяет AST входных данных JavaScript, сохраняя с высокой вероятностью структуру кода, влияющую на общие потоки управления, и исходные типы используемых переменных.

Мутации узла направлены на замену 25% узлов в AST-дереве на инверсные, либо случайно выбранные «интересные» значения из представленного словаря. Случайная мутация выбирает случайный узел дерева и заменяет его случайно сгенерированным новым поддеревом, корнем которого является тот же нетерминал. Мутация объединения объединяет входные данные, которые нашли разные пути, помещая поддерево из одного файла в другой. Для этого выбирается случайный внутренний узел, который становится корнем заменяемого

поддерева. и из дерева в очереди берется случайное поддерево, корнем которого является тот же нетерминал. Мутация AFL выполняет мутации, которые также используются AFL, например, перестановку битов с целью проверки синтаксического анализатора интерпретатора.

```
Input: the test input to be trimmed input
Output: the set of trimmed test T
Function Trimming(input)
T = ∅
E = ∅ // Множество ошибок
tree ← ParseToAst(input)
coverage ← RunEngine(tree) // Вычиление начального
 покрытия
seq ← TraverseAst(tree)
count ← CountNodes(tree) // Вычиление числа узлов дерева
step = count
while step > 1 do
    removable ← seq[step] // Выбор узла для обрезки
    trimmed, E ← RemoveNode(tree, removable) // Обрезка
     дерева
    if E == ∅ then
        coverage_new ← RunEngine(trimmed)
        if coverage_new ≥ coverage then
            T = T ∪ {trimmed} // Добавление экземпляра в
             последовательность
        end
    end
    step = step − 1
end
return T
```

Рис. 4. Псевдокод алгоритма обрезки AST-дерева.
Fig. 4. Pseudocode of the AST trimming algorithm.

```
Input: Мутируемые входные данные input
Output: Мутированное AST mutatedTree
Function Mutation(input)
E = ∅ // Множество ошибок
tree ← ParseToAst(input)
while True do
    mutStrategy = RandomChoice(mutateNodes,
                               mutateLiterals,
                               mutateExpressions,
                               mutateSubtrees)

    mutatedTree, E ← mutStrategy(tree)
    if mutatedTree ≠ None ∧ E == ∅ ∧ mutatedTree ≠ tree then
        mutatedCode ← ParseToCode(mutatedTree)
        return mutatedCode
    end
end
```

Рис. 5. Псевдокод общего алгоритма мутаций AST-дерева JavaScript кода.
Fig. 5. Pseudocode of the main AST mutation algorithm for JavaScript code.

```
Input: Мутируемое AST tree,
       Пул фрагментов регрессионных тестов P
Output: Мутированное AST mutatedTree, Множество ошибок E
Function mutateNodes(tree)
E = ∅
fragSeq ← GetFrags(tree)
replacedIdx = getRandomInt(Length(fragSeq))
trimmedTree, fragType = RemoveNode(fragSeq, replacedIdx)
newFrag = RandomChoice(P(fragType))
mutatedTree ← ReplaceNode(trimmedTree, newFrag)
return mutatedTree, E
```

Рис. 6. Псевдокод алгоритма мутации узла AST JavaScript кода.
Fig. 6. Pseudocode of the AST node mutation algorithm for JavaScript code.

```
Input: Мутируемое AST tree, множество AST для вставки T
Output: Мутированное AST mutatedTree, Множество ошибок E
Function mutateSubtrees(tree)
E = ∅
// Случайный выбор дерева для вставки
sourceTree = RandomChoice(T)
count ← GetFrags(tree) // Вычисление числа узлов дерева
// Выбор места для вставки
replacedIdx ← getRandomInt(count)
for node ∈ tree do
    count + +
    if mutationFrag == True then
        break// Вставка реализована, выход из цикла
    else
        if count == replacedIdx then
            // Получение подходящего узла
            newNode, sourceTree = getNode(node.type)
            // Подготовка к вставке узла
            prepareNodeForInsertion(newNode, sourceTree)
            mutationFrag = True
            break
        end
        mutatedTree = tree ∪ {newNode}
        return mutatedTree, E
    end
end
```

Рис. 7. Псевдокод алгоритма мутации объединения AST JavaScript кода.
Fig. 7. Pseudocode of the union mutation algorithm for JavaScript code AST.

4. Оценка эффективности представленного способа

Для проверки эффективности предложенных стратегий было собрано 49475 файлов регрессионных тестов различных интерпретаторов. Для лучшей работы фаззера AFL данные были предварительно проверены на синтаксическую корректность, а также сжаты утилитой afl-cmin для интерпретаторов JavaScriptCore и v8 до 247 и 81 файлов соответственно, на которых в итоге проводилось тестирование.

4.1 Эффективность разработанной стратегии обрезки

В табл. 1 представлено сравнение соотношения тестовых входных данных, которые являются валидными (грамматически допустимыми) после обрезки с использованием встроенной обрезки в AFL и разработанным способом.

В численном отношении разработанная стратегия увеличила коэффициент достоверности грамматики входных данных для v8 и JavaScriptCore с 74,1%, 83,7% до 89.7%, 97.3% что может увеличить вероятность эффективного дальнейшего применения мутации. И тем самым повысить эффективность создания тестовых входных данных, которые могут инициировать новое покрытие.

Таким образом, разработанная стратегия обрезки с учетом грамматики может улучшить коэффициент достоверности грамматики для тестовых входных данных после обрезки, что облегчает дальнейшую мутацию.

Табл. 1. Результаты оценки эффективности предложенной стратегии обрезки
Table 1. Results of evaluating the effectiveness of the proposed trimming strategy

Тестируемый интерпретатор	**Процент валидности**	
	AFL++ (%)	**Обрезка AST** (%)
V8	74.1	89.7
JavaScriptCore	83.7	97.3

4.2 Эффективность разработанной стратегии мутации

Эффективность разработанной стратегии мутации демонстрируется графиками скорости обнаружения новых путей в коде интерпретаторов на рис. 8-9. Что наглядно демонстрирует увеличение скорости при применении встроенного модуля мутации.

Число обнаруженных воспроизводимых уникальных зависаний протестированных интерпретаторов представлено в табл. 2.

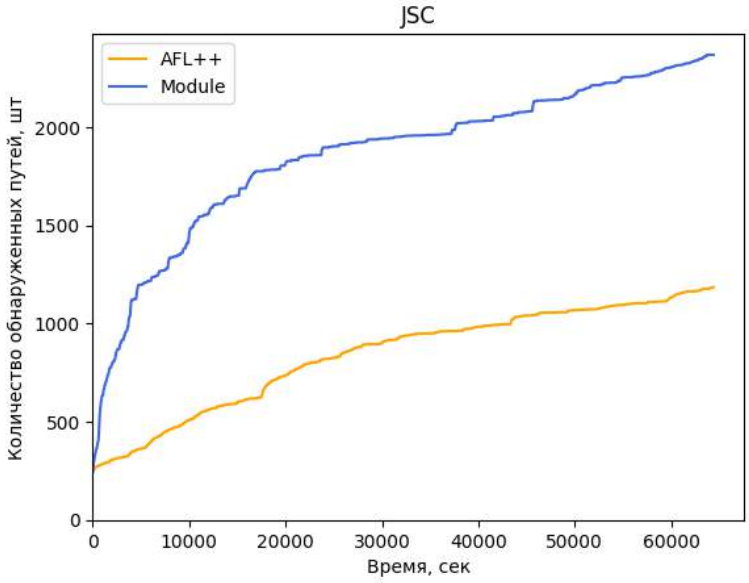


Рис. 8. Графики скорости обнаружения новых путей в коде в интерпретаторе JSC.
Fig. 8. Graphs of the speed of discovering new paths in code in the JSC engine.

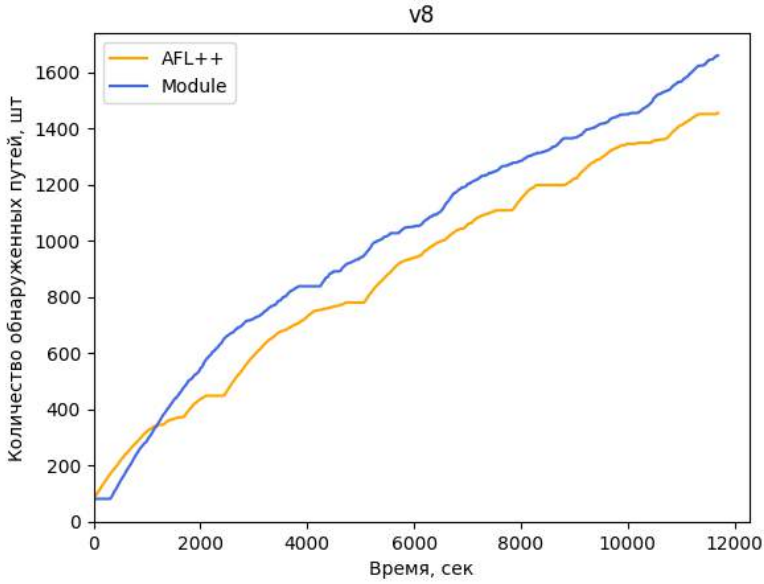


Рис. 9. Графики скорости обнаружения новых путей в коде в интерпретаторе v8.
Fig. 9. Graphs of the rate of discovery of new paths in the code in the v8 engine.

Табл. 2. Сравнение количества обнаруженных зависаний
Table 2. Comparison of number of detected hangs

	JSC	**V8**
AFL++	82	52
Module	500+	322

5. *Заключение*

Проблема неэффективных мутаций является одной из ключевых в фаззинге программного обеспечения, обрабатывающего сложноструктурированные входные данные, такие как программный код. Необходимость сохранения синтаксиса и семантики JavaScript кода требует изменения текущего подхода к обрезке и мутации кода. Мутации на уровне AST-деревьев позволяют производить эффективные мутации, сохраняя требуемую семантику для эффективного обнаружения новых путей в тестируемом коде.

Список литературы / References

[1]. Козачок, А. В., Козачок, В. И., Осипова, Н. С., Пономарев, Д. В. Обзор исследований по применению методов машинного обучения для повышения эффективности фаззинг-тестирования // Вестник ВГУ. Серия: Системный анализ и информационные технологии, 2021 (4), С. 83-106, DOI: 10.17308/sait.2021.4/3800.

[2]. Groß S. FuzzIL: Coverage guided fuzzing for javascript engines // Department of Informatics, Karlsruhe Institute of Technology, 2018.

[3]. H. Han, D. Oh, and S. K. Cha. Codealchemist: Semantics-aware code generation to find vulnerabilities in javascript engines. In Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, Feb. 2019

[4]. Christian Holler, Kim Herzig, and Andreas Zeller. 2012. Fuzzing with code fragments. In Proceedings of the 21st USENIX Security Symposium (USENIX Security). 445–458. https://doi.org/10.5555/2362793.2362831.

[5]. Wang J, Chen B, Wei L, Liu Y. Skyfire: Data-Driven Seed Generation for Fuzzing. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE; 2017 – с. 579–94. DOI: 10.1109/SP.2017.23.

[6]. Aschermann C. et al. NAUTILUS: Fishing for Deep Bugs with Grammars //NDSS. – 2019, DOI: 10.14722/ndss.2019.23xxx.
[7]. Junjie Wang, Bihuan Chen, Lei Wei, and Yang Liu. 2019. Superion: grammar-aware greybox fuzzing. In Proceedings of the 41st International Conference on Software Engineering (ICSE). 724–735. https: //doi.org/10.1109/ICSE.2019.00081.
[8]. Fioraldi A. et al. AFL++ combining incremental steps of fuzzing research //Proceedings of the 14th USENIX Conference on Offensive Technologies. – 2020. – p. 10.
[9]. Козачок А. В., Николаев Д. А., Ерохина Н. С. ПОДХОДЫ К ОЦЕНКЕ ПОВЕРХНОСТИ АТАКИ И ФАЗЗИНГУ ВЕБ-БРАУЗЕРОВ //Вопросы кибербезопасности. – 2022. – №. 3 (49). – С. 32-43, DOI: 10.21681/2311-3456-2022-3-32-43.
[10]. Lee S. et al. Montage: A neural network language model-guided javascript engine fuzzer //Proceedings of the 29th USENIX Conference on Security Symposium. – 2020. – С. 2613-2630, DOI: 10.48550/arXiv.2001.04107.
[11]. Gopinath R., Görz P., Groce A. Mutation analysis: Answering the fuzzing challenge //arXiv preprint arXiv:2201.11303. – 2022, DOI: 10.48550/arXiv.2201.11303.
[12]. Старцев Е. В. Разработка алгоритмов и моделирование динамической типизации в программах для технических систем: дис. – URL: http://www.csu.ru/scientific-departments/PublishingImages/D21229602/startsev_ev/Диссертация (Старцев ЕВ).pdf – С. 73-86, 2015.

Информация об авторах / Information about authors

Наталья Сергеевна ЕРОХИНА является сотрудником Академии ФСО России. Её научные интересы включают информационную безопасность, фаззинг-тестирование, алгоритмы машинного обучения.

Natalya EROKHINA is employee in Academy of the Federal Guard Service of Russian Federation. Her research interests include information security, fuzzing testing, and machine learning algorithms.

DOI: 10.15514/ISPRAS-2023-35(5)-5

Разработка методов автоматизации высокоуровневого моделирования сетей на кристалле

А.А. Американов, ORCID: 0000-0002-5970-2125 <*aamerikanov@hse.ru*>
Т.В. Таржанов, ORCID: 0009-0003-5847-1702 <*tvtarzhanov@edu.hse.ru*>
И.И. Романова, ORCID: 0000-0002-2047-4225 <*iromanova@hse.ru*>
А.Ю. Романов, ORCID: 0000-0002-9410-9431 <*a.romanov@hse.ru*>

Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20.

Аннотация. В статье проведен анализ существующих методов для оптимизации временных затрат и вычислений при высокоуровневом моделировании сетей на кристалле. Приведено описание параметров и характеристик сетей на кристалле, рассчитываемых различными моделями, и проанализировано их влияние на скорость высокоуровневого моделирования. Проведена адаптация существующих методов оптимизации моделирования для внедрения в систему автоматизации проектирования сетей на кристалле.

Ключевые слова: САПР; сеть на кристалле; автоматизация проектирования; высокоуровневое моделирование.

Для цитирования: Американов А.А., Таржанов Т.В., Романова И.И., Романов А.Ю. Разработка методов автоматизации высокоуровневого моделирования сетей на кристалле. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 67–80. DOI: 10.15514/ISPRAS–2023–35(5)–5.

Благодарности: Исследование осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ.

Development of Methods for Automating High-Level Modeling of Networks-On-Chip

A.A. Amerikanov 0000-0002-5970-2125 <*aamerikanov@hse.ru*>
T.V. Tarzhanov, ORCID: 0009-0003-5847-1702 <*tvtarzhanov@edu.hse.ru*>
I.I. Romanova, ORCID: 0000-0002-2047-4225 <*iromanova@hse.ru*>
A.Y. Romanov, ORCID: 0000-0002-9410-9431 <*a.romanov@hse.ru*>

HSE University,
20, Myasnitskaya st., Moscow, 101000, Russia.

Abstract. The paper analyzes the existing methods to optimize the time costs and increase the accuracy of calculations in the high-level simulation of networks-on-chip. The description of parameters and characteristics of networks-on-chip calculated by different models is given, and their influence on the speed of high-level

simulation is analyzed. Adaptation of existing methods of modeling optimization for implementation in the system of automation of networks-on-chip design is carried out.

Keywords: CAD; networks-on-chip; design automation; high-level modeling.

For citation: Amerikanov A.A., Tarzhanov T.V., Romanova I.I., Romanov A.Y. Development of methods for automating high-level modeling of networks-on-chip. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 67-80 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-5.

Acknowledgements: This article is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University).

1. Введение

Применение сетей на кристалле (СтнК) является одним из наиболее перспективных подходов к организации подсистемы связи для передачи данных между различными узлами многоядерных вычислительных систем на чипе. В работе [1] СтнК используется для решения проблем, ограничивающих эффективность связи между элементами системы на кристалле. В другой работе [2] дан пример использования СтнК как подсистему связи для маршрутизатора.

В этой связи возникает необходимость организовать эффективную и качественную разработку СтнК. Процесс проектирования сетей на кристалле состоит из 6 основных этапов [3]: составление технического задания; проектирование; высокоуровневое моделирование; низкоуровневое моделирование; прототипирование или косимуляция; этап производства. Данная работа посвящена разработке методов для оптимизации этапа высокоуровневого моделирования, который необходим для анализа эффективности моделируемой сети. На этом этапе с помощью различных моделей по заданным входным параметрам рассчитываются характеристики СтнК и оценивается максимальная пропускная способность, которой способна будет достигнуть сеть.

Одной из основных проблем высокоуровневого моделирования СтнК является то, что в случае моделирования больших СтнК или нескольких сетей одновременно требуется значительное количество вычислительных мощностей, и такое моделирование может занимать слишком много времени. Также существует риск того, что высокоуровневое моделирование будет недостаточно точным, и неверный расчет характеристик может привести к критическим ошибкам при последующих этапах разработки СтнК. Таким образом, требуется разработка методов для оптимизации временных при высокоуровневом моделировании СтнК, а также внедрение этих методов в систему автоматизации проектирования (далее – САПР) СтнК [4]. Для этого в данной статье описано решении следующих задач: результаты анализа существующих методов для оптимизации временных затрат моделирования; разработка новых методов для оптимизации временных затрат моделирования; внедрение разработанных методов для оптимизации временных затрат моделирования в САПР СтнК и результаты их тестирования.

2. Анализ существующих методов оптимизации временных затрат моделирования

2.1 Анализ особенностей высокоуровневого моделирования

Одной из основных задач высокоуровневого моделирования СтнК является проведение исследования поведения сети при обмене пакетами данных между различными элементами сети и вычисление максимальной пропускной способности, которой способна достичь эта сеть [5]. В отличие от низкоуровневого моделирования [6] среднее время высокоуровневого моделирования значительно меньше и позволяет моделировать более сложные сети большого размера.

При высокоуровневом моделировании необходимо учесть топологию сети, так как топология является ключевым свойством СтнК. Основные характеристики топологии – это количестве вершин (роутеров), количество физических соединений между роутерами, количество ребер, диаметр графа и среднее расстояние среди наиболее коротких путей между всеми узлами графа. Чем меньше количество ребер, тем меньше затраты ресурсов, а чем меньше среднее расстояние путей графа и его диаметр, тем быстрее пакеты достигают цели [7].

Еще одним важным фактором, влияющим на производительность, является алгоритм маршрутизации пакетов [8]. Чаще всего данный алгоритм задается матрицей маршрутизации, которая определяет из какого порта и по какому маршруту нужно двигаться из текущего узла к узлу назначения. От таблицы маршрутизации напрямую зависит, насколько эффективно будет работать СтнК, поэтому важно правильно ее моделировать и учесть все нюансы. Все алгоритмы маршрутизации можно классифицировать на статические и адаптивные [9]. В статических алгоритмах маршрутизации маршрут, по которому проходят пакеты данных, определяется только источником и роутером назначения и никак не учитывает текущее состояние сети в отличие от адаптивной маршрутизации, которая в свою очередь учитывает все изменения, происходящие в сети.

Также на скорость высокоуровневого моделирования СтнК значительное влияние оказывают такие параметры, как количество узлов сети, размер флита (логическая единица информации), скорость генерации флитов, размер пакета данных, размер буферов и распределение трафика.

Как правило, параметры моделей задаются в виде конфигурационного файла, который содержит в себе информацию о топологии сети, алгоритме передачи пакетов, а также все другие необходимые параметры. Соответственно рассчитанные при высокоуровневом моделировании характеристики тоже, как правило, записываются в какой-то итоговый выходной файл. Например, в модели Universal On-Chip Network Simulator (UOCNS) [10] в конфигурационном файле задаются следующие параметры: топология (ячеистая, тор, циркулянт, оптимальный циркулянт); аргументы топологии (количество узлов по вертикали и горизонтали для ячеистой топологии и тора; общее количество узлов, шаг меньшего генератора и шаг большего генератора для циркулянта; общее количество узлов для оптимального циркулянта); алгоритм маршрутизации (алгоритм XY для ячеистой топологии и тора; алгоритм Дейкстры, алгоритм маршрутизации по часовой стрелке или адаптивный алгоритм для циркулянта и оптимального циркулянта) [11]; количество виртуальных каналов; размер буфера виртуального канала (флиты); размер флита (биты); средняя длина пакета (флиты); фиксированный размер пакета (да/нет); средний период формирования пакетов (циклы); количество полученных пакетов, при котором заканчивается симуляция; количество полученных пакетов, при котором заканчивается период разогрева сети; количество работающих симуляторов.

Выходные данные записываются в итоговый файл, который содержит следующие характеристики моделирования: время моделирования (циклы); количество отправленных пакетов; количество полученных пакетов; количество ошибок генерации пакетов; частота генерации пакетов (пакеты/цикл); частота генерации флитов (флиты/цикл/ядро); время доставки пакета (циклы); количество хопов пакетов; пропускная способность сети (флитов/цикл); пропускная способность маршрутизатора (флитов/цикл); загрузка буферов принимающего узла (%); загрузка буферов передающего узла (%); загрузка буферов принимающего маршрутизатора (%); загрузка буферов передающего маршрутизатора (%); загрузка буферов сети (%); загрузка физических сетевых каналов (%).

Точность высокоуровневого моделирования напрямую зависит от правильности входных параметров, а также от проработанности модели. На скорость моделирования из всех входных параметров наиболее значительно влияет количество узлов, далее идет алгоритм маршрутизации, а затем топология, но их влияние уже не так ярко выражено.

2.2 Исследование зависимости входных параметров и выходных характеристик моделей

Выходных характеристик, которые могут рассчитывать модели СтнК при высокоуровневом моделировании существует значительное количество. Среди них можно выделить следующие основные характеристики: количество отправленных и принятых пакетов данных; время передачи пакета; время передачи флита; количество хопов (процесс передачи пакета между узлами); загруженность буферов маршрутизаторов; пропускная способность сети.

Некоторые модели могут иметь свои индивидуальные уникальные характеристики. Например, модель Topaz [11] умеет вычислять такую характеристику, как average distance - среднее расстояние между узлами сети.

Так как одной из основных задач высокоуровневого моделирования является определение наиболее эффективной конфигурации СтнК, что определяется максимальной пропускной способностью, за ключевой вычисляемый параметр возьмем частоту генерации пакетов данных, в которой достигается эта максимальная пропускная способность. Эта точка называется точкой насыщения сети.

2.3 Сравнительный анализ существующих методов для оптимизации временных затрат

Обычно целью моделирования является поиск максимальной пропускной способности СтнК, Максимальной пропускной способностью является точка перегиба, при которой скорость генерации флитов пакета перестает быть равной пропускной способности сети.

В работе [12] рассмотрены большинство существующих методов для оптимизации временных затрат. Выделим основные из них, наиболее подходящие для решения поставленных задач. Далее приведен обзор каждого из них.

Последовательный поиск [13]. Данный метод является наиболее простым, но не самым эффективным. Он заключается в том, чтобы последовательно просматривать весь массив данных с определенным интервалом (шагом) до момента нахождения нужного значения [14]. Преимуществом данного метода является то, что он прост и может работать на любых даже неупорядоченных массивах данных без каких-либо дополнительных условий. Недостатки – медленное выполнение поиска при большом объеме данных, отсутствие масштабируемости.

Генетический поиск [15]. Это эвристический алгоритм поиска, заключающийся в случайном подборе и комбинировании искомых параметров с использованием механизмов, похожих на естественный отбор в природе. Особенно хорошо такой метод поиска показывает себя в многомерных пространствах поиска. К недостаткам генетического поиска относятся плохая масштабируемость в зависимости от сложности решаемой задачи и тенденция сходиться к локальному оптимуму.

Метод Монте-Карло [16]. Суть метода заключается в следующем: процесс описывается математической моделью с использованием генератора случайных величин, модель многократно обсчитывается, и, на основе полученных данных, вычисляются вероятностные характеристики рассматриваемого значения. Метод Монте-Карло позволяет достичь любой необходимой точности результатов, но недостатками данного метода являются большое количество времени, затрачиваемое на поиск, высокая техническая сложность, невозможность адекватно моделировать события с очень высокой или очень низкой вероятностью появления.

Бинарный (двоичный) поиск [17]. Алгоритм поиска заданного значения, осуществляемый путем неоднократного деления массива данных на две части таким образом, что искомый элемент попадает в одну из этих частей. Преимуществами бинарного поиска являются высокая скорость нахождения результата и низкая трудоемкость. Недостаток бинарного

поиска состоит в том, что, если массив данных неупорядоченный, перед использованием бинарного поиска его необходимо отсортировать.

Метод золотого сечения (поиск Фибоначчи) [18]. Это метод поиска заданной переменной на целевом отрезке. В основе метода лежит принцип деления изначального массива данных в пропорциях золотого сечения. Достоинством методом является высокая скорость нахождения результата, так как целевой отрезок быстро уменьшает область поиска данных по сравнению с изначальным массивом [19]. Недостатком является невозможность отыскать все экстремумы, так как если значений, удовлетворяющих заданному условию, несколько, то одно из них может оказаться за пределами целевого отрезка.

2.4 Анализ возможности применения существующих методов при высокоуровневом моделировании СтнК

Последовательный поиск является наиболее простым в реализации методом, подходящим для решения любых задач, поэтому может быть применим в высокоуровневом моделировании СтнК.

Генетический поиск плохо подходит для применения в высокоуровневом моделировании СтнК, так как в большинстве случаев он будет сходиться к локальному оптимуму, и результат не будет являться корректным. Также этот алгоритм лучше всего работает в многомерных пространствах поиска [20], а в задаче поиска точки насыщения массив данных является одномерным.

Метод Монте-Карло подходит для решения задачи поиска насыщения, но является технически сложным в реализации и не будет показывать надежного результата, так как использует случайные числа, среди которых долгое время может не появляться искомого значения.

Недостатком бинарного поиска является то, что он применим только в упорядоченных массивах данных. Этот недостаток при высокоуровневом моделировании не является существенным, так как частоты генерации данных моделируются последовательно, и их массив всегда будет упорядочен. Поэтому бинарный поиск может быть применим для нахождения точки насыщения.

Метод золотого сечения применим для высокоуровневого моделирования СтнК, так как его единственным ограничением является невозможность отыскать все нужные заданные значения, если их несколько. Но поиск максимальной пропускной способности сети – это поиск одной точки насыщения.

3. Разработка методов для оптимизации временных затрат моделирования

3.1 Проектирование методов оптимизации моделирования

Целью моделирования СтнК является поиск точки насыщения моделируемой сети. Для достижения этой цели всего были спроектированы три метода оптимизации моделирования, или поиска точки насыщения.

3.1.1 Последовательный поиск с фиксированным шагом (constant step)

Данный метод заключается в том, что перед началом процесса моделирования задается начальное значение частоты генерации данных, шаг и количество шагов (n). Далее происходят итерации путем добавления значения заданного пользователем шага к предыдущему значению частоты генерации данных. На первой итерации предыдущим значения частоты генерации данных является заданная пользователем начальная частота.

На каждой итерации рассчитывается пропускная способность сети на новом значении частоты генерации данных и сравнивается отношение значения частоты к значению пропускной способности. Такие итерации повторяются n раз, результаты каждой итерации сохраняются в формате словаря, ключами которого являются частоты генерации данных, а значениями – пропускные способности для этих частот.

По окончании итераций словарь с результатами анализируется и последний ключ словаря, чье отношение к значению больше или равно 0,9, является искомой точкой насыщения моделируемой сети.

3.1.2 Бинарный поиск (binary search)

Данный метод заключается в том, что перед началом процесса моделирования задается начальное и конечное значения частоты генерации данных и количество итераций (n). Расстояние от начальной частоты генерации до конечной частоты генерации данных считается целевым отрезком.

Далее на каждой итерации находится новое значение частоты генерации данных путем вычисления среднего значения между начальной и конечной частотами с округлением в меньшую сторону. Затем на вычисленной частоте рассчитывается пропускная способность сети и их значения сравниваются, если их отношение больше или равно 0,9, то вычисленная частота становится новым начальным значением частоты, то есть сдвигается левая граница целевого отрезка, иначе вычисленная частота становится новым конечным значением частоты, то есть сдвигается правая граница целевого отрезка. Такие итерации повторяются n раз.

Последняя вычисленная частота генерации данных является искомой точкой насыщения моделируемой сети.

3.1.3 Метод поиска золотого сечения (golden ratio)

Данный метод заключается в том, что перед началом процесса моделирования задается начальное и конечное значения частоты генерации данных и количество итераций (n). Расстояние от начальной частоты генерации до конечной частоты генерации данных считается целевым отрезком.

Далее на каждой итерации находится 2 новых значения частот генерации по формулам (1) и (2):

$$x1 = max - \frac{(max - min)}{\Phi}, \quad (1)$$

$$x2 = min + \frac{(max - min)}{\Phi}, \quad (2)$$

где $x1, x2$ – новые вычисленные значения частот генерации данных, max – заданное конечное значение частоты генерации данных, min – заданное начальное значение частоты генерации данных, $\Phi = 1{,}618$ – константа, равная пропорции золотого сечения (число Фибоначчи).

Затем на двух вычисленных частотах рассчитываются пропускные способности сети и их значения сравниваются. Если отношение пропускной способности к частоте в точке $x1$ меньше 0,9, то частота генерации данных $x1$ становится новым конечным значением, целевой отрезок – $[min, x1]$. Иначе если отношений пропускной способности к частоте в точке $x2$ больше или равно 0,9, то частота генерации данных $x2$ становится новым начальным значением частоты, целевой отрезок – $[x2, max]$. Иначе целевой отрезок – $[x1, x2]$. Такие итерации повторяются n раз.

Середина последнего вычисленного целевого отрезка является искомой точкой насыщения моделируемой сети.

3.2 Улучшение методов оптимизации моделирования и оценка их погрешности

Недостатком разработанных методов оптимизации является невозможность определения необходимого количества итераций для нахождения точки насыщения перед началом моделирования, из-за чего количество итераций приходится находить путем подбора. Чтобы решить эту проблему, решено сделать так, чтобы методы сами определяли необходимое им количество итераций, и вместо того, чтобы подбирать и задавать количество итераций, будет задаваться максимально допустимое для метода расхождение значений в процентах, то есть точность вычислений.

3.2.1 Самоостанавливающийся последовательный поиск с фиксированным шагом (smart constant step)

Данный метод заключается в том, что перед началом процесса моделирования задается начальное значение частоты генерации данных, шаг и максимально допустимое расхождение значений ($accuracy$). Далее происходят итерации путем добавления значения заданного пользователем шага к предыдущему значению частоты генерации данных. На первой итерации предыдущим значения частоты генерации данных является заданная начальная частота.

На каждой итерации рассчитывается пропускная способность сети на новом значении частоты генерации данных и сравнивается отношение значения частоты к значению пропускной способности. Итерации будут повторять до тех пор, пока полученное таким образом отношение не станет меньше, чем заданное пользователем максимально допустимое расхождение значений.

Последняя вычисленная частота генерации данных является искомой точкой насыщения моделируемой сети. Погрешность вычислений данного метода равна шагу, заданному в начале моделирования.

3.2.2 Самоостанавливающийся бинарный поиск (smart binary search)

Данный метод заключается в том, что перед началом процесса моделирования задается начальное и конечное значения частоты генерации данных и максимально допустимое расхождение значений ($accuracy$). Расстояние от начальной частоты генерации до конечной частоты генерации данных считается целевым отрезком.

На первой итерации находится новое значение частоты генерации данных путем вычисления среднего значения между начальной и конечной частотами с округлением в меньшую сторону. Затем на вычисленной частоте рассчитывается пропускная способность сети, и их значения сравниваются, если их отношение больше или равно заданной $accuracy$, то вычисленная частота становится новым начальным значением частоты, то есть сдвигается левая граница целевого отрезка, иначе вычисленная частота становится новым конечным значением частоты, то есть сдвигается правая граница целевого отрезка. Затем сдвигается та же граница целевого отрезка, пока результат сравнения вычисленной частоты генерации данных и пропускной способности не станет другим, после этого сдвигается другая граница целевого отрезка. Такие итерации повторяются, пока результат сравнения не совпадет с первоначальным.

Последняя вычисленная частота генерации данных является искомой точкой насыщения моделируемой сети. Погрешность вычислений данного метода вычисляется по формуле (3):

$$err = max - \frac{(max - min)}{2^{n-2}}, \tag{3}$$

где err – погрешность вычислений, max – заданное конечное значение частоты генерации данных, min – заданное конечное значение частоты генерации данных, n – количество итераций, которые потребовались методу.

3.2.3 Самоостанавливающийся метод поиска золотого сечения (smart golden ratio)

Данный метод заключается в том, что перед началом процесса моделирования задается начальное и конечное значения частоты генерации данных и максимально допустимое расхождение значений ($accuracy$). Расстояние от начальной частоты генерации до конечной частоты генерации данных считается целевым отрезком.

Далее на каждой итерации находится 2 новых значения частот генерации по формулам (1) и (2). Затем на вычисленных двух частотах рассчитываются пропускные способности сети и их значения сравниваются. Если отношение пропускной способности к частоте в точке $x1$ меньше $accuracy$, то частота генерации данных $x1$ становится новым конечным значением, целевой отрезок – $[min, x1]$. Иначе если отношений пропускной способности к частоте в точке $x2$ больше или равно $accuracy$, то частота генерации данных $x2$ становится новым начальным значением частоты, целевой отрезок – $[x2, max]$. Иначе целевой отрезок – $[x1, x2]$. Когда искомая точка попадает в $[x1, x2]$, итерации повторяются еще, пока эта точка не выйдет за границы отрезка $[x1, x2]$.

Середина последнего вычисленного целевого отрезка является искомой точкой насыщения моделируемой сети. Погрешность вычислений данного метода вычисляется по формуле (4):

$$err = \left(1 - \frac{1}{\Phi}\right) \cdot \left|\frac{(max - \min)}{x}\right|, \quad (4)$$

где err – погрешность вычислений, max – заданное конечное значение частоты генерации данных, min – заданное начальное значение частоты генерации данных, $\Phi = 1{,}618$ – константа, равная пропорции золотого сечения (число Фибоначчи), x – середина последнего вычисленного целевого отрезка (точка насыщения).

4. Внедрение методов оптимизации временных затрат в САПР СтнК и их тестирование

Предложенные в предыдущей главе методы оптимизации высокоуровневого моделирования СтнК были реализованы программно и внедрены в САПР СтнК. Была проведена их апробация и выполнен анализ их эффективности [21]. Далее рассмотрены способы ускорения вычислений, которые могут быть применены совместно с разработанными методами.

4.1 Параллельный запуск вычислений

Для оптимизации временных затрат при высокоуровневом моделировании СтнК в САПР было разработано метод для запуска множества моделей не последовательно, а параллельно с помощью асинхронного программирования, что дает существенный прирост в скорости моделирования. Расчеты каждой из моделей было решено запускаются асинхронно.

Для реализации асинхронности вычислений для оптимизации временных затрат использованы средства языка C# (Task<T>, async и await), которые позволяют выполнять разработку асинхронного программного кода без использования обратных вызовов и загрузки сторонних библиотек [22].

Принцип работы асинхронности вычислений, реализованной в САПР СтнК, представлен на рис. 1.

На 1 стадии задачи Launch происходит распараллеливание работы нескольких моделей с различными входными параметрами. Каждое моделирование на 2 стадии асинхронно запускает сразу несколько итераций вычисления характеристик. На 3 стадии реализована проверка наличия результатов расчетов в базе данных, и если они там имеются, то САПР СтнК не тратит лишние ресурсы на проведение вычисления, а сразу достает результат из базы данных [23], что также положительно влияет на уменьшение времени моделирования.

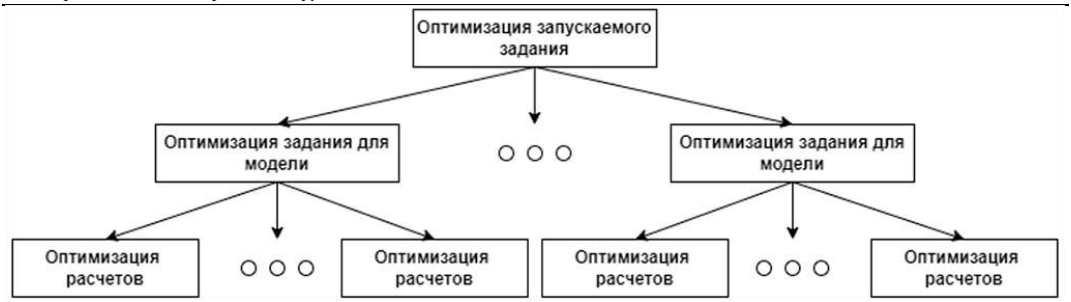


Рис. 1. Асинхронный запуск вычислений.
Fig. 1. Asynchronous calculation start.

4.2 Тестирование внедренных методов оптимизации

Для проверки работоспособности разработанных методов были разработаны сценарии их тестирования и проведено полное их тестирования в специальном программном окружении и в одинаковых условиях. Все тесты были пройдены успешно.

4.3 Анализ эффективности внедренных методов оптимизации

Для анализа эффективности внедренных методов оптимизации временных затрат при высокоуровневом моделировании была взята модель BookSim [24] со следующими входными параметрами: топология: ячеистая; аргументы топологии: 16x2 (256 ядер); функция маршрутизации: dimension-order; количество виртуальных каналов: 4; размер буфера виртуальных каналов: 4; распределение трафика: uniform; размер пакета: 10; тип симуляции: throughput; период выборки: 5000; период разогрева сети: 1; максимальное количество образцов: 10.

Сначала данная модель была запущена без использования САПР СтнК в консоли, в качестве топологии СтнК была выбрана mesh 16x16. Моделирование заняло примерно 44 секунды. Для того чтобы провести 10 различных расчетов с различными значениями итерируемого параметра, пользователю требуется 10 раз последовательно запускать моделирование таким способом. Итого время одного эксперимента заняло около $(44 + t1) \cdot 10$ секунд, где $t1$ – время, необходимое для редактирования конфигурационного файла, передаваемого на вход модели BookSim.

Затем эта же модель с идентичными входными параметрами была запущена в САПР СтнК методом последовательного поиска с константным шагом с 10 итерациями. Результаты моделирования представлены на рис. 2.

В результате все итерации запустились примерно в одно время, выполнялись параллельно с длительностью в среднем примерно равной 43 секундам. Итого время одного эксперимента занимает примерно $43{,}6 + t2$ секунд, где $t2$ – время, необходимое для задания входных параметров модели BookSim в САПР СтнК.

Таким образом, разница во временных затратах между запуском моделирования вручную последовательно и запуском моделирования параллельно с помощью САПР СтнК составила $440 + 10t1 - 43{,}6 + t2$ секунд, где промежуток времени $t1$ примерно равен промежутку времени $t2$.

Графическая демонстрация работы моделирования СтнК моделью BookSim, запущенного в САПР СтнК методом оптимизации с фиксированным шагом, приведена на рис. 3. В соответствии с графиком, точка насыщения сети за 10 итераций так и не была найдена. Необходимо дальше увеличивать количество итераций или подобрать другие входные параметры. Далее было запущено моделирование этой же модели другими

разработанными методами оптимизации с целью оценки того, какой из методов является наиболее эффективным в заданных условиях.

status text	start_time timestamp with time zone	end_time timestamp with time zone	duration interval
Completed	2023-05-06 15:05:54.519633+00	2023-05-06 15:06:37.412424+00	00:00:42.892791
Completed	2023-05-06 15:05:54.528036+00	2023-05-06 15:06:37.544727+00	00:00:43.01669
Completed	2023-05-06 15:05:54.612807+00	2023-05-06 15:06:37.443104+00	00:00:42.830297
Completed	2023-05-06 15:05:54.534117+00	2023-05-06 15:06:37.50613+00	00:00:42.972013
Completed	2023-05-06 15:05:54.49058+00	2023-05-06 15:06:37.392373+00	00:00:42.901793
Completed	2023-05-06 15:05:54.428765+00	2023-05-06 15:06:37.423603+00	00:00:42.994838
Completed	2023-05-06 15:05:54.566677+00	2023-05-06 15:06:37.520846+00	00:00:42.954169
Completed	2023-05-06 15:05:54.352917+00	2023-05-06 15:06:37.364624+00	00:00:43.011707
Completed	2023-05-06 15:05:54.574295+00	2023-05-06 15:06:37.502192+00	00:00:42.927896
Completed	2023-05-06 15:05:54.490212+00	2023-05-06 15:06:37.567618+00	00:00:43.077406

Рис. 2. Моделирование СтнК с помощью BookSim в разработанной САПР СтнК.
Fig. 2. NoC modeling with BookSim model in developed NoC CAD system.

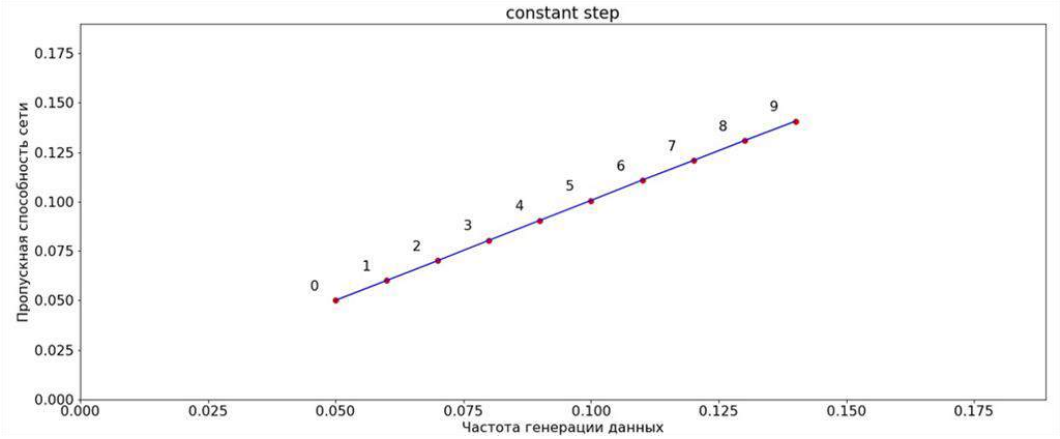


Рис. 3. Метод с фиксированным шагом.
Fig. 3 Constant step method.

Графическая демонстрация работы моделирования СтнК моделью BookSim, запущенного в САПР СтнК методом оптимизации бинарный поиск, приведена на рис. 4. Была задана слишком большая максимальная частота генерации данных, из-за чего произошло несколько бесполезных итераций, но точка насыщения найдена.

Графическая демонстрация работы моделирования СтнК моделью BookSim, запущенного в САПР СтнК методом оптимизации золотого сечения, приведена на рис. 5. Данный метод произвел меньше ненужных итераций и нашел точку насыщения с меньшей погрешностью, чем метод бинарного поиска.

Графическая демонстрация работы моделирования СтнК моделью BookSim, запущенного в САПР СтнК самоостанавливающемся методом оптимизации с фиксированным шагом, приведена на рис. 6. Для нахождения точки насыщения методу понадобилось 15 итераций. Моделирование заняло значительное время.

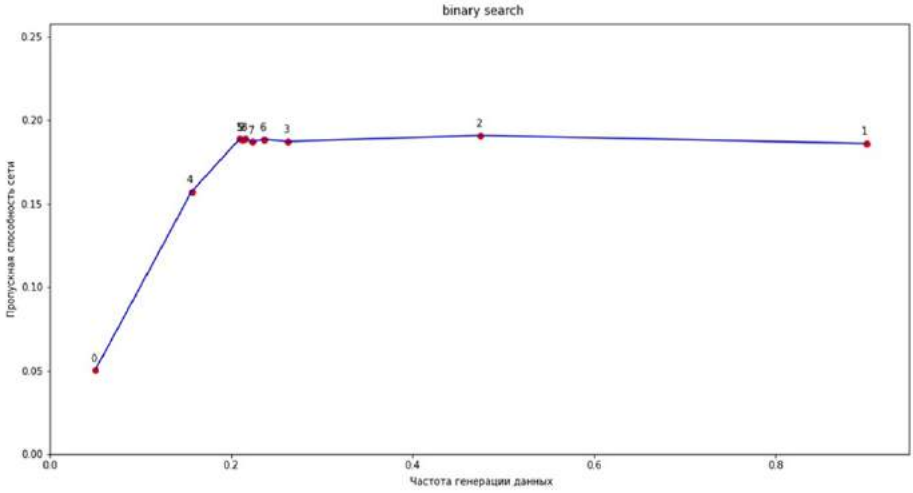


Рис. 4. Метод бинарного поиска .
Fig. 4. Binary search method.

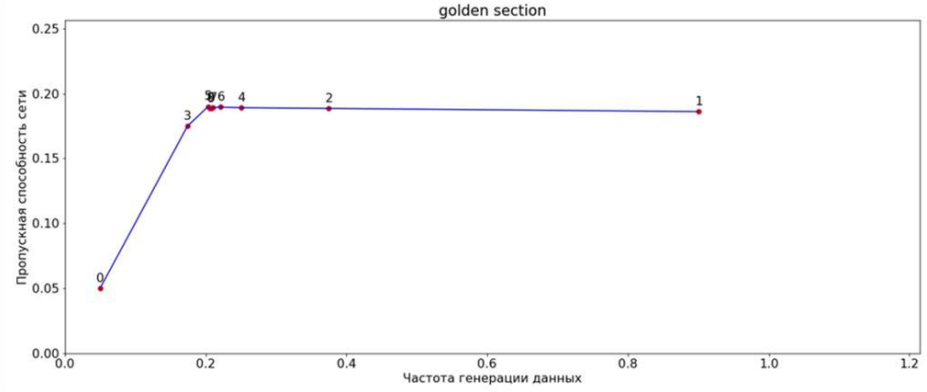


Рис. 5. Метод золотого сечения.
Fig. 5. Golden ratio method.

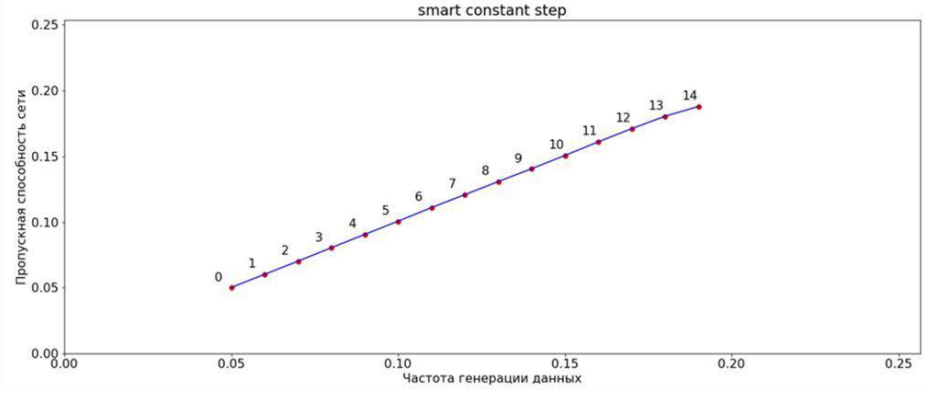


Рис. 6. Метод самоостанавливающегося последовательного поиска с константным шагом.
Fig. 6. Self-stopping sequential search with constant step method.

Графическая демонстрация работы моделирования СтнК моделью BookSim, запущенного в САПР СтнК самовосстанавливающем методом оптимизации бинарного поиска, приведена на рис. 7. Для нахождения точки насыщения методу понадобилось всего 6 итераций, но погрешность расчетов немного выше, чем у других методов.

Графическая демонстрация работы моделирования СтнК моделью BookSim, запущенного в САПР СтнК самоостанавливающемся методом оптимизации Золотого сечения, приведена на рис. 8. Для нахождения точки насыщения методу понадобилось 11 итераций, это меньше, чем у самоостанавливающегося метода с фиксированным шагом, и больше, чем у самоостанавливающегося метода бинарного поиска, но при этом точка насыщения найдена с наименьшей погрешностью.

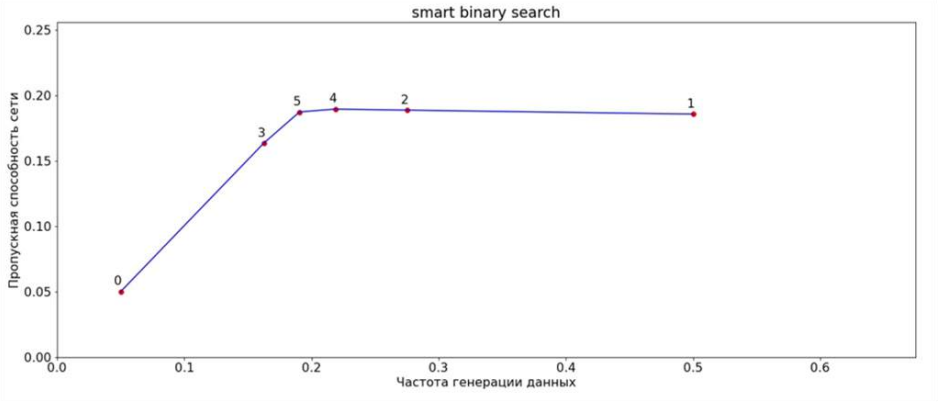


Рис. 7. Самоостанавливающийся метод бинарного поиска.
Fig. 7. Self-stopping binary search method.

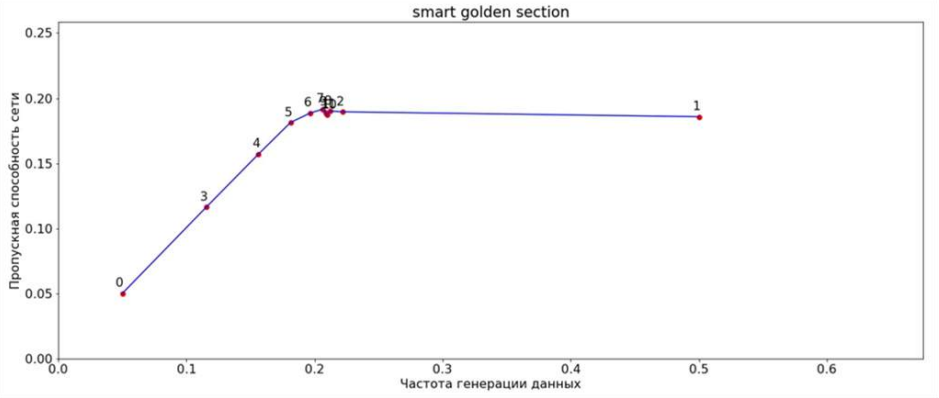


Рис. 8. Самоостанавливающийся метод Золотого сечения.
Fig. 8. Self-stopping Golden ratio method.

5. Выводы

Таким образом, был проведен сравнительный анализ существующих методов для оптимизации временных затрат моделирования, а также анализ возможности применения этих методов при высокоуровневом моделировании СтнК.

В рамках работы выполнена адаптация существующих методов для оптимизации временных затрат при высокоуровневом моделировании СтнК. Эти методы были реализованы программно на языке C# и внедрены в САПР СтнК. Было проведено их функциональное тестирование.

Разработанные методы позволили сократить количество запусков моделирования в несколько раз. Кроме того, в САПР СтнК был реализован режим асинхронного запуска моделирования и вычислений характеристик СтнК, который позволил значительно сократить временные затраты на моделирование, в частности, в рамках выполнения исследования, временные затраты на моделирование сократились до 10 раз.

Список литературы / References

[1]. Kundu S., Chattopadhyay S. Network-on-chip: the next generation of system-on-chip integration. Taylor and Francis, 2014. 389 p.

[2]. Dimitrakopoulos G., Psarras A., Seitanidis I. Microarchitecture of Network-on-chip Routers. Springer, 2015. 175 p.

[3]. Американов А. А. Автоматизация высокоуровневого моделирования сетей на кристалле. Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС), №1, 2021 г., стр. 39–45. / Amerikanov A. A. Avtomatizaciya vysokourovnevogo modelirovaniya setej na kristalle. Problemy razrabotki perspektivnyh mikro-i nanoelektronnyh sistem (MES), 2021, №1, pp. 39–45. (in Russian). DOI: DOI: 10.31114/2078-7707-2021-1-39-45.

[4]. Tarzhanov T.V., Ponomarev A.S., Borodin N.Y. (2023) UHLNoCSim-SE. Доступно по ссылке: https://github.com/asponomarev/cad, 25.10.2023.

[5]. Короткий Е. В., Лысенко А. Н. Метод моделирования реконфигурируемых сетей на кристалле. Вестник Национального технического университета Украины «Киевский политехнический институт», № 51, 2009 г., стр. 221–227. / Korotkij E.V., Lysenko A.N. Metod modelirovaniya rekonfiguriruemyh setej na kristalle. Vestnik Nacional'nogo tekhnicheskogo universiteta Ukrainy "Kievskij politekhnicheskij institut", 2009, № 51, pp. 221–227. (in Russian).

[6]. Лежнев Е. В. Автоматизация низкоуровневого моделирования сетей на кристалле. Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС), № 1, 2021 г., стр. 46–50. / Lezhnev E.V. Automation of Low-Level Modeling of Networks-on-Chip. Problemy razrabotki perspektivnyh mikro-i nanoelektronnyh sistem (MES), 2021, № 1, pp 46–50. (in Russian). DOI: 10.31114/2078-7707-2021-1-46-50.

[7]. Dally W.J., Towles B.P. Principles and practices of interconnection networks. Elsevier, 2004. 554 p.

[8]. Ладыженский Ю. В., Мирецкая В. А. Моделирование алгоритмов маршрутизации в сетях на кристалле. Научные труды Донецкого национального технического университета, 2008 г., № 9, стр. 79–86. / Ladyzhenskij Y.V., Mireckaya V.A. Modeling of routing algorithms in on-chip networks. Nauchnye trudy Doneckogo nacional'nogo tekhnicheskogo universiteta, 2008, № 9, pp. 79–86 (in Russian).

[9]. Palesi M., Daneshtalab M. Routing algorithms in networks-on-chip. Springer, 2014. 410 p.

[10]. Amerikanov A.A., Ponomarev A.S. Universal On-Chip Network Simulator for Networks-on-Chip Developmentю 2021 International Russian Automation Conference (RusAutoCon), 2021, pp. 677–682. DOI: 10.1109/RusAutoCon52004.2021.9537564.

[11]. Abad, P., Prieto, P., Menezo, L. G., Puente, V., & Gregorio, J. Á. TOPAZ: An Open-Source Interconnection Network Simulator for Chip Multiprocessors and Supercomputers. 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip. IEEE. 2012, pp. 99–106. DOI: 10.1109/NOCS.2012.19.

[12]. Аттетков А.В., Зарубин В.С., Канатников А.Н. Введение в методы оптимизации. Общество с ограниченной ответственностью «Научно-издательский центр ИНФРА-М», 2008. 272 с. / Attetkov A.V., Zarubin V.S., Kanatnikov A.N. Vvedenie v metody optimizacii. Obshchestvo s ogranichennoj otvetstvennost'yu "Nauchno-izdatel'skij centr INFRA-M", 2008. 272 p. (in Russian).

[13]. Седжвик Р. Фундаментальные алгоритмы на С. Анализ/Структуры данных/Сортировка/Поиск/Алгоритмы на графах // СПб. Диасофт, 2003. / Sedzhvik R. Fundamental'nye algoritmy na S. Analiz/Struktury dannyh/Sortirovka/Poisk/Algoritmy na grafah // SPb. Diasoft. 2003. (in Russian).

[14]. Кнут Д. Искусство программирования. Том 3. Сортировка и поиск. Litres, 2022. 824 с. / Knut D. Iskusstvo programmirovaniya. Tom 3. Sortirovka i poisk. Litres, 2022. 824 p. (in Russian).

[15]. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. ООО Издательская фирма «Физико-математическая литература», 2010. 366 с. / Gladkov L.A., Kurejchik V.V., Kurejchik V.M. Geneticheskie algoritmy. OOO Izdatel'skaya firma "Fiziko-matematicheskaya literatura", 2010. 366 p. (in Russian).

[16]. Соболь И. М. Метод Монте-Карло. Наука, 1985. 64 с. / Sobol' I. M. Metod Monte-Karlo. Nauka, 1985. 64 p. (in Russian).

[17]. Левитин А.В. Алгоритмы: введение в разработку и анализ. Издательский дом Вильямс, 2006. 575 с. / Levitin A.V. Algoritmy: vvedenie v razrabotku i analiz. Izdatel'skij dom Vil'yams, 2006, 575 p. (in Russian).

[18]. Альхимович М. А. Метод золотого сечения для решения задач минимизации. БНТУ, 2017 г., стр. 122–123 / Al'himovich M.A. Metod zolotogo secheniya dlya resheniya zadach minimizacii. BNTU, 2017, pp. 122–123 (in Russian).
[19]. Subasi M., Yildirim N., Yildiz B. An improvement on Fibonacci search method in optimization theory. Appl. Math. Comput. Elsevier, vol. 147, № 3, 2004, pp. 893–901.
[20]. Курейчик В. М. Генетические алгоритмы и их применение. Таганрогский государственный радиотехнический университет, 2002. 242 с. / Kurejchik V.M. Geneticheskie algoritmy i ih primenenie. Taganrogskij gosudarstvennyj radiotekhnicheskij universitet, 2002. 242 p. (in Russian).
[21]. Рыбалко М. А., Иванова Е. А. Тестирование программного обеспечения, методы тестирования. Информационное общество: современное состояние и перспективы развития, 2017 г., стр. 320–322. / Rybalko M.A., Ivanova E.A. Testirovanie programmnogo obespecheniya, metody testirovaniya. Informacionnoe obshchestvo: sovremennoe sostoyanie i perspektivy razvitiya, 2017, pp. 320–322 (in Russian).
[22]. Дэвис А. Асинхронное программирование в C# 5.0. Litres, 2022. / Devis A. Asinhronnoe programmirovanie v C# 5.0. Litres, 2022 (in Russian).
[23]. Питолин М. В., Мачтаков С. Г. Организация поиска в базе данных на основе интеллектуализации принятия решений. Пожарная безопасность проблемы и перспективы, том 2, № 1 (6), 2015 г., стр. 395–399. / Pitolin M.V., Machtakov S.G. Organizaciya poiska v baze dannyh na osnove intellektualizacii prinyatiya reshenij. Pozharnaya bezopasnost' problemy i perspektivy, 2015, vol. 2, № 1 (6), pp. 395–399 (in Russian).
[24]. Jiang N. Michelogiannakis G., Becker D., Towles B., Dally W.J. Booksim 2.0 user's guide. Standford Univ., 2010. 11 p.

Информация об авторах / Information about authors

Александр Александрович АМЕРИКАНОВ – кандидат технических наук, доцент национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: разработка САПР, разработка логических устройств, ПЛИС, сети на кристалле.

Aleksandr Aleksandrovich AMERIKANOV – Cand. Sci (Tech.), Associate Professor at the HSE University. Research interests: CAD development, development of logic devices, FPGA, networks-on-chip.

Тимофей Владимирович ТАРЖАНОВ – магистр национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: сети на кристалле, разработка САПР, системы на кристалле, методы оптимизации.

Timofei Vladimirovich TARZHANOV – master's degree from the HSE University. Research interests: networks-on-chip, CAD development, systems-on-chip, optimization methods.

Ирина Ивановна РОМАНОВА – старший преподаватель национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: проектирование логических устройств, разработка САПР, сети на кристалле.

Irina Ivanovna ROMANOVA – Senior Lecturer at the HSE University. Research interests: design of logic devices, CAD development, networks-on-chip.

Александр Юрьевич РОМАНОВ – кандидат технических наук, доцент, заведующий лабораторией систем автоматизированного проектирования национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: сети на кристалле, системы на кристалле, нейронные сети, система автоматизированного проектирования.

Aleksandr Yur'yevich ROMANOV – Cand. Sci (Tech.), Associate Professor, Head of the Laboratory of Computer-Aided Design Systems at the HSE University. Research interests: networks-on-chip, systems-on-chip, neural networks, CAD systems.

DOI: 10.15514/ISPRAS-2023-35(5)-6

CC BY

The Open System for Storing and Processing of a Dataset of Combinational Circuits

D.A. Miachin, ORCID: 0009-0009-4538-5286 <*danchikmiachin@gmail.com*>
V.P. Pugach, ORCID: 0009-0003-3260-1011 <*vppugach@edu.hse.ru*>
S.S. Avdeiuk, ORCID: 0009-0000-9183-4497 <*ssavdeyuk@edu.hse.ru*>
V.V. Zunin, ORCID: 0000-0002-9117-4879 <*vzunin@hse.ru*>
A.Y. Romanov, ORCID: 0000-0002-9410-9431 <*a.romanov@hse.ru*>

HSE University,
20, Myasnitskaya st., Moscow, 101000, Russia.

Abstract. This paper presents an open-source software for generation, storage, and analysis of combinational circuits. The previously created methods for generating combinational circuits have been optimized, and a dataset has been formed. The generation of combinational circuits is carried out on various devices. The application implements the possibility to combine the generated datasets into a single storage (Synology Drive), as well as analyze the fault tolerance of combinational circuits using various methods for their evaluation. New possible methods for assessing combinational circuits' reliability using machine learning are proposed.

Keywords: combinational circuits; dataset; reliability; machine learning; client-server application.

For citation: Miachin D.A., Pugach V.P., Avdeiuk S.S., Zunin V.V., Romanov A.Y. The open system for storing and processing of a dataset of combinational circuits. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 81-90. DOI: 10.15514/ISPRAS-2023-35(5)-6.

Открытая система хранения и обработки набора данных комбинационных схем

Д.А. Мячин, ORCID: 0009-0009-4538-5286 <*danchikmiachin@gmail.com*>
В.П. Пугач, ORCID: 0009-0003-3260-1011 <*vppugach@edu.hse.ru*>
С.С. Авдеюк, ORCID: 0009-0000-9183-4497 <*ssavdeyuk@edu.hse.ru*>
В.В. Зунин, ORCID: 0000-0002-9117-4879 <*vzunin@hse.ru*>
А.Ю. Романов, ORCID: 0000-0002-9410-9431 <*a.romanov@hse.ru*>

Национальный исследовательский университет «Высшая школа экономики»,
101000, Россия, г. Москва, ул. Мясницкая, д. 20.

Аннотация. В этой статье представлено программное обеспечение с открытым исходным кодом для генерации, хранения и анализа комбинационных схем. Оптимизированы созданные ранее методы генерации комбинационных схем и сформирован датасет. Генерация комбинационных схем может осуществляться на различных устройствах. В приложении реализована возможность объединения сгенерированных наборов данных в единое хранилище (Synology Drive), а также анализа отказоустойчивости комбинационных схем с использованием различных методов их оценки. Предложены новые возможные методы оценки надежности комбинационных схем с использованием машинного обучения.

Ключевые слова: комбинационные схемы; датасет; отказоустойчивость; машинное обучение; клиент-серверное приложение.

Для цитирования: Мячин Д.А., Пугач В.П., Авдеюк С.С., Зунин В.В., Романов А.Ю. Открытая система хранения и обработки набора данных комбинационных схем. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 81–90 (на английском языке). DOI: 10.15514/ISPRAS–2023–35(5)–6.

1 Introduction

A combinational circuit [1] is a digital circuit that has no memory and (depending on the discrete signals received at the inputs) produces unambiguously defined logic signals at the outputs [2].

In any modern electronic devices, both memory circuits and combinational circuits are used. And, like any other electronic devices, combinational circuits are subject to possible failures and malfunctions. Failure of the circuit can lead to malfunction of the whole device, which (in turn) can lead to serious consequences. That is why fault tolerance of the combinational circuit is one of its key characteristics and indicators of its quality affecting the operation of various electronic devices.

Due to the ubiquitous use of digital circuits, the study and creation of methods for their generation, as well as the estimation of the parameters of such circuits (including fault tolerance [3]), is a relevant topic for scientific research and is of great importance for the development of modern technologies. Thus, there is a need to create open-source software that will allow generating large amounts of data for conducting research on predicting the parameters of combinational circuits using machine learning methods.

Existing fault tolerance evaluation methods (e.g., [3–7]) have a number of drawbacks and problems, which will be described in detail in this paper; optimization of these methods and creation of new ones are relevant at present.

2 Problem statement

Currently, there are no client-server applications for users to conveniently generate large datasets of combinational circuits on various devices with calculation of their basic parameters, while having only an Internet access or connecting computing and storage devices locally. Often, this task requires the installation of specialized software. Also, currently developed programs for generating datasets of combinational circuits [8–10] do not have functionality for visualizing the generated circuits and assessing their fault tolerance, which is the key in analyzing the quality of the circuit.

Thus, the purpose of this study is to develop a unified system for storing and processing a dataset of digital circuits to provide a convenient user interaction with a program for generating circuits using a web interface. The created software provides a user with the ability to select an algorithm for generating combinational circuits, configure the necessary parameters for generation, and also reduce the time for analyzing the fault tolerance of combinational circuits and their other parameters in the future. The developed client-server application allows evaluating the fault tolerance of the generated combinational circuits directly from the application without requiring any installation of additional software, as well as visualize the generated combinational circuits.

One of the additional problems being solved in the course of the study was the problem of optimizing the previously developed software [11] used as the basis for the study – to speed up its work and improve its convenience. The software was also translated (by manual transpilation into C++) from the C# language.

3 Analog analysis

The search and analysis of existing applications and programs for generating combinational circuits showed that at present there are no full-fledged analogues that allow generating combinational circuits using various generation algorithms and estimating their parameters; algorithms for constructing combinational circuits based on truth tables and trees are mainly used.

Consider the programs capable of working with virtual (emulated) combinational circuits: Multisim Workbench [9] and Logicly [10].

To use Multisim Workbench [9], one must manually (graphically) enter the logic elements of each circuit and manually create the circuits themselves. Due to this implementation, the process of placing circuit components becomes more complicated than using code or automated creation. Transferring data in text form to a graphics window complicates program development. Thus, their advantages, the ability to visualize circuits becomes a disadvantage, if automated generation of combinational circuits is required. Also, a big disadvantage of Multisim Workbench is that it is a closed commercial program with a rather high license price.

The main advantage of Logicly [10] is a fairly low entry threshold and a convenient graphic design, but it is limited in that it allows modeling a circuit with only a small number of elements. The program cannot generate combinational circuits based on the given generation parameters, but only allows manual creation of circuits.

Thus, the lack of automation and any integrated means of storing and processing the results of work do not allow the full use of existing solutions for working with combinational circuits; so, the development of new ones is needed.

4 Software architecture

In the course of this study, a client-server application for generating combinational circuits and their aggregation in the selected storage was developed. The client-server architecture was chosen, since this is what will allow setting up several server parts for simultaneous generation of circuits on several devices, which will significantly speed up the circuit generation. Also, if necessary, the server parts can be accessed remotely for ease of interaction.

To provide a simplified procedure for adding new circuit generation methods, a special structure for the interaction of generation methods with the dataset generator was developed. The architecture of the developed client-server application consists of two main components (Fig. 1), the client and the server, interacting with each other using the API.

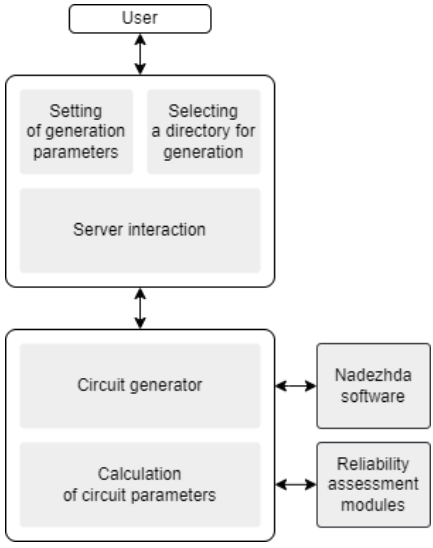


Fig. 1. Architecture of the developed application.

The server can run a generator module associated with various generation methods. Thus, when adding new methods, the principle of the program does not change, and the user and developers have the opportunity to asynchronously modify the software to suit their needs without making large-scale changes to the program. The main advantage of this architecture is the flexibility for future improvements.

The frontend part allows the user to configure parameters for generating combinational circuits and select a directory for their aggregation, while the backend part calls a function to generate combinational circuit datasets, performs visualizations of the generated circuits, and also calculates various circuit parameters, including their fault tolerance using various reliability assessment methods.

The first and main advantage of the developed software is the ability not only to generate combinational circuits, but also to evaluate their fault tolerance within one software, as well as modify it for other tasks.

The second advantage of the developed software is that the client-server application allows using it without in-depth knowledge in the field of programming and circuitry.

The third advantage is that the developed software allows automating the process of filling in the parameters for generation; at the same time, the functionality of automatic visualization of the generated circuits is implemented.

The fourth advantage is the availability of aggregation of generated circuits in one place. The developed generation program allows saving the generated datasets of combinational circuits in the Verilog format, as well as saving the information about each circuit in the JSON format [12]. Also, in the folder with each circuit, there is a PNG visualization generated using the Yosys software [13]. All generated circuits are aggregated in the given storage, for which Synology Drive [14] was selected. It was chosen due to its simple library for interacting with the storage, as well as ease of use.

4.1 Web-component

A web application which interacts with the program for generating combinational circuit datasets and displays the generation results to the user on the site using the server is implemented. The user has the ability to set generation parameters that are stored in the database on the server. The web interface also prevents the user from entering invalid values and controls limits on the generation parameters, depending on the chosen generation method.

After the user sets all the parameters necessary for generating, a JSON request with these parameters is generated; it is sent to the server, which (upon receiving requests from the client) launches the combinational circuit dataset generator for these parameters and returns a link to the location of the generated datasets.

During the generation process, the user can track the progress of the generator program using progress bars. At the end of the program, the web interface receives a request from the server containing a link to the generated datasets divided into different folders (depending on the generation method) and also receives the parameters of the generated datasets, such as the number of circuit inputs and outputs.

Combination circuits are generated asynchronously on the server, which eliminates the need for the user to keep any program or site running. Thus, the user does not have to worry about their safety, since the service is available wherever there is an Internet connection. At the same time, installation of additional software is also not required, as all the interaction takes place in a web browser.

Using a server with generation, there is a potential opportunity to use the developed service from various devices on which each user has a unique account with unique generation data. As a result, all generated circuits will be aggregated in a single remote folder. That is, as a result of creating a web application, the basis for the future project scaling or further integration into other applications is formed.

To create the service, all modern methods applied in web development were used:

- the development was done using the React JS framework;
- the possibility of obtaining up-to-date data and data updating (during the operation of the site using asynchronous requests) was implemented.

4.2 Server component

The server component of the application receives and transmits data from both the web application and the generation program. Interaction with the web application takes place using the API. This is a modern interface for communication between backend and frontend components.

The interaction of the server component with the generation program is carried out using sockets. This interface was chosen because it allows programs developed in different programming languages to exchange data, which increases the convenience of further development. The server is implemented in Python3 using the Django REST framework, while the circuits generator is implemented in C++. This is done in order to be able to use for generation the programs developed in other programming languages, not changing the server part.

For a more convenient demonstration of the generation result to the user, the capability to visualize the generated combinational circuits using the Yosys software [13] was added to the server.

Yosys software creates DOT files from Verilog files generated by the program. Next, using the Graphviz library [15] a schema file in png format is generated from a DOT format file, which has a human-readable form (Fig. 2).

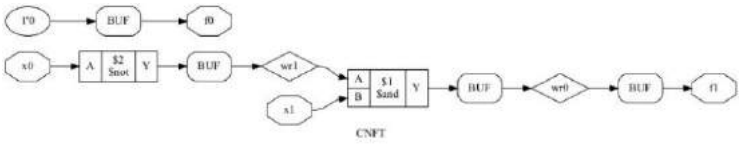


Fig. 2. Example of a generated combinational circuit.

Thus, the user can analyze the graphical interpretation of the circuit, which is located in the directory selected for generation next to the Verilog format file.

The possibility to run the generation program and the server on the same computer is Implemented. Due to the adopted engineering solutions, this does not cause additional problems, unlike the implementation method using system interrupts. There are enough ports on the computer to transfer the data via sockets; so, running programs on one device is not difficult. It also allows scaling the program by running many back-ends for parallel generation of combinational circuits.

4.2.1 Optimization of the generation algorithms

The program for generating circuits allows generating a dataset to study their fault tolerance. In the case of conventional prediction methods, the fault tolerance of the circuits generated is compared with an ideal value, and the predictive power of the new methods is evaluated. It is also possible to use the datasets generated to train standard Machine Learning models.

When training neural networks, a large data set allows (in addition to predicting fault tolerance) generating new circuits corresponding to the same logical function but with greater fault tolerance.

Different methods of generating combinational circuits provide for the presence of different necessary parameters. From the point of view of machine learning, this is useful, as it avoids the homogeneity of circuits and prevents the occurrence of the retraining factor for some specific type of generation.

Four types of generation are currently implemented [16]:

1) In truth tables (Combinational Circuit Generation using Random Truth Tables; CCGRTT).
2) By specifying the number of circuit levels (Combinational Circuits Generation by the Random Connection of Gates; CCGRCG);
3) By setting the number of logical operators in the circuit (Combinational Circuits Generation by Random Vertex Connection; CCGRVC);
4) By using a genetic algorithm (Combinational Circuits Generation based on Genetic Algorithm; CCGGA);

5) By using custom circuit and their post-processing (Combinational Circuits Generation based on the User-Defined Schemes, CCGUDS).

All these generation methods do not correlate with each other and allow obtaining a variety of data. The circuits generated by these generation methods are built on the basis of the implemented directed graph class, which makes it possible to conveniently interact with them inside the program.

CCGRTT method is one of the simplest. For this generation method, the user must specify the desired number of inputs and outputs for the circuit, as well as the method for minimizing logical functions for logical expressions (PDNF and/or PCNF).

CCGRCG generates a combinational circuit depending on the limit on the number of levels (the tree depth) specified by the user and on the number of elements at each level.

CCGRVC generates combinational circuits depending on the limit on the number of logical blocks, which is set by the user. The user specifies the number of different logic elements to be present in the circuit. Circuits are generated based on these parameters.

The genetic algorithm [17] is one of the promising methods. It lies in the fact that the program generates a starting population, depending on the type of chromosomes specified by the user. The initial population is then checked against the stopping criterion. If the requirements are not met, in the resulting population, parents are selected, which are then crossed with each other as a result of which mutations occur. Then there is a new population selection among the descendants. The cycle continues until the stop criterion is reached.

CCGUDS gives the possibility to manually add combinational circuits in Verilog format for their subsequent processing and adding to the final dataset. This method makes it possible to manually supplement the dataset with the necessary circuits, as well as check the operability of the software as it is being finalized.

The generation algorithms are described in more detail in [16]. In this study, they were finalized and optimized to speed up their work using the C ++ language. Using C++ allows making the program cross-platform since C# is focused on the Windows operating system. Also, the advantage of C++ is that in tasks that require processing a large amount of data, such as generating large datasets, it can be solved with fewer resources unlike C#. In addition, there are more libraries in C++ to simplify the development.

The algorithms for generating combinational circuits were also improved by adding validation of input data in each algorithm, not just in the user interface. This prevents any errors that might occur when using generation methods separate from the server side. Thus, all necessary restrictions on the user-entered parameters are displayed directly in the web interface. Thus, the user immediately receives information about the correctness of the data and has the opportunity to correct errors before sending a request for the generation.

A support for launching the software from the command line with specified parameters, which allows running it on the server is implemented.

4.2.2 Reliability calculation

The calculation of the fault tolerance of combinational circuits is carried out using the Nadezhda software [18], as well as using the software implementation of the algorithms described in various works [3–7]. They use various methods, ranging from probabilistic ones [3], [6] to methods using machine learning. This makes it possible to compare the presented algorithms and obtain more complete data for the further research.

4.2.3 Unit tests

To simplify the development process, a CMake [19] file was developed the presence of which is a necessary requirement for using the program. The project uses exactly this build system since it is quite simple for the user and allows not thinking about dependencies because they are described in

the build files and are installed automatically. This was an additional advantage over the previous version of the C# program.

Also, in the same build system, a separate build of regression tests is implemented, which greatly simplifies the work of developers.

CI/CD is configured in the project repository, which, each time a new version of the project is loaded, builds an application for generation, and also launches all the tests. This way the developers of the project will be notified if incorrect changes are accidentally sent to the server (which sometimes happens due to human error).

5 Approbation

Using the improved software, a dataset (about 30 thousand circuits) needed for further research in the development of methods for assessing the fault tolerance of microelectronic circuits was generated. The dataset contains a variety of combinational circuits due to the non-correlation of the algorithms used for generation and the diversification of parameters for generation. The non-correlation of generation methods makes it possible to avoid retraining a neural network for a specific type of generation, which can adversely affect the universality of the fault tolerance methods developed.

As is shown in work [16], the use of machine learning methods, neural networks or XGBoost [20] makes it possible to make fault tolerance predictions with high accuracy. But in the work, only the parameters of the combinational circuits were used for training and not the circuits themselves, which leads to the fact that several circuits can be mapped to the same parameters with different fault tolerance values. To improve the quality of fault tolerance predictions or other parameters, consider other approaches that can be involved to use the generated dataset for application with machine learning methods.

The first possible use of a dataset is based on the fact that a combinational circuit is basically a graph. Thus, a graph neural network (GNN) can be used, which is a specialized neural network that takes a graph as input. In [21], modern GNNs are divided into four categories: recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial-temporal GNNs. Using convolutional GNNs, one can, for example, make combinational circuit fault tolerance predictions based on a generated dataset or other known parameters. Another example is the use of graph autoencoders to improve the characteristics of circuits (fault tolerance, delays, etc.), which is also possible using the generated dataset with its additional processing and grouping based on the required parameters. GNNs can also be used for the tasks of generating combinational circuits (GAN [22]).

The second application of the dataset created is the use of natural language processing (NLP) methods [23]. Currently, NLP is used to perform various tasks, such as speech recognition [24], text classification [25], named entity extraction, etc. Since the combinational circuit in the dataset is presented as source code in the Verilog language, such text can be processed by NLP methods to classify the combinational circuits into separate categories, for example, by fault tolerance or size, as well as to generate combinational circuits through their Verilog descriptions.

The third possible way to use the generated dataset is to represent the combinational circuit in the form of graph. In [26], a similar solution showed good results for graph classification problems is described. Based on this, we can conclude that this approach can be extended to the problem of classifying combinational circuits and assessing their fault tolerance.

6 Conclusion

Thus, the result of this study is a new open client-server application that allows generating combinational circuits using various generation methods and aggregating them in a selected location. Based on it, a system that allows generating, storing, and processing datasets of microelectronic circuits on various devices was implemented. The program also provides

functionality for assessing the fault tolerance of microelectronic circuits and for visualizing the generated circuits using other open Yosys software.

Previously developed methods for assessing fault tolerance were analyzed, and some of them were also implemented. A CMake file was developed, and a CI/CD development method was organized to optimize the software. This increased the reliability of making changes to the program. The software is now organized on the principles of client-server systems. A web interface was developed to simplify user interaction with the program. Restriction checks applied to generation parameters and validation of parameters entered by the user were realized.

During the approbation of software, an extensive dataset of combinational circuits was generated, and an analysis of various machine learning methods that can be used was made.

References

[1]. Harris S. L., Harris D. M. Digital Design and Computer Architecture: RISC-V Edition. Morgan Kaufmann, 2021. 592 p.

[2]. Roy S., Tilak C. T. On Synthesis of Combinational Logic Circuits. International Journal of Computer Applications, vol. 127, no. 1, 2015, pp. 21–26. DOI: 10.5120/IJCA2015906311.

[3]. Han J., Chen H., Boykin E., Fortes J. Reliability evaluation of logic circuits using probabilistic gate models. Microelectronics Reliability, vol. 51, no. 2, 2011, pp. 468–476. DOI: 10.1016/j.microrel.2010.07.154.

[4]. Choudhury M. R., Mohanram K. Reliability Analysis of Logic Circuits. IIEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 3, 2009, pp. 392–405. DOI: 10.1109/TCAD.2009.2012530.

[5]. Стемпковский А. Л., Тельпухов Д. В., Соловьев Р. А., Тельпухова Н. В. Исследование вероятностных методов оценки логической уязвимости комбинационных схем. Проблемы разработки перспективных микро- и наноэлектронных схем, №4, 2016, сс. 121-126. / Stempkovskiy A.L., Telpukhov D.V., Soloviev R.A., Telpukhova N.V. Probabilistic methods for reliability evaluation of combinational Circuits. Problems of Perspective Micro- and Nanoelectronic Systems Development, no. 4, 2016, pp. 121-126 (in Russian).

[6]. Тельпухов Д. В., Соловьев Р. А., Тельпухова Н. В., Щелоков А. Н. Оценка параметра логической чувствительности комбинационной схемы к однократным ошибкам с помощью вероятностных методов. Известия ЮФУ. Технические науки, №7, 2016, сс. 149–158. DOI: 10.18522/2311-3103-2016-7-149158. / Telpukhov D.V., Soloviev R. A., Telpukhova N. V., Schelokov A.N. Ocenka parametra logicheskoj chuvstvitel'nosti kombinacionnoj shemy k odnokratnym oshibkam s pomoshh'ju verojatnostnyh metodov. Izvestija JuFU. Tehnicheskie nauki, no. 7, 2016, pp. 149-158. DOI: 10.18522/2311-3103-2016-7-149158 (in Russian).

[7]. Стемпковский А. Л., Тельпухов Д. В., Соловьев Р. А., Мячиков М. В., and Тельпухова Н. В., “Разработка технологически независимых метрик для оценки маскирующих свойств логических схем,” Вычислительные технологии, том 21, №2, 2016, сс. 53–62. / Stempkovskiy A.L., Telpukhov D.V., Soloviev R.A., Myachikov M. V., Telpukhova N. V. The development of technology-independent metrics for evaluation of the masking properties of logic, vol. 21, no. 2, 2016, pp. 53-62.

[8]. Icarus Verilog, Available at: https://github.com/steveicarus/iverilog, accessed 27.07.2023.

[9]. NI Multisim Circuit Design Suite, Available at: https://www.studica.com/NI-Circuit-Design-Suite-Student-Edition-Download, accessed 27.07.2023.

[10]. Logicly, Available at: https://logic.ly/, accessed 27.07.2023.

[11]. CAD_Combinational_Circuits, Available at: https://github.com/RomeoMe5/CAD_Combinational_Circuits, accessed 27.07.2023.

[12]. Json.NET – Newtonsoft, Available at: https://www.newtonsoft.com/json, accessed 27.07.2023.

[13]. Yosys Open Synthesis Suite, Available at: https://yosyshq.net/yosys/, accessed 27.07.2023.

[14]. Synology Drive API, Available at: https://github.com/zbjdonald/synology-drive-api, accessed 27.07.2023.

[15]. Graphviz, Available at: https://graphviz.org/, accessed 28.07.2023.

[16]. Zunin V. V., Romanov A. Y., Solovyev R. A. Developing Methods for Combinational Circuit Generation. In Proceedings - 2022 International Russian Automation Conference (RusAutoCon), 2022, pp. 842–846. DOI: 10.1109/RUSAUTOCON54946.2022.9896390.

[17]. Banzhaf W., Nordin P., Keller R., Francone F. Genetic programming: an introduction on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers, 1998.

[18]. Nadezhda - Reliability Enhancement Logic tool for Integrated Circuits design, Available at: https://alphachip-tools.ru/nadezhda.php, accessed 27.07.2023.
[19]. CMake, Available at: https://cmake.org/, accessed 28.06.2023.
[20]. Chen T., Guestrin C. XGBoost. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.
[21]. Wu Z., Pan S., Chen F., Long G., Zhang C., Yu P. S. A Comprehensive Survey on Graph Neural Networks. IEEE Trans. Neural Networks Learn. Syst., vol. 32, no. 1, 2021, pp. 4–24, DOI: 10.1109/TNNLS.2020.2978386.
[22]. Зунин В.В., Романов А.Ю. Intel OpenVINO™ Toolkit: анализ производительности выполнения генеративно-состязательных нейронных сетей. Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), выпуск 2, 2021, сс. 83–90. DOI: 10.31114/2078-7707-2021-2-83-90. / Zunin V.V., Romanov A. Y. Intel OpenVINO™ Toolkit: Performance Analysis of Generative Adversarial Neural Networks. Problems of Perspective Micro- and Nanoelectronic Systems Development, issue 2, 2021, pp. 83-90. DOI: 10.31114/2078-7707-2021-2-83-90.
[23]. Otter D. W., Medina J. R., Kalita J. K. A Survey of the Usages of Deep Learning for Natural Language Processing. IEEE Trans. NEURAL NETWORKS Learn. Syst., vol. 32, no. 2, 2021, DOI: 10.1109/TNNLS.2020.2979670.
[24]. Kamath U., Liu J., Whitaker J. Deep Learning for NLP and Speech Recognition. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-14596-5.
[25]. Romanov A., Kozlova E., Lomotin K. Application of NLP Algorithms: Automatic Text Classifier Tool. In Communications in Computer and Information Science, vol. 859, 2018, pp. 310–323. DOI: 10.1007/978-3-030-02846-6_25.
[26]. Tixier A. J. P., Nikolentzos G., Meladianos P., Vazirgiannis M. Graph Classification with 2D Convolutional Neural Networks. Lecture Notes in Computer Science, vol. 11731, 2017, pp. 578–593, DOI: 10.1007/978-3-030-30493-5_54.

Information about authors / Информация об авторах

Данил Александрович МЯЧИН – студент национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: машинное обучение, разработка САПР.

Danil Aleksandrovich MIACHIN – student from the HSE University. Research interests: machine learning, CAD development.

Виктория Павловна ПУГАЧ – студент национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: машинное обучение, разработка САПР.

Viktoriia Pavlovna PUGACH – student from the HSE University. Research interests: machine learning, CAD development.

Степан Сергеевич АВДЕЮК– студент национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: машинное обучение, разработка САПР.

Stepan Sergeevich AVDEIUK– student from the HSE University. Research interests: machine learning, CAD development.

Владимир Викторович ЗУНИН – аспирант департамента компьютерной инженерии национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: машинное обучение, системы автоматизированного проектирования.

Vladimir Viktorovich ZUNIN – postgraduate student from the HSE University. Research interests: machine learning, CAD systems.

Aleksandr Yur'yevich ROMANOV – PhD in Technical Sciences, Associate Professor, Head of the Laboratory of Computer-Aided Design Systems at the HSE University. Research interests: networks-on-chip, systems-on-chip, machine learning, CAD systems.

Александр Юрьевич РОМАНОВ – кандидат технических наук, доцент, заведующий лабораторией систем автоматизированного проектирования национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: сети на кристалле, системы на кристалле, машинное обучение, системы автоматизированного проектирования.

DOI: 10.15514/ISPRAS-2023-35(5)-7

О проблемах использования библиотеки OpenBLAS в продуктовом коде на RISC-V

К.А. Зайцева, ORCID: 0009-0009-8645-8795 <*k.zaytseva@yadro.com*>
В.В. Пузикова, ORCID: 0000-0003-0712-4519 <*v.puzikova@yadro.com*>
А.Д. Соколов, ORCID: 0009-0007-3404-5660 <*an.sokolov@yadro.com*>

ООО YADRO,
123022, Россия, г. Москва, ул. Рочдельская, д. 15, стр. 15.

Аннотация. Использование для численного решения задач механики сплошной среды метода граничных элементов приводит к необходимости решения системы линейных алгебраических уравнений с заполненной матрицей. Стандартами де-факто интерфейса программных реализаций функций над заполненными матрицами являются BLAS/LAPACK. Среди оптимизированных открытых реализаций BLAS/LAPACK, только библиотека OpenBLAS включает в себя оптимизации под самый широкий спектр аппаратных платформ – Intel, AMD, ARM и RISC-V. Экосистема открытой архитектуры RISC-V в настоящее время активно развивается: европейские суперкомпьютерные центры открыли центры компетенции RISC-V в рамках правительственной грантовой поддержки EuroHPC, поскольку решения, основанные на архитектуре ARM, не были признаны частью европейской инициативы по развитию собственной технологической независимости. В настоящее время в мире разрабатываются не только высокопроизводительные RISC-V процессоры, но и AI-ускорители, а также видеокарты на RISC-V архитектуре. OpenBLAS активно поддерживается и оптимизируется под появляющееся RISC-V оборудование и расширения. Однако, к библиотекам, использующимся в продуктовом коде, традиционно предъявляются серьезные требования по стабильности и надежности, чтобы минимизировать возможные ошибки и сбои в продукте. Как оказалось, с этой точки зрения, OpenBLAS имеет ряд проблем, которые нам пришлось решить с целью продуктизации этой библиотеки. В данной статье описывается тестовая система OpenBLAS, рассматриваются проблемы тестирования LAPACK-функционала библиотеки и пути их решения. Кроме того, анализируется тестовое покрытие BLAS-функционала и обсуждаются достигнутые результаты по его увеличению. В дальнейшем планируется внести описанные изменения в проект OpenBLAS.

Ключевые слова: метод граничных элементов; система линейных алгебраических уравнений с заполненной матрицей; OpenBLAS; LAPACK; RISC-V; тестирование; продуктизация.

Для цитирования: Зайцева К.А., Пузикова В.В., Соколов А.Д. О проблемах использования библиотеки OpenBLAS в продуктовом коде на RISC-V. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 91–106. DOI: 10.15514/ISPRAS–2023–35(5)–7.

On Problems in OpenBLAS Library Usage in Productized Code on RISC-V

K.A. Zaytseva, ORCID: 0009-0009-8645-8795 <*k.zaytseva@ yadro.com*>
V.V. Puzikova, ORCID: 0000-0003-0712-4519 <*v.puzikova@yadro.com*>
A.D. Sokolov, ORCID: 0009-0007-3404-5660 <*an.sokolov@ yadro.com*>
YADRO,
15, bld.15, Rochdelskaya st., Moscow, 123022, Russia.

Abstract. The boundary element method usage for the numerical simulation in continuum mechanics problems leads to the need to solve a system of linear algebraic equations with a dense matrix. The de facto standards for the interface of functions over dense matrices and vectors software implementations are BLAS/LAPACK. Among the optimized open-source BLAS/LAPACK implementations, only the OpenBLAS library includes optimizations for the widest range of hardware platforms. This library is optimized for Intel, AMD, ARM and RISC-V architectures. The open RISC-V architecture ecosystem is currently actively developing. European supercomputing centers have opened RISC-V competence centers as part of the government's EuroHPC grant support, since solutions based on the ARM architecture are not recognized as part of the European initiative to develop its own technological independence. Currently, companies included in the international RISC-V consortium are developing not only high-performance RISC-V processors, but also AI accelerators, as well as video cards based on RISC-V architecture. OpenBLAS is actively supported and optimized for emerging RISC-V hardware and extensions. However, libraries used in product code are traditionally subject to strict requirements for stability and reliability in order to minimize possible errors and failures in the product. As it turned out, from this point of view, OpenBLAS has a number of problems that we had to solve in order to productize this library. In this article the OpenBLAS test system is described, the problems of testing the LAPACK functionality of the library and ways to solve them are discussed. In addition, the test coverage of the BLAS functionality is analyzed and the results achieved in increasing it are presented. It is planned to contribute the described changes to the OpenBLAS project.

Keywords: boundary element method; system of linear equations with dense matrix; OpenBLAS; LAPACK; RISC-V; testing; productization.

For citation: Zaytseva K.A., Puzikova V.V., Sokolov A.D. On Problems in OpenBLAS Library Usage in Productized Code on RISC-V. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 91-106 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-7.

1. Введение

Б*о*льшая часть задач механики сплошной среды не имеет аналитического решения и может быть решена только численно. Численные методы решения краевых задач механики сплошной среды глобально можно разделить на две группы.

Первую группу составляют методы, в которых происходит построение разностных аналогов исходных дифференциальных уравнений в частных производных во всей расчетной области. Методы этой группы можно разделить [1-2] на методы конечных разностей (МКР), методы контрольных объемов (МКО) и методы конечных элементов (МКЭ). МКР подразумевают замену производных из исходных уравнений на разностные на построенной в расчетной области сетке. В МКО ячейки построенной сетки являются контрольными объемами, в которых интегрируются уравнения, описывающие физические процессы в задаче. Затем, при помощи принципов сохранения массы, импульса и энергии, полученные уравнения связываются между соседними контрольными объемами через граничные условия. Наконец, в МКЭ, каждая ячейка сетки является конечным элементом, на котором выбирается вид аппроксимирующей функции, вне элемента равной нулю. Значения этих функций на границах элементов в узлах сетки являются решением задачи. Их коэффициенты определяются из условия равенства значений функций на границах между элементами [3].

Ко второй группе относят метод граничных элементов (МГЭ) или метод граничных интегральных уравнений [4-6]. Этот метод принципиально отличается от методов первой

группы тем, что сначала краевая задача механики сплошной среды, состоящая из дифференциальных уравнений в частных производных и краевых условий, при помощи формул Грина сводится к интегральному уравнению на границе расчетной области, а затем уже для этого граничного интегрального уравнения строится дискретный аналог. МГЭ был разработан в 1960-х годах и с тех пор претерпел множество изменений и усовершенствований. Сначала МГЭ использовался преимущественно для решения задач электростатики и потенциального течения. Он позволял точно моделировать эффекты на поверхности объекта, такие как распределение электрического заряда или давления. В дальнейшем МГЭ был расширен для решения уравнений упругости и теплопроводности, что позволило моделировать напряжения и деформации в материалах, а также тепловые потоки. В настоящее время в механике сплошной среды МГЭ находит широкое применение во многих областях, включая [4-5]:

- структурную механику – МГЭ используется для анализа напряжений и деформаций в конструкциях, таких как мосты, здания, автомобили и самолеты, что позволяет оценить прочность и надежность конструкций, а также оптимизировать их дизайн;
- геотехнику – МГЭ применяется для моделирования поведения грунтов и горных пород при различных нагрузках и условиях, анализа устойчивости склонов, расчета осадок при строительстве фундаментов и тоннелей, а также для моделирования поведения грунта при землетрясении;
- аэродинамику и гидродинамику – МГЭ применяется для моделирования потока воздуха или жидкости вокруг объектов, таких как крылья самолетов или корпуса судов, позволяет оценить силы сопротивления и подъемные силы, а также оптимизировать форму объекта для улучшения его аэродинамических или гидродинамических характеристик;
- акустику – МГЭ используется для моделирования звуковых полей и распространения звука в сплошных средах, анализа шумовых и вибрационных характеристик зданий, автомобилей или других объектов, а также для оптимизации звукового дизайна;
- электромагнетизм – МГЭ применяется для моделирования распределения электрического или магнитного поля вокруг объектов, анализа электромагнитной совместимости, проектирования антенн или оптимизации электромагнитных устройств.

Общей отличительной чертой всех численных методов первой группы (МКР, МКО, МКЭ) является разреженная структура матриц систем линейных алгебраических уравнений (СЛАУ), получающихся после дискретизации дифференциальных уравнений по времени и пространству. На всякий случай поясним, что разреженными называются матрицы, б*о*льшая часть элементов которых является нулями. В программной реализации для хранения таких матриц используются специальные форматы, такие как CSR, CSC, COO, ELL, JAD, BSR, BSC и так далее. [7], чтобы не расходовать память на хранение нулевых элементов. Для решения СЛАУ с разреженными матрицами, как правило, используются специальные итерационные методы, например, крыловские методы (CG, BiCGStab, FGMRES и др.), зачастую со специальными предобуславливателями, учитывающими разреженную структуру матрицы [8].

В результате использования метода граничных элементов [4-6], напротив, получается СЛАУ с заполненной матрицей, для решения которой используют прямые методы, а не перечисленные выше итерационные. Стандартом де-факто интерфейса программных реализаций функций линейной алгебры над заполненными матрицами и векторами является BLAS (Basic Linear Algebra Subprograms – базовые подпрограммы линейной алгебры). Этот стандарт был опубликован в 1979 г. [9] и использован, в частности, при создании LAPACK (Linear Algebra PACKage) – библиотеки для решения широкого спектра задач линейной

алгебры, таких как решение систем линейных уравнений, вычисление собственных значений и векторов, построение матричных разложений и мн. др. [10]. Таким образом, пакеты математических программ, позволяющие проводить моделирование методом граничных элементов, должны содержать либо собственные реализации отдельных необходимых функций BLAS/LAPACK, либо использовать сторонние BLAS/LAPACK библиотеки.

Так, например, один из наиболее универсальных инструментов для построения дискретных аналогов для граничных интегральных операторов и решения задач методом МГЭ, open-source проект BEM++ [11,12] может собираться с реализациями BLAS/LAPACK из библиотек Intel MKL, GotoBLAS и OpenBLAS [13]. В настоящее время использование первой из перечисленных библиотек может быть сопряжено с юридическими рисками, а проект GotoBLAS с 2008 года не обновляется. Поэтому наиболее предпочтительным является использование для BEM++ библиотеки OpenBLAS.

Необходимо отметить, что среди других оптимизированных высокопроизводительных открытых реализаций BLAS/LAPACK, таких как, например, Eigen [14] и Armadillo [15], библиотека OpenBLAS выделяется оптимизациями под самый широкий спектр аппаратных платформ – Intel, AMD, ARM и RISC-V, причем под RISC-V [16] к настоящему моменту оптимизирована только она одна. В то же время экосистема открытой архитектуры RISC-V сейчас начинает активно развиваться в направлении HPC (High Performance Computing): ведущие европейские HPC центры, такие как BSC (Barcelona Supercomputing Center) и EPCC (Edinburgh Parallel Computing Centre), открыли центры компетенции RISC-V в рамках правительственной грантовой поддержки EuroHPC [17]. Это вызвано тем, что решения, основанные на архитектуре ARM, не были признаны частью европейской инициативы по развитию собственной технологической независимости. Первые RISC-V HPC системы ожидаются в 2025-2026 гг. в рамках проекта европейского суперкомпьютера BSC MareNostrum 6. Идет разработка не только высокопроизводительных RISC-V CPU, таких как 432-ядерный Occamy [18] от ETH или Atrevido 423 от Semidynamics, но и AI-ускорителей в рамках проекта EUPILOT [19]. Кроме того, существует проект разработки RISC-V GPU Vortex: сейчас с ним можно работать на FPGA, причем имеется конвейер для запуска программ, написанных на CUDA.

Целью данной работы является исследование стабильности и надежности работы библиотеки OpenBLAS на архитектуре RISC-V, а также минимизация возможных ошибок и сбоев в функционале OpenBLAS, которые ухудшают качество продуктов, использующих OpenBLAS на RISC-V. Это актуально, поскольку в случае использования OpenBLAS на RISC-V имеется сразу два потенциальных источника проблем. Во-первых, OpenBLAS как и многие другие open-source проекты может изначально иметь проблемы с качеством тестирования. Во-вторых, библиотека OpenBLAS оперативно оптимизируется большим сообществом под появляющееся RISC-V оборудование и расширения RISC-V ISA (Instruction Set Architecture), а портирование библиотек на новые архитектуры зачастую может приводить к некорректной работе функционала. Если начать портировать на RISC-V другие открытые HPC библиотеки, использующие OpenBLAS, такие как упомянутый выше BEM++, не убедившись в корректности и стабильности работы OpenBLAS на RISC-V, процесс отладки сильно осложнится.

2. Библиотека OpenBLAS

Библиотека OpenBLAS (Open Basic Linear Algebra Subprograms) является открытым проектом, разработанным для эффективного выполнения операций линейной алгебры на многоядерных процессорах. Она предоставляет набор оптимизированных операций над заполненными матрицами и векторами, таких как умножение матриц, решение систем линейных уравнений и вычисление собственных значений. В библиотеке реализованы BLAS и LAPACK API.

2.1 История развития

История создания и развития OpenBLAS началась в 2002 году [13] с проекта GotoBLAS, разработанного Кадзусигэ Гото (Kazushige Goto) и Робертом ван де Гейном (Robert van de Geijn) в Университете Техаса в Остине. GotoBLAS представляла собой оптимизированную библиотеку базовых линейных алгебраических подпрограмм, в которой использовались ассемблерные инструкции для достижения высокой производительности.

В 2008 году проект GotoBLAS был переименован в ATLAS (Automatically Tuned Linear Algebra Software) и стал открытым проектом с открытым исходным кодом. Библиотека ATLAS предоставляла возможность автоматической настройки для конкретного аппаратного обеспечения, что позволяло достичь еще большей производительности.

В 2011 году разработчики ATLAS объединили свои усилия с коллективом проекта LAPACK (Linear Algebra PACKage) и создали новый проект под названием OpenBLAS. OpenBLAS была разработана с использованием оптимизаций из ATLAS и предоставляла полную совместимость с LAPACK. С тех пор OpenBLAS продолжает развиваться и улучшаться благодаря активному вкладу большого сообщества разработчиков.

2.2 Функционал OpenBLAS

OpenBLAS полностью реализует интерфейсы BLAS и LAPACK, а также предоставляет дополнительное множество «BLAS-like» функций для расширения функциональности BLAS. BLAS-часть библиотеки содержит в себе набор базовых операций линейной алгебры, чаще всего использующихся в прикладных программах. Все функции оптимизированы, как алгоритмически, так и на микроархитектурном уровне, и на их основе строятся реализации более сложных алгоритмов, входящих в LAPACK, PBLAS (Parallel Basic Linear Algebra Subprograms), ScaLAPACK(Scalable Linear Algebra PACKage).

Функции BLAS делятся на следующие 3 группы, называемые «уровнями»:

- 1 уровень – операции над векторами (например, скалярное произведение или умножение вектора на скаляр);
- 2 уровень – векторно-матричные операции (например, умножение матрицы на вектор, причем вид матрицы учитывается при оптимизации алгоритма);
- 3 уровень – матрично-матричные операции (например, умножение матрицы на матрицу или решение СЛАУ с треугольной матрицей).

Интерфейсы BLAS для языков Fortran (BLAS) и C (CBLAS) отличаются только списком аргументов и способом их обработки. В основе каждой функции лежит один и тот же код, написанный на языке C с использованием низкоуровневых оптимизаций и поддержкой многопоточности. Например, для умножения вектора на скаляр и сложения результата с вектором в случае Fortran вызывается функция `saxpy()`, а в случае C/C++ – `cblas_saxpy()`.

LAPACK-часть библиотеки можно разделить на четыре большие группы функций, каждая из которых включает в себя также и выполнение ряда сопутствующих вычислительных задач:

1) построение разложений плотных матриц, таких как разложения LU, QR, Холецкого, Шура, SVD и др.;
2) решение систем линейных алгебраических уравнений с плотными матрицами различных видов (симметричными, несимметричными, эрмитовыми и т.д.);
3) решение линейных задач наименьших квадратов;
4) решение задач на собственные значения.

Каждая группа содержит не только вычислительные функции, но и функции-драйверы, обеспечивающие решение общих задач. Вычислительные функции вызываются из функций-драйверов и выполняют различные подзадачи, например, LU-разложение для функции-драйвера решения СЛАУ.

Также как и BLAS часть LAPACK имеет интерфейс для использования в программах, написанных на языке C/C++. Этот интерфейс называется LAPACKE. Он представляет собой функции-обертки для LAPACK функций. Малая часть LAPACK функциональности реализована непосредственно в OpenBLAS на языке C и содержит дополнительные оптимизации для исполнения в многопоточной среде. Для предоставления полного множества функций LAPACK и LAPACKE в библиотеке OpenBLAS используется реализация из групп библиотек Netlib, написанная на языке Fortran. Кроме того, на случай отсутствия Fortran компилятора в OpenBLAS предусмотрен вариант сборки с использованием исходных файлов реализации LAPACK из Netlib, сконвертированных на язык C с помощью f2clapack скрипта из репозитория PRIMME [20].

2.3 Система сборки OpenBLAS

Библиотека OpenBLAS имеет развитую систему сборки с множеством ключей, позволяющих осуществлять:

- выбор целевой архитектуры сборки с микроархитектурными оптимизациями;
- включение динамической диспетчеризации оптимизаций, когда собранная библиотека содержит оптимизации для нескольких микроархитектур и может определять оптимальную оптимизацию в ходе исполнения программы;
- выбор между использованием пары компиляторов C/Fortran и использованием только одного C компилятора;
- включение/отключение многопоточного режима с использованием технологий параллельного программирования PThreads (POSIX Threads) и OpenMP (Open Multi-Processing);
- выбор ширины знакового целочисленного типа в интерфейсах функций (32/64-битные).

Рассмотрим подробнее выбор целевой архитектуры. Как уже отмечалось выше, OpenBLAS поддерживает многие архитектуры: X86/X86_64, PowerPC, MIPS, ARM, Elbrus, RISC-V и др. Для каждой из перечисленных архитектур в библиотеке OpenBLAS имеются оптимизации для конкретной микроархитектуры, учитывающие особенности набора команд и размеров кэшей. Для рассматриваемой в данной статье архитектуры RISC-V в OpenBLAS в настоящее время существует три варианта сборки:

1) Generic – включает только скалярные инструкции без векторных оптимизаций;
2) RVV 0.7.1 – включает низкоуровневые оптимизации с использованием векторных инструкций из RISC-V Vector Extension v.0.7.1;
3) RVV v.1.0 – включает низкоуровневые оптимизации с использованием векторных инструкций из RISC-V Vector Extension v.1.0.

Здесь необходимо пояснить, что последней ратифицированной версий векторного расширения RISC-V является RVV 1.0, однако в настоящее время RISC-V процессоры с поддержкой данного расширения все еще недоступны в свободной продаже. Промежуточная версия векторного расширения, RVV 0.7.1, поддерживается в ядрах C906 и C910 компании XuanTie. Эти ядра входят в состав процессоров на доступных для покупки платах Sipeed Nezha, Mango Pi, Lichee Pi 4A и др. По этим причинам далее мы будем рассматривать только Generic и RVV 0.7.1 сборки. Отметим, что многопоточные сборки библиотеки не будут рассматриваться в данной статье.

2.4 Тестовая система OpenBLAS

Существует пять открытых наборов тестов для тестирования частей функционала OpenBLAS:

1) тесты для BLAS API, заимствованные из Netlib BLAS с минимальными изменениями (написаны на языке Fortran);
2) тесты для CBLAS API, аналогичные BLAS API тестам, но адаптированные для тестирования CBLAS (изначально написаны на Fortran, но есть возможность сборки с вариантом тестов, сконвертированных на C);
3) собственный OpenBLAS фреймворк Utest для тестирования, схожий с фреймворком GTests (Google Test) и содержащий специфические сценарии тестирования, например, воспроизводители ошибок или вырожденных случаев;
4) отдельный проект BLAS-Tester [21] для тестирования BLAS алгоритмов и сравнения производительности со сторонними реализациями BLAS;
5) тесты для LAPACK API, заимствованные из Netlib BLAS с минимальными изменениями (написаны на языке Fortran).

Отметим, что проект BLAS-Tester является более гибким в настройке, чем исходные тесты из Netlib. Так, например, в тестах из Netlib размеры матриц/векторов указываются по отдельности, в то время как BLAS-Tester позволяет задать сразу диапазон значений с определённым шагом. Кроме того, в BLAS-Tester имеется возможность указывать дополнительные параметры для тестов, например, смещения адресов массивов для тестирования с невыровненной памятью.

Также требует пояснений пятая группа тестов – тестов LAPACK, поскольку это отдельная большая система для тестирования более чем тысячи функций. Тесты этой группы делятся на 18 подгрупп для функций над аргументами одинарной точности и на 19 подгрупп для функций над аргументами двойной точности. Разделение на подгруппы производится не только по типам решаемых задач, но и по способам хранения матриц. Каждая подгруппа тестов запускается с помощью собственного исполняемого файла, которому однозначно соответствует входной файл с параметрами тестирования. Этот файл содержит не только размеры и типы тестовых матриц, но и значения параметров исполнения тестируемых функций, передаваемые в качестве переменных окружения выполнения, а не аргументов. Результатом выполнения тестовой программы служит текстовый файл с логом запуска тестов и подробным описанием возникших ошибок в случае их наличия.

Подчеркнем, что тестирование функций на точность не использует каких бы то ни было заданных эталонных результатов, а опирается исключительно на математические свойства тестируемых алгоритмов (нормы невязки и т.д.). Например, для функции, реализующей LU-разложение матрицы A, значение ошибки δ вычисляется по формуле

$$\delta = \frac{|\mathrm{L} * \mathrm{U} - \mathrm{A}|_1}{\mathrm{N} * |\mathrm{A}|_1 * \varepsilon}, \quad (1)$$

где ε – относительная машинная точность, $|\cdot|_1$ – первая матричная норма (максимальная сумма элементов столбца), N – размер квадратной матрицы. Соответственно, тест будет считаться пройденным, если значение δ меньше заданного при запуске тестов порога.

3. Тестирование библиотеки OpenBLAS на RISC-V

Для сборки библиотеки использовался GCC (GNU Compiler Collection) тулчейн с открытым исходным кодом от компании T-HEAD [22]. Он включает в себя компиляторы для языка C (gcc с поддержкой набора инструкций RVV 0.7.1) и Fortran (gfortran), а также симулятор Qemu с поддержкой RVV 0.7.1. Запуски и отладка осуществлялись не только на Qemu, но и на плате Mango Pi с SoC (System On Chip) AllWinner D1, включающим одно ядро C906 компании XuanTie с поддержкой RVV 0.7.1.

3.1 Анализ тестового покрытия BLAS-функционала и его увеличение

Тесты – важная часть качественного и надежного программного продукта. При использовании открытых библиотек в своих приложениях важно оценивать покрытие кода тестами и при необходимости увеличивать его.

Для того чтобы проанализировать покрытие BLAS-функционала библиотеки OpenBLAS тестами использовалась утилита LCOV [23]. Чтобы собрать отчет о тестовом покрытии, необходимо построить библиотеку с включенной инструментацией. Для этого необходимо добавить ключи сборки `-fprofile-arcs` и `-ftest-coverage`. В результате после построения библиотеки также должны сгенерироваться файлы с расширением gcno.

Теперь нужно создать «базовый уровень» LCOV – файл данных покрытия, содержащих нулевое покрытие для каждой инструментированной строки проекта. Это требуется для того, чтобы получить данные о покрытии всех файлов исходного кода, участвующих в сборке, даже тех, которые не будут использоваться непосредственно во время запуска тестов. Если OpenBLAS компилировался при помощи gcc, то «базовый уровень» LCOV создается при помощи следующей команды:

```
lcov --gcov-tool  <PATH_TO_TOOLCHAIN>/bin/riscv64-unknown-linux-
gnu-gcov --capture --initial --directory driver/ --directory
kernel/--directory interface/ --output-file coverage_base.info
```

Если же для компиляции библиотеки OpenBLAS использовался компилятор clang, возникает проблема с некорректной обработкой пробела между `llvm-cov` и `gcov`. Для решения этой проблемы удобно использовать небольшой скрипт, llvm-gcov.sh, см. листинг 1.

```
#!/bin/bash
exec <path-to-clang-toolchain>/llvm/bin/llvm-cov gcov "$@"
```

Листинг 1. Скрипт llvm-gcov.sh.
Listing 1. Script llvm-gcov.sh.

Данный скрипт необходимо сделать исполняемым и использовать как `gcov-tool` при создании «базового уровня» LCOV:

```
lcov --gcov-tool llvm-gcov.sh --capture –initial --directory driver/
--directory kernel/ --directory interface/ --output-file
coverage_base.info
```

После этого можно запускать BLAS-тесты. Отметим, что помимо тестов из OpenBLAS в исследовании также использовались тесты из проекта BLAS-Tester. По мере прохождения тестов генерируются файлы с расширением gcda. После прохождения всех тестов необходимо снова запустить LCOV, в случае использования gcc при помощи команды

```
lcov --gcov-tool  <PATH_TO_TOOLCHAIN>/bin/riscv64-unknown-linux-
gnu-gcov --capture --directory driver/ --directory kernel/ --
directory interface/ --output-file coverage_tests.info
```

а в случае clang

```
lcov --gcov-tool llvm-gcov.sh --capture --directory driver/ --
directory kernel/ --directory interface/ --output-file
coverage_tests.info
```

Теперь можно объединить файлы трассировки:

```
lcov --add-tracefile coverage_base.info --add-tracefile
     coverage_tests.info --output-file coverage_total.info
```

Перед генерацией отчета о тестовом покрытии необходимо удалить ненужные исходные файлы из файлов трассировки:

```
lcov --remove coverage_total.info -o coverage_total_filtered.info
<path_to_openblas>/interface/lapack/*
<path_to_riscv>/sysroot/usr/include/* <path_to_openblas>/common*
<path_to_openblas>/symcopy.h
```

После выполнения всех этих действий можно сгенерировать отчет о покрытии кода библиотеки тестами, выполнив следующую команду:

```
genhtml --output-directory html coverage_total_filtered.info
```

В итоге в html директории должен появиться файл index.html с отчетом. Отчеты LCOV для Generic (рис. 1) и RVV 0.7.1 (рис. 2) сборок OpenBLAS показали, что покрытие кода тестами не превышает 79%.

Для функций с покрытием менее 60% были написаны недостающие тесты и включены в наш форк OpenBLAS, после чего отчеты были собраны снова. Результаты представлены на рис. 3 и 4.

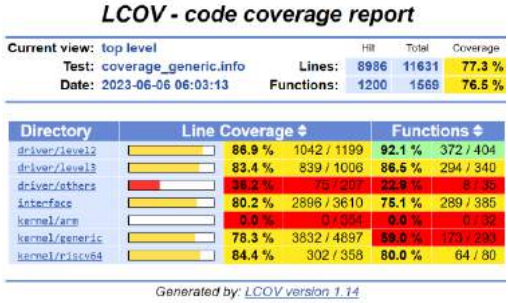


Рис. 1. Отчет LCOV о тестовом покрытии Generic сборки OpenBLAS.
Fig. 1. LCOV report on test coverage of OpenBLAS Generic build.

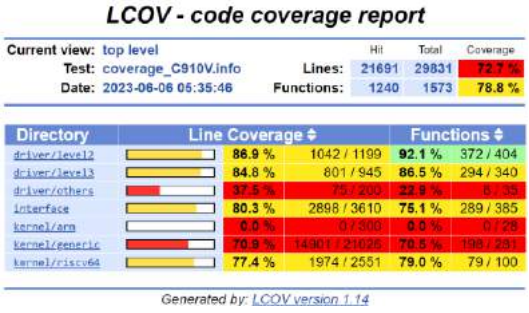


Рис. 2. Отчет LCOV о тестовом покрытии RVV 0.7.1 сборки OpenBLAS.
Fig. 2. LCOV report on test coverage of OpenBLAS RVV 0.7.1 build.

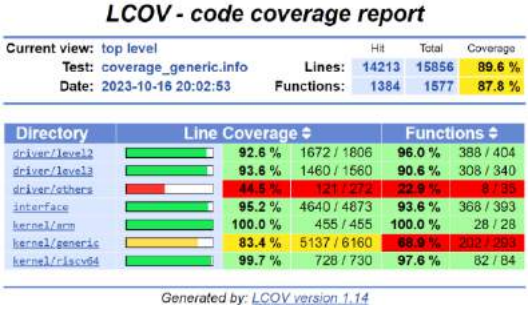


Рис. 3. Отчет LCOV о тестовом покрытии Generic сборки OpenBLAS после проведения работ по его увеличению и исправлению обнаруженных ошибок.
Fig. 3. LCOV report on test coverage of OpenBLAS Generic build after work to increase it and fix detected bugs.

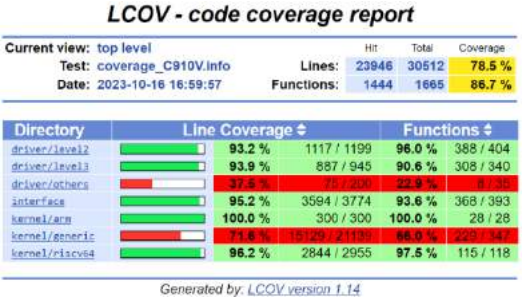


Рис. 4. Отчет LCOV о тестовом покрытии RVV 0.7.1 сборки OpenBLAS после проведения работ по его увеличению и исправлению обнаруженных ошибок.
Fig. 4. LCOV report on test coverage of OpenBLAS RVV 0.7.1 build after work to increase it and fix detected bugs.

В результате суммарное покрытие BLAS функций второго и третьего уровня превысило 92%, а покрытие функций с RISC-V оптимизациями – 96%. При этом добавленные тесты позволили выявить и устранить ряд ошибок в BLAS-функционале, например:

- `imatcopy()` – ошибка выделения памяти и некорректная обработка краевых случаев для комплексных матриц, затрагивающая все архитектуры;
- `gemmt()` – некорректная обработка комплексно-сопряженных матриц, затрагивающая все архитектуры;
- `dsdot()` – отсутствие реализации для RVV 0.7.1 (использовалась реализация `sdot()`, не обеспечивавшая необходимой точности);
- `asum()` и `axpby()` – некорректная работа оптимизаций при некоторых значениях инкремента;
- `i[s/d]max/min()` – ошибка в приведении типов.3.2 Решение проблем с прохождением тестов LAPACK-функционала

Перейдем к тестированию LAPACK-функционала библиотеки OpenBLAS. Для этого запустим описанный выше набор LAPACK тестов для Generic и RVV 0.7.1 сборок библиотеки с помощью симулятора Qemu, а также на плате Mango Pi. Для каждого набора оптимизаций будет протестировано четыре конфигурации библиотеки:

1) «Fortran, Int32» – сборка с использованием исходных файлов LAPACK на Fortran и 32-битными целочисленными типами;
2) «Fortran, Int64» – сборка с использованием исходных файлов LAPACK на Fortran и 64-битными целочисленными типами;
3) «C, Int32» – сборка с использованием файлов LAPACK, сконвертированных на C, и 32-битными целочисленными типами;
4) «C, Int64» – сборка с использованием файлов LAPACK, сконвертированных на C, и 64-битными целочисленными типами;

Общее количество LAPACK-тестов для каждого типа данных представлено в табл. 1. Однако, первая попытка запуска тестов для Generic сборки OpenBLAS показала, что по непонятным причинам запускаются не все тесты (табл. 2). Кроме того, в конфигурации «C, Int64» падает в сумме по всем типам данных около 9500 тестов (табл. 3).

Табл. 1 Количество запускаемых LAPACK тестов.
Table 1. Number of launched LAPACK tests.

Тип данных	Float	Double	Complex Float	Complex Double
Число тестов	1 327 023	1 327 845	786 775	787 842

Табл. 2 Количество пропущенных LAPACK тестов для Generic сборки
Table 2. Number of skipped LAPACK tests for Generic build

Конфигурация	Float	Double	Complex Float	Complex Double
Fortran, Int32	9 336 (0.70%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Fortran, Int64				
C, Int32	10 260 (0.77%)		385 (0.00%)	
C, Int64	529 067 (39.86%)	518 817 (39.07%)	23 907 (3.03%)	23 778 (3.01%)

Для RVV 0.7.1 сборки результаты запуска тестов еще хуже (табл. 4 и 5): около 75% тестов не запускается, все тесты при использовании сконвертированных файлов для double зависают. Анализ результатов тестирования RVV 0.7.1 сборки показал, что большинство тестов пропущены по причине ошибок сегментации, возникающих из-за выхода за пределы массива. Кроме того, были зафиксированы ошибки в вычислениях и зависания тестов.

Табл. 3 Статистика падений LAPACK тестов для Generic сборки
Table 3. Fail statistics of LAPACK tests for Generic build

Конфигурация	Float	Double	Complex Float	Complex Double
Fortran, Int32	3 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Fortran, Int64				
C, Int32	1 (0.00%)		1 (0.00%)	
C, Int64	2 206 (0.27%)	2 220 (0.27%)	2 469 (0.32%)	2 469 (0.32%)

Табл. 4 Количество пропущенных LAPACK тестов для RVV 0.7.1 сборки
Table 4. Number of skipped LAPACK tests for RVV 0.7.1 build

Конфигурация	Float	Double	Complex Float	Complex Double
Fortran, Int32	584 250 (44.02%)	1 297 545 (97.71%)	507 395 (64.49%)	527 820 (66.99%)
Fortran, Int64				
C, Int32	583 158 (43.94%)	Нет данных (зависания)	513 746 (65.29%)	534 042 (67.78%)
C, Int64	1 101 965 (83.04%)		531 302 (67.52%)	551 598 (70.01%)

Табл. 5 Статистика падений LAPACK тестов для RVV 0.7.1 сборки
Table 5. Fail statistics of LAPACK tests for RVV 0.7.1 build

Конфигурация	Float	Double	Complex Float	Complex Double
Fortran, Int32	404 (0.05%)	4 033 (13.31%)	2 076 (0.74%)	0 (0.00%)
Fortran, Int64				
C, Int32	382 (0.05%)	Нет данных (зависания)	2 077 (0.74%)	
C, Int64	2 620 (1.16%)		4 549 (1.78%)	2 471 (1.00%)

Поиск источников перечисленных проблем было решено начинать с Generic сборки, поскольку возникло предположений, что Generic реализация BLAS-функций, вызываемых из LAPACK алгоритмов, корректна и ошибки связаны с использованием сконвертированных на язык C файлов. Отладка производилась с помощью пары Qemu и GDB, которая позволяет отлаживать программы без использования конкретного RISC-V процессора. В результате было установлено, что сконвертированные на язык C файлы LAPACK из репозитория OpenBLAS содержат две проблемы, связанные с их взаимодействием с тестовой системой.

Первая проблема заключается в том, что размер типа logical, заменяющего на C тип LOGICAL из Fortran, в C файлах равен 32 битам. Однако при сборке Fortran кода используется ключ компиляции `-fdefault-integer-8`, который изменяет ширину не только INTEGER типа, но и LOGICAL. Отсутствие ошибок компиляции обусловлено особенностью передачи явных аргументов в Fortran-функции исключительно по указателю. После исправления этой проблемы количество запущенных тестов стало соответствовать другим конфигурациям, однако остались падения тестов с неверными статусами.

Эти падения были связаны со второй проблемой. Она заключалась в ошибке конвертации файлов, в которых функция обработки ошибок `xerbla()`, реализованная в тестовой системе на Fortran, вызывается с неверным количеством аргументов. Дело в том, что для каждой функции на Fortran неявно указываются последним аргументом размеры строк, переданные в эту функцию (листинг 2). Исправления этой проблемы представлены в листинге 3.

После исправления этих двух проблем число падений тестов для Generic сборки стало незначительным (табл. 6 и 7). Эти падения связаны исключительно с проблемами в точности алгоритмов для некоторых тестовых матриц. С пропусками менее 0.8% тестов float функций для всех конфигураций еще предстоит разобраться.

```
// Реализация на Fortran
SUBROUTINE XERBLA(SRNAME, INFO)
// Вызов функции на C
xerbla_("CGBCON", &i__1);
```

Листинг 2. Вызов xerbla_ из Си кода.
Listing 2. xerbla_ call from C code.

```
// Вызов функции на C
xerbla_("CGBCON", &i__1, (ftnlen)6);
```

Листинг 3. Исправленный вызов xerbla_ из Си кода.
Listing 3. Corrected xerbla_ call from C code.

Табл. 6 Количество пропущенных LAPACK тестов для Generic сборки после исправлений
Table 6. Number of skipped LAPACK tests for Generic build after fixes

Конфигурация	Float	Double	Complex Float	Complex Double
Fortran, Int32	9 336 (0.70%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Fortran, Int64				
C, Int32	10 260 (0.77%)		385 (0.00%)	
C, Int64				

Табл. 7 Статистика падений LAPACK тестов для Generic сборки после исправлений
Table 7. Fail statistics of LAPACK tests for Generic build after fixes

Конфигурация	Float	Double	Complex Float	Complex Double
Fortran, Int32	3 (0.00%)	0 (0.00%)	0 (0.00%)	0 (0.00%)
Fortran, Int64				
C, Int32	1 (0.00%)		1 (0.00%)	
C, Int64				

Следующим шагом было исправление падений и зависаний LAPACK тестов для RVV 0.7.1 сборки библиотеки. Оказалось, что они связаны с проблемами в реализации вызываемых BLAS-функций с векторными оптимизациями под RISC-V. В ходе отладки были обнаружены и исправлены следующие проблемы в функциях:

- `idamax()` – ошибка в использовании RVV интринсиков, приводящая к индексации по значению переменной беззнакового типа, которой могло быть присвоено

отрицательное значение (в результате чего возникал SegFault);

- `dgemm()` – ошибка в использовании на ассемблере неверной инструкции для обнуления float регистра (в результате старшие 32 бита остались ненулевыми и могли появляться NaN значения);
- `nrm2()` – ошибка при обработке последней части массива, так называемого «хвоста», длина которого меньше длины векторного регистра (в результате чего функция возвращала неверный результат).

Таким образом, несмотря на имеющуюся развитую систему тестирования BLAS, некоторые проблемы в BLAS функционале оказалось возможным определить только благодаря LAPACK тестам.

Финальная статистика прохождения LAPACK-тестов для RVV 0.7.1 сборки представлена в табл. 8 и 9. В результате проведенной работы количество запущенных тестов стало почти соответствовать ожидаемому, однако с некоторыми тестами для комплексных чисел float точности еще предстоит разобраться.

Табл. 8 Количество пропущенных LAPACK тестов для RVV 0.7.1 сборки после исправлений
Table 8. Number of skipped LAPACK tests for RVV 0.7.1 build after fixes

<table>
<tr><th>Конфигурация</th><th>Float</th><th>Double</th><th>Complex Float</th><th>Complex Double</th></tr>
<tr><td>Fortran, Int32</td><td rowspan="2">1092 (0.00%)</td><td rowspan="4">0 (0.00%)</td><td rowspan="2">0 (0.00%)</td><td rowspan="4">0 (0.00%)</td></tr>
<tr><td>Fortran, Int64</td></tr>
<tr><td>C, Int32</td><td rowspan="2">0 (0.00%)</td><td rowspan="2">895 (0.11%)</td></tr>
<tr><td>C, Int64</td></tr>
</table>

Табл. 9 Статистика падений LAPACK тестов для RVV 0.7.1 сборки после исправлений
Table 9. Fail statistics of LAPACK tests for RVV 0.7.1 build after fixes

<table>
<tr><th>Конфигурация</th><th>Float</th><th>Double</th><th>Complex Float</th><th>Complex Double</th></tr>
<tr><td>Fortran, Int32</td><td rowspan="2">5 (0.00%)</td><td rowspan="4">0 (0.00%)</td><td rowspan="2">0 (0.00%)</td><td rowspan="4">0 (0.00%)</td></tr>
<tr><td>Fortran, Int64</td></tr>
<tr><td>C, Int32</td><td rowspan="2">0 (0.00%)</td><td rowspan="2">2 (0.00%)</td></tr>
<tr><td>C, Int64</td></tr>
</table>

4. Заключение

Несмотря на большое сообщество разработчиков и оперативную поддержку новых архитектур, библиотека OpenBLAS имеет ряд проблем, связанных с невысоким тестовым покрытием BLAS-функционала и включенными в состав библиотеки, очевидно без должного тестирования, сконвертированными на язык C исходными файлами с LAPACK функционалом из библиотек группы Netlib. Эти проблемы были выявлены при анализе Generic и RVV 0.7.1 сборок OpenBLAS.

Тестовое покрытие BLAS-функционала, собранное при помощи утилиты LCOV, не превышало 79% для обеих сборок. С целью продуктизации библиотеки были покрыты тестами функции с начальным покрытием менее 60%, что позволило повысить суммарное покрытие BLAS функций второго и третьего уровня до 92% и выше, а покрытие функций с RISC-V оптимизациями – до 96-99%. Благодаря добавленным тестам были обнаружены и устранены несколько ошибок в BLAS-функционале библиотек OpenBLAS.

При попытке запуска LAPACK тестов было обнаружено, что значительная часть тестов не запускается в случае сборки OpenBLAS на основе сконвертированных файлов с кодом LAPACK функций. Были локализованы ошибки, содержащиеся в сконвертированных файлах

из репозитория OpenBLAS. Их устранение позволило исправить ситуацию с запуском LAPACK тестов для Generic сборки. Отладка RVV 0.7.1 сборки показала, что падения и зависания тестов LAPACK функционала связаны с ошибками в реализации векторных оптимизаций в BLAS функциях `idamax()`, `dgemm()` и `nrm2()`, которые не обнаруживались без LAPACK тестов, несмотря на развитую систему тестирования BLAS-функционала.

В дальнейшем планируется внести полученные изменения (новые тесты и исправления ошибок) в проект OpenBLAS. Кроме того, становится возможным перейти к портированию на RISC-V открытых HPC библиотек, использующих OpenBLAS, таких как, например, библиотека BEM++ для моделирования методом граничных элементов.

Список литературы / References

[1]. Charpa S., Canale R. Numerical Methods for Engineers. McGraw Hill, 2014. 992 p.
[2]. Moin P. Fundamentals of Engineering Numerical Analysis. Cambridge University Press, 2010. 256 p.
[3]. Zienkiewicz O. C., Taylor R. L. The Finite Element Method: Its Basis and Fundamentals. Butterworth-Heinemann, 2013. 756 p.
[4]. Katsikadelis J. T. The Boundary Element Method for Engineers and Scientists, Second Edition: Theory and Applications. Academic Press, 2016. 464 p.
[5]. Brebbia C. A., Walker S. Boundary Element Techniques in Engineering. Newnes, 2016. 210 p.
[6]. Gwinner J., Stephan E. P. Advanced Boundary Element Methods: Treatment of Boundary Value, Transmission and Contact Problems. Springer, 2018. 670 p.
[7]. Garney S., Heroux M. A., Li G., Pozo R., Remington K. A., Wu K. A Revised Proposal for a Sparse BLAS Toolkit, Available at: https://sdm.lbl.gov/~kewu/ps/SparseBLAS96.pdf, accessed 23.10.2023.
[8]. Saad Y., Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2003. 184 p.
[9]. Lawson C. L., Hanson R. J., Kincaid D. R., Krogh F. Basic linear algebra subprograms for FORTRAN usage. ACM Transactions on Mathematical Software, vol. 5, issue 3, 1979, pp. 308-323. DOI: 10.1145/355841.355847.
[10]. Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Croz J. D., Greenbaum A., Hammarling S., McKenney A., Sorensen D. LAPACK Users' Guide. Society for Industrial and Applied Mathematics, 1987. 429 p.
[11]. BEM++, Available at: https://github.com/bempp/bempp-legacy/tree/master, accessed 23.10.2023.
[12]. Лукашин П.С., Стрижак С.В., Щеглов Г.А. Тестирование возможностей открытого кода BEM++ по решению задач акустики. Труды ИСП РАН, том 29, вып. 1, 2017, стр. 39-52. / Lukashin P.S., Strijhak S.V., Shcheglov G.A. Validation of open-source BEM++ code for simulation of acoustics problems. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 1, 2017, pp. 39-52 (in Russian).
[13]. OpenBLAS, Available at: https://github.com/OpenMathLib/OpenBLAS, accessed 23.10.2023.
[14]. Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms, Available at: https://gitlab.com/libeigen/eigen, accessed 23.10.2023.
[15]. Armadillo: C++ Library for Linear Algebra & Scientific Computing, Available at: https://gitlab.com/conradsnicta/armadillo-code, accessed 23.10.2023.
[16]. Cui E., Li T., Wei Q. RISC-V Instruction Set Architecture Extensions: A Survey. IEEE Access, 2023, pp. 1-16. DOI: 10.1109/ACCESS.2023.3246491.
[17]. The European High Performance Computing Joint Undertaking (EuroHPC JU), Available at: https://eurohpc-ju.europa.eu/index_en, accessed 23.10.2023.
[18]. Occamy, Available at: https://pulp-platform.org/occamy/, accessed 23.10.2023.
[19]. The EUPILOT Project, Available at: https://eupilot.eu/, accessed 23.10.2023.
[20]. PRIMME: PReconditioned Iterative MultiMethod Eigensolver, Available at: https://github.com/primme/primme, accessed 23.10.2023.
[21]. BLAS-Tester, Available at: https://github.com/xianyi/BLAS-Tester, accessed 23.10.2023.
[22]. T-HEAD GNU Compiler Toolchain, Available at: https://github.com/T-head-Semi/xuantie-gnu-toolchain, accessed 23.10.2023.
[23]. LCOV, Available at: https://github.com/linux-test-project/lcov, accessed 23.10.2023.

Информация об авторах / Information about authors

Ксения Алексеевна Зайцева является младшим инженером-программистом Департамента разработки высокопроизводительных библиотек компании YADRO. Ее научные интересы включают низкоуровневые оптимизации, генераторы случайных чисел.

Ksenia Alexeyevna Zaytseva is a junior software engineer of the Department of High Performance Libraries Development of YADRO. Her research interests include low-level optimizations, random number generators.

Валерия Валентиновна ПУЗИКОВА – кандидат физико-математических наук, эксперт по разработке программного обеспечения Департамента разработки высокопроизводительных библиотек компании YADRO. Сфера научных интересов: решатели и предобуславливатели СЛАУ, разработка прикладных математических программ, вычислительная гидродинамика, физические движки для AR/VR, высокопроизводительные вычисления, численные методы.

Valeria Valentinovna PUZIKOVA – Cand. Sci. (Phys.-Math.), Software Development Expert in High Performance Libraries Department, YADRO. Research interests: solvers and preconditioners for SLAE, applied mathematics software development, computational hydrodynamics, physics engines for AR/VR, high performance computations, numerical methods.

Андрей Дмитриевич СОКОЛОВ является инженером-программистом Департамента разработки высокопроизводительных библиотек компании YADRO. Его научные интересы включают разработку и тестирование математических библиотек, низкоуровневые оптимизации, вычисления на GPU, сверточные нейронные сети.

Andrey Dmitrievich SOKOLOV is a software engineer of the Department of High Performance Libraries Development of YADRO. His research interests include development and testing of mathematical libraries, low-level optimizations, GPU computing, convolutional neural networks.

DOI: 10.15514/ISPRAS-2023-35(5)-8

Разработка доверенных средств проектирования ИС в базисе гетерогенных ПЛИС

С.В. Гаврилов, ORCID: 0000-0003-0566-4482 <*s00v@yandex.ru*>
Д.А. Железников, ORCID: 0000-0003-2477-1793 <*zheleznikov_d@ippm.ru*>
М.А. Заплетина, ORCID: 0000-0001-9845-7823 <*zapletina_m@ippm.ru*>
И.В. Тиунов, ORCID: 0009-0008-6241-5451 <*tiunov_i@ippm.ru*>
В.М. Хватов, ORCID: 0000-0003-2285-4341 <*khvatov_v@ippm.ru*>
Р.Ж. Чочаев, ORCID: 0009-0007-0304-0496 <chochaev_r@*ippm.ru*>
Д.Б. Шокарев, ORCID: 0009-0009-2806-1570 <shokarev_d@*ippm.ru*>
Институт проблем проектирования в микроэлектронике РАН,
124365, Россия, г. Москва, г. Зеленоград, ул. Советская, д. 3.

Аннотация. Данная статья посвящена разработке доверенных средств проектирования цифровых схем в базисе гетерогенных программируемых логических интегральных схем (ПЛИС). Проектирование гетерогенных ПЛИС представляет собой одно из наиболее активно развивающихся направлений в российской микроэлектронике в настоящее время. В статье рассматриваются основные проблемы и вызовы, связанные с разработкой целевых доверенных средств проектирования. Авторы предлагают актуальный подход к разработке системы автоматизированного проектирования, основанный на использовании программных средств с открытым исходным кодом совместно с собственными наработками для её критически важных компонентов. Такой подход позволяет повысить эффективность и надёжность процесса проектирования в базисе гетерогенных ПЛИС. В статье рассмотрены такие этапы маршрута проектирования цифровых схем в базисе гетерогенных ПЛИС, как логический синтез и технологическое отображение, этапы топологического синтеза и статического временного анализа. Работа представляет интерес для специалистов в области микроэлектроники, а также для исследователей, занимающихся разработкой средств и систем проектирования ИС. Результаты исследования способствуют улучшению существующих методов и инструментов проектирования ИС, а также развитию и расширению отечественной электронной компонентной базы.

Ключевые слова: СБИС; ПЛИС; физическое проектирование; топологический синтез; логический синтез; размещение; трассировка; доверенное программное обеспечение.

Для цитирования: Гаврилов С.В., Железников Д.А., Заплетина М.А., Тиунов И.В., Хватов В.М., Чочаев Р.Ж., Шокарев Д.Б. Разработка доверенных средств проектирования ИС в базисе гетерогенных ПЛИС. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 107–126. DOI: 10.15514/ISPRAS–2023–35(5)–8.

Development of the Trusted Tools for IC Design on Heterogeneous FPGAs

S.V. Gavrilov, ORCID: 0000-0003-0566-4482 <*s00v@yandex.ru*>
D.A. Zheleznikov, ORCID: 0000-0003-2477-1793 <*zheleznikov_d@ippm.ru*>
M.A. Zapletina, ORCID: 0000-0001-9845-7823 <*zapletina_m@ippm.ru*>
I.V. Tiunov, ORCID: 0009-0008-6241-5451 <*tiunov_i@ippm.ru*>
V.M. Khvatov, ORCID: 0000-0003-2285-4341 <*khvatov_v@ippm.ru*>
R.Z. Chochaev, ORCID: 0009-0007-0304-0496 <chochaev_r@*ippm.ru*>
D.B. Shokarev, ORCID: 0009-0009-2806-1570 <shokarev_r@*ippm.ru*>

Institute for Design Problems in Microelectronics of RAS,
3, Sovetskaya st., Zelenograd, Moscow, 124365, Russia.

Abstract. This paper focuses on the development of trusted tools for designing digital circuits in the basis of heterogeneous field programmable gate arrays (FPGAs). Designing heterogeneous FPGAs is one of the most actively growing areas in Russian microelectronics at present. The paper discusses the main problems and challenges associated with the development of trusted computer-aided design tools. The authors propose a relevant approach to the development of a computer-aided design system based on the use of open-source software tools together with proprietary developments for its critical components. This approach allows to increase the efficiency and reliability of the design process in the basis of heterogeneous FPGAs. The paper considers such stages of the design flow in the basis of heterogeneous FPGAs as logic synthesis and technology mapping, different stages of layout synthesis and static timing analysis. The work is of interest to specialists in the field of microelectronics, as well as to researchers involved in the development of IC design tools and systems. The research results contribute to the improvement of existing IC design methods and tools, as well as to the development and expansion of the Russian electronic component base.

Keywords: VLSI; FPGA; layout synthesis; logic synthesis; placement; routing; trusted software.

For citation: Gavrilov S.V., Zheleznikov D.A., Zapletina M.A., Tiunov I.V., Khvatov V.M., Chochaev R.Z., Shokarev D.B. Development of the Trusted Tools for IC Design on Heterogeneous FPGAs. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 107-126 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-8.

1. Введение

Одним из наиболее активно развивающихся направлений российской микроэлектроники в настоящее время является разработка программируемых логических интегральных схем (ПЛИС). Отечественные ПЛИС постепенно выходят на российский рынок и завоёвывают свою долю популярности на фоне сложившихся сложностей с получением их зарубежных аналогов. Модельный ряд программируемых микросхем расширяется и переходит от классических гомогенных решений на основе регулярных структур из программируемых логических блоков (ПЛБ) к более сложным гетерогенным вариантам, включающим специализированные программируемые сложно-функциональные блоки (СФ-блоки).

Разработка интегральных схем с применением ПЛИС неотрывно связана с применением специализированных систем автоматизированного проектирования (САПР). Чтобы следовать в ногу за постоянно расширяющейся номенклатурой отечественных ПЛИС, ИППМ РАН занимается разработкой доверенных средств автоматизированного проектирования в базисе новейших программируемых микросхем, комплексно учитывающих особенности и специфику применения последних. Выбранная стратегия решения задачи создания специализированных программных средств заключается в сочетанном применении открытых программ проектирования, анализа и оптимизации схем в базисе ПЛИС и собственных разработок коллектива ИППМ РАН в области логического и топологического синтеза интегральных схем.

Разрабатываемая доверенная САПР X-CAD построена по модульному принципу и включает следующие составные части: графическую оболочку пользователя; набор подключаемых

внешних программ для технологически-независимого логического синтеза, программу X-Map для технологического отображения и логического ресинтеза, программный модуль X-Place для работы с размещением логических вентилей, СФ-блоков и контактов проектируемой схемы; внешние программные модули для кластеризации, быстрого анализа временных характеристик и функционального моделирования, а также ядро системы для выполнения топологического синтеза, обработки прошивки и загрузки её в конфигурационную память программируемого кристалла ПЛИС.

В данной работе рассматриваются ключевые архитектурные особенности некоторых целевых отечественных ПЛИС, включая гетерогенные, а также программные средства, применяемые в маршруте проектирования ИС в их базисе. Дальнейшее содержание работы организовано следующим образом. В разделе 2 рассматриваются используемые в X-CAD инструменты логического синтеза и технологического отображения. В разделе 3 приведен обзор применяемых методов и алгоритмов топологического проектирования. Раздел 4 посвящён инструментам статического временного анализа.

2. Логический синтез и технологическое отображение

В области проектирования интегральных схем под логическим синтезом понимается процесс трансляции RTL-описания проектируемого устройства с языка описания аппаратуры в так называемый список соединений логических вентилей. На данном этапе решаются задачи преобразования исходного представления схемы в многоуровневую или одноуровневую логическую сеть (или Булеву сеть), логической оптимизации полученной сети (например, по площади или быстродействию), и отображения этой сети в список соединений логических вентилей. Целевой логический базис, в который происходит отображение, определяется, как правило, заданной библиотекой элементов, передаваемой на вход средствам логического синтеза вместе с RTL-описанием схемы. Элементы такой библиотеки всегда оптимизированы под целевую технологию. Такой подход к отображению называется технологически ориентированным.

В маршруте проектирования схем в базисе ПЛИС и других программируемых устройств, основанных на таблицах соответствия (ТС, от англ. «LUT» – Lookup Table), помимо отображения в предопределённый библиотечный базис, также существует и технологически независимый подход. В этом случае исходная схема преобразуется в набор промежуточных LUT-элементов от k-входов, или k-LUT. Здесь k – количество входов комбинационной части ПЛБ ПЛИС, которая в подавляющем большинстве архитектур представлена таблицей соответствия. Таким образом, при таком подходе, синтез не ограничен библиотекой элементов и логический базис формируется из полного набора возможных функций от k входов. Полученный список соединений, как правило, не может быть транслирован в конкретную архитектуру без дополнительных преобразований и требует внесения корректировок, учитывающих особенности целевого кристалла ПЛИС.

В разрабатываемой доверенной САПР X-CAD за этап логического синтеза отвечает инструмент с открытым исходным кодом – Yosys [1]. На текущий момент Yosys считается эталонным программным средством логического синтеза с открытым исходным кодом и используется повсеместно. За счет открытости исходного программного кода он обеспечивает довольно высокую степень доверенности, однако детальный анализ используемых алгоритмов и структур данных не проводился. К сожалению, на данный момент, Yosys является единственным возможным выбором и наиболее узким местом в процессе построения полностью доверенного маршрута проектирования, в следствии полного отсутствия отечественного аналога инструмента логического синтеза.

Программное обеспечение Yosys, а вместе с ним и САПР X-CAD, поддерживают оба упомянутых подхода к логическому синтезу: как технологически ориентированный – отображение в заданную библиотеку элементов, так и не зависящий от технологии – отображение в k-LUT элементы. С помощью Yosys в X-CAD, кроме отображения схем в

набор вентилей, также осуществляется экстракция и отображение сложно-функциональных блоков и частичная генерация блоков ввода/вывода.

Несмотря на то, что Yosys хорошо зарекомендовал себя в качестве средства логического синтеза схем для ПЛИС, на данный момент он поддерживает лишь некоторые из зарубежных архитектур [2] и при этом – ни одной отечественной. В связи с этим, в X-CAD используется дополнительное средство для трансляции результатов синтеза Yosys в специальный внутренний формат САПР на основе языка Tcl – инструмент X-Map. Помимо преобразования Verilog-описания, создаваемого Yosys, во внутреннее Tcl-описание, он также используется в маршруте для проведения технологически ориентированного ресинтеза, а также генерации необходимых периферийных и управляющих блоков ячеек ввода/вывода. Результат работы X-Map формируется в виде внутреннего формата на основе языка Tcl - библиотеки элементов и списка соединений. Также, X-Map, при необходимости, может дополнительно сформировать результат в виде набора файлов на языке Verilog – библиотек элементов и списка соединений. Это может потребоваться, если необходимо передать результаты обработки в другие инструменты маршрута, не поддерживающих Tcl-описание. Кроме того, результат может быть дополнительно сформирован в DOT формате для дальнейшей визуализации схемы в виде графа (например, с помощью программы Graphviz).

2.1 Отображение логических элементов и логический ресинтез

Каждая из архитектур ПЛИС (как зарубежных, так и отечественных) имеет уникальный набор особенностей, учёт которых влияет не только на конечные характеристики проектируемой схемы, но и на саму возможность реализации её в целевой ПЛИС. В САПР X-CAD за учёт этих особенностей отвечает инструмент X-Map, поддерживающий маршрут синтеза схемы как в рамках заданной библиотеки, так и на основе k-LUT элементов. Результат логического синтеза в виде Verilog-описания, вместе с библиотеками элементов во внутреннем формате и командным сценарием, передаётся в инструмент X-Map, который проводит дополнительную обработку схемы, позволяя подготовить её к имплементации на ПЛИС, выдавая в качестве выходных данных список соединений уже не просто логических вентилей, а экземпляров ПЛБ, запрограммированных на выполнение необходимых функций. Эти выходные данные и составляют внутренний формат САПР X-CAD на основе языка Tcl: список соединений элементов и библиотеки ячеек, содержащих информацию о программировании элементов: ПЛБ, ЯВВ и СФ-блоков.

Возможности X-Map в контексте задач ресинтеза и упаковки покрывают полный набор особенностей архитектур поддерживаемых отечественных ПЛИС. Кроме того, с помощью набора команд управляющего файла сценария, X-Map может быть настроен на новую архитектуру с указанием какие из возможностей по ресинтезу и упаковке поддерживаются. При этом, на данный момент, алгоритмические возможности инструмента не могут быть расширены пользователем извне, однако работа по разработке инструментов гибкой настройки процесса работы X-Map ведется.

2.1.1 Технологически ориентированный ресинтез

В случае, когда программа Yosys выполняет отображение в заданный набор библиотечных элементов, их дополнительное преобразование требуется только в том случае, когда для реализации схемы в целевой ПЛИС необходимо внести изменения в структуру самой схемы. Под «ресинтезом» здесь следует понимать процесс преобразования схемы с учетом особенностей целевой архитектуры: преобразование логических элементов (изменение подключений и логических функций), генерация источников земли и питания, вставка буферов и инверторов и др.

Например, в архитектуре ПЛИС 5400ТС015 разработки АО «Дизайн центр «Союз» [3] сигнал с тактового дерева ПЛИС может приходить только на определённый комбинационный вход

ПЛБ. Для того чтобы учесть эту особенность архитектуры, вентиль, на вход которого приходит глобальный синхросигнал, необходимо преобразовать таким образом, чтобы этот сигнал попадал на подходящий вход. Это означает, что необходимо изменить не только подключение, но и логическую функцию, выполняемую этим вентилем. Программа X-Map осуществляет необходимые преобразования, после чего элемент с получившейся логической функцией необходимо отобразить в заданную библиотеку. Если подходящий по функции элемент в библиотеке не найден, программа возвращает прежний вариант вентиля и внедряет в схему буфер, проходя через который, тактовый сигнал попадает на нужный вход этого вентиля.

Кроме описанного выше, X-Map может, при необходимости, генерировать дополнительные источники питания (VDD) и земли (GND) в проектируемой схеме, учитывая при этом особенность архитектуры некоторых отечественных ПЛИС – наличие на отдельных входах ПЛБ собственных источников VDD- или GND-сигналов.

2.1.2 Упаковка элементов

Подготовка схемы для имплементации в ПЛИС – не единственная задача X-Map. Основная его задача – упаковка элементов в ПЛБ и формирование набора сигналов для их программирования. В большинстве архитектур ПЛИС в ПЛБ присутствует как программируемая комбинационная часть, реализованная в виде k-LUT элемента, так и последовательностная часть, представленная триггером (как правило, D-триггером). Программный модуль X-Map, используя внутренние межсоединения, может объединять в одном ПЛБ комбинационный и последовательностный вентили. При этом инструмент может учитывать особенности и ограничения внутренней структуры программируемого блока. Например, в ПЛИС 5510XC3AT разработки АО «НИИМЭ» [4], в ПЛБ, основанной на 4-LUT, встроенный D-триггер может получать данные либо с выхода LUT, либо с одного из четырех информационных входов ПЛБ (рис. 1).

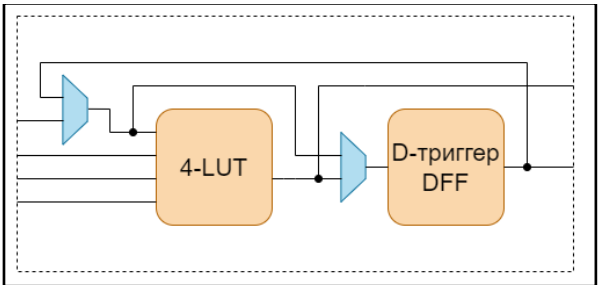


Рис. 1. Обобщенная структура ПЛБ ПЛИС 5510XC3AT.
Fig. 1. Generalized structure of 5510XC3AT FPGA's PLB.

Такая особенность ПЛБ позволяет объединить последовательно соединенные 4-LUT и триггер, или 3-LUT с триггером, независящие друг от друга (рис. 2).

В архитектуре ПЛИС 5400TC015, в отличие от 5510XC3AT, комбинационная часть представлена элементом 3-LUT, а для триггера имеется выделенный вход данных, что позволяет объединить его с независимым LUT-элементом с функцией от всех трех информационных входов (рис. 3).

В каждой из упомянутых выше архитектур также присутствует и другая особенность – наличие обратной связи с выхода триггера на вход LUT, что позволяет объединить элементы, как показано на рис. 4.

В процессе упаковки также существуют два подхода к объединению в случае множественной нагрузки: обычное объединение с активацией дополнительного входа и объединение с копированием [5]. В первом случае объединение с триггером происходит стандартным образом, описанным выше, а второй триггер подключается через прямой выход с LUT, как

показано на рис. 5. Такой вариант объединения через прямой выход LUT – очевидное решение в такой ситуации.

Второй вариант объединения – объединение с копированием – продемонстрирован на рис. 6. Данный режим используется в X-Map по умолчанию. Его особенность заключается в том, что для комбинационного элемента создаётся копия, и затем каждый из двух элементов (оригинал и копия) объединяется со своим триггером. Это позволяет синхронизировать приход сигналов на каждый из триггеров, но при этом уменьшает общее количество доступных комбинационных блоков ПЛИС, что может повлиять, например, на возможность независимого объединения с другими элементами, и на реализуемость межсоединений схемы на этапе трассировки.

Благодаря использованию подобных особенностей архитектуры может быть достигнуто значительное улучшение по площади, выражаемое в количестве использованных ячеек ПЛИС и количестве межсоединений, участвующих в трассировке, относительно исходного синтезированного описания [6].

Помимо упомянутых возможностей, инструмент X-Map также учитывает и ограничение на количество подключений ПЛБ к тактовому дереву, если таковые имеются.

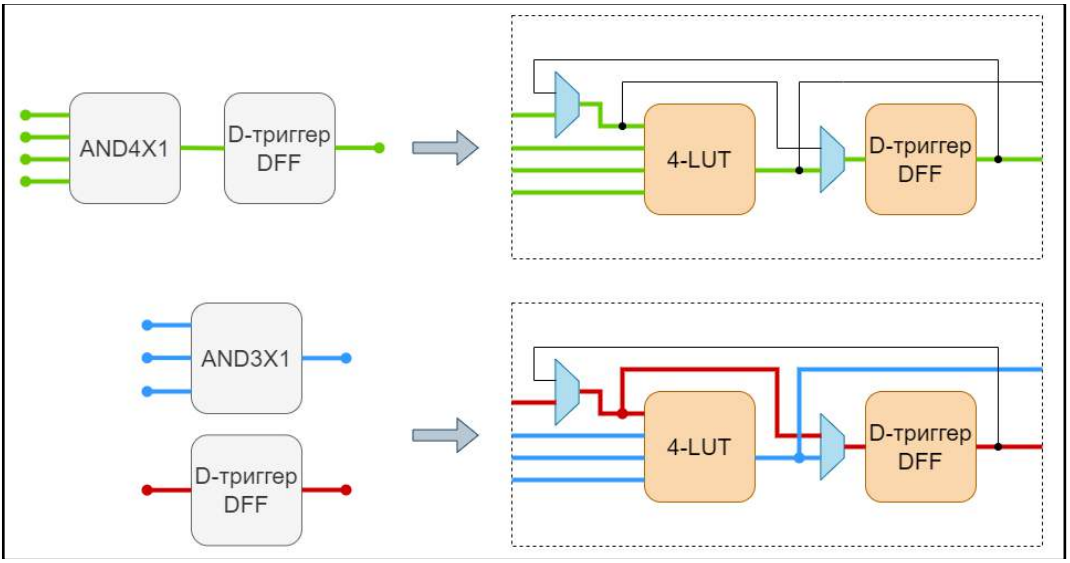


Рис. 2. Пример использования архитектурных особенностей ПЛИС 5510XC3AT в процессе упаковки элементов.

Fig. 2. An example of using the architectural features of FPGA 5510XC3AT during the packing stage.

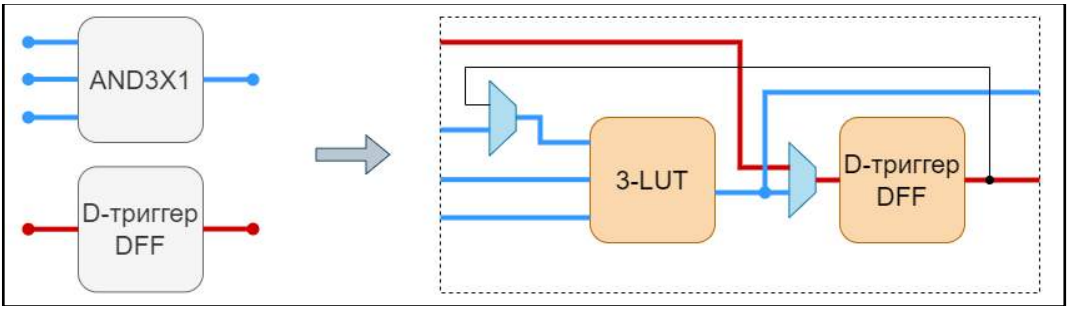


Рис. 3. Пример использования архитектурных особенностей ПЛИС 5400TC015 в процессе упаковки элементов.

Fig. 3. An example of using the architectural features of FPGA 5400TC015 during the packing stage.

2.2 Отображение контактов ввода/вывода

Отображение входных, выходных и двунаправленных контактов из функционального описания разрабатываемой схемы в библиотечные ячейки ввода/вывода (ЯВВ) в разрабатываемой САПР X-CAD выполняется совместно с помощью программы Yosys и инструмента X-Map. Yosys генерирует в процессе логического синтеза экземпляры библиотечных ЯВВ для двунаправленных контактов, а после окончания его работы на этапе ресинтеза полученного списка соединений для генерации экземпляров ЯВВ для однонаправленных входных и выходных контактов используется X-Map.

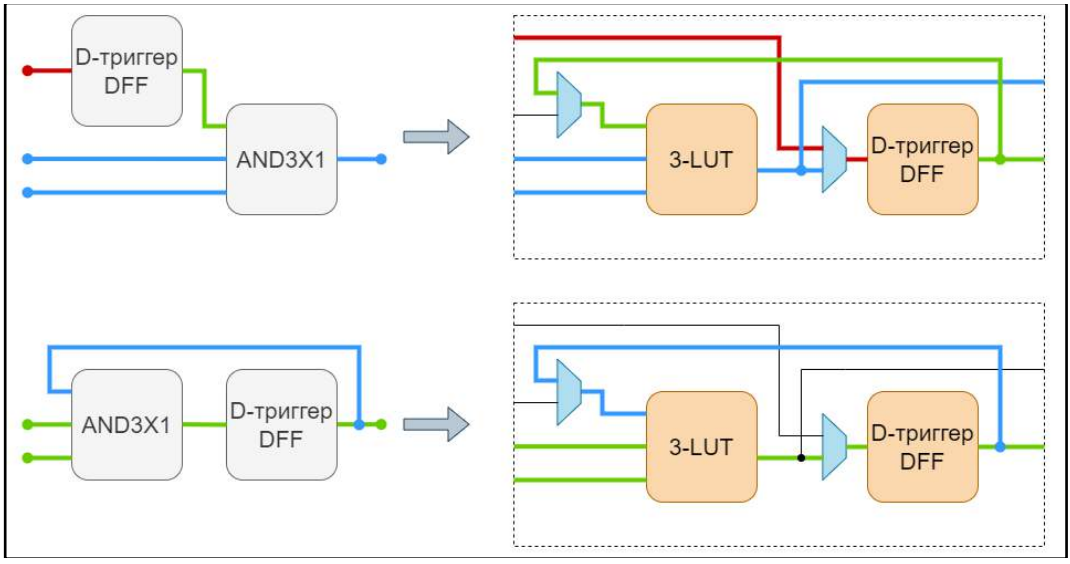


Рис. 4. Пример использования обратной связи в ПЛБ ПЛИС 5400ТС015.
Fig. 4. An example of using the feedback in 5400TC015 FPGA's PLB.

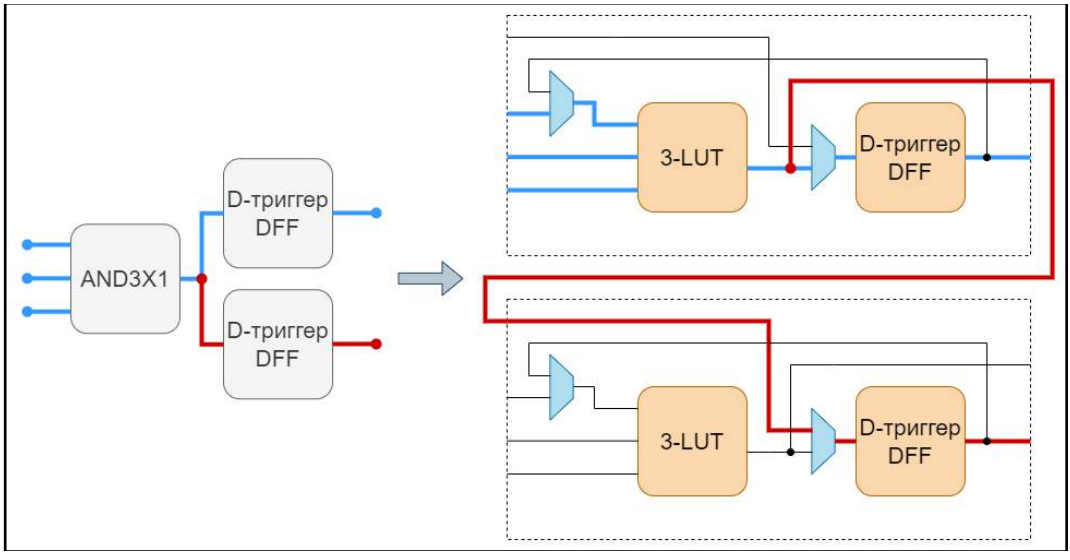


Рис. 5. Пример упаковки элементов для ПЛИС 5400ТС015 в случае нескольких триггеров, соединенных с выходом комбинационного элемента.
Fig. 5. Example of packing elements for FPGA 5400TC015 in the case of several triggers connected to the output of a combinational element.

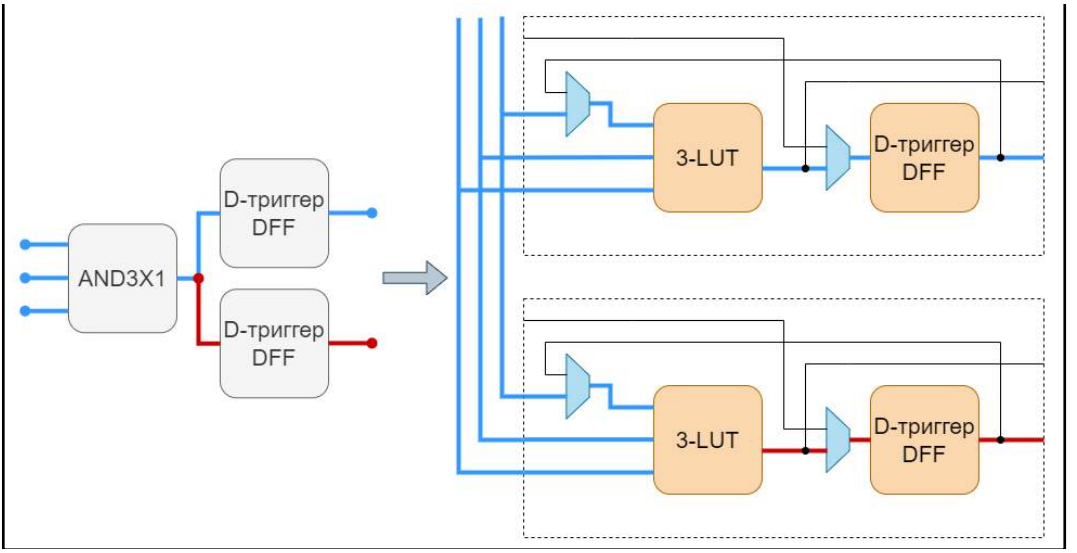


Рис. 6. Пример упаковки элементов для ПЛИС 5400ТС015 в случае нескольких подсоединенных к элементу триггеров с копированием источника сигнала.
Fig.6. Example of packing elements for FPGA 5400TC015 in the case of several triggers connected to the output of a combinational element with copying.

Для генерации двунаправленных ЯВВ ПО Yosys имеет специальную команду «iopadmap», которая преобразует внешние контакты проектируемой схемы в элементы базиса ПЛИС. Данной команде требуется указать имя библиотечного элемента требуемой ячейки, его входные и выходные контакты для передачи данных, а также контакт для управления третьим состоянием выхода, соединенного с периферией ПЛИС.

X-Map позволяет гибко настроить процесс генерации ячеек для связи с периферией под различные архитектуры ПЛИС и сформировать ЯВВ с учётом их особенностей, получая данные о доступных ЯВВ из библиотеки, разработанной во внутреннем формате САПР с помощью лингвистических средств языка Tcl, а также из управляющего сценария, содержащего дополнительные команды и данные, необходимые для работы программы. Одной из особенностей архитектуры каждой ПЛИС является количество глобальных тактовых сигналов. В случае, когда количество тактовых сигналов схемы превышает количество возможных ЯВВ ПЛИС для тактового сигнала, программа генерирует для недостающих тактовых сигналов обычные информационные ЯВВ. Очередность генерации ячеек для тактовых сигналов, при этом, определяется количеством подключений к различным блокам ПЛИС. Также X-Map определяет ситуацию, когда контакт схемы указан как двунаправленный, но используется только в одном из направлений - вход или выход. В этом случае программа генерирует для контакта не двунаправленную ЯВВ, а ячейку нужного направления.

Задать требуемый библиотечный элемент ЯВВ и назначить его на определенный контакт ПЛИС в программе X-CAD возможно с помощью графического интерфейса X-Place. ЯВВ, назначенные таким образом, имеют приоритет над указанными в управляющем сценарии X-Map для автоматической генерации. Генерация ЯВВ в X-Map, по умолчанию выполняется для контактов разрабатываемой схемы, не имеющих соединения с экземпляром ЯВВ, и при необходимости может быть отключена.

Кроме ячеек ввода/вывода в библиотечный базис ПЛИС, интегрированных в X-CAD, добавлены блоки управления ячейками ЯВВ. Блоки управления присутствуют только в ряде архитектур ПЛИС, например, в архитектуре 5510ХС3АТ, и необходимы для

конфигурирования режима работы ячейки ввода/вывода. Они позволяют включить передачу данных от периферии к ПЛИС, от ПЛИС к периферии или в оба направления одновременно. С помощью представленных блоков можно активировать дополнительную задержку сигнала на ячейке ЯВВ, регистры, синхронизированные с ПЛИС, или режим удвоенной передачи данных (англ. Double Data Rate, DDR).

По умолчанию, для каждого из контактов функционального описания разрабатываемой схемы наряду с ЯВВ генерируется блок управления в стандартном режиме прямой передачи данных. В случае необходимости изменения режима работы ЯВВ, в схеме может быть вызвана требуемая конфигурация блока управления, доступная в библиотеке ПЛИС. X-Map позволяет настроить генерацию этих блоков, учитывая особенность их подключения в ПЛИС. Кроме того, X-Map не только учитывает уже имеющиеся во входном описании ЯВВ и управляющие блоки и генерирует недостающие, но и проверяет все имеющиеся (в том числе и заданные пользователем) ЯВВ и управляющие блоки на соответствие правлению порта (вход, выход или двунаправленный вывод) и его назначению, т.е. какого типа сигнал: информационный, синхросигнал, сигнал сброса, сигнал установки, а также подключен ли он к источнику земли или питания.

2.3 Отображение СФ-блоков

Кроме программируемых логических блоков и ячеек ввода/вывода гетерогенные ПЛИС могут содержать программируемые сложно-функциональные блоки (СФ-блоки) двух типов: жёсткие и гибкие [7]. Жёсткие СФ-блоки представляют собой полузаказные схемы, которые интегрируются в архитектуру ПЛИС, располагаясь в строках или столбцах регулярной матрицы ПЛБ. Они имеют общие коммутационные ресурсы с регулярной частью ПЛИС и, как правило, программируются с помощью конфигурационной памяти, также осуществляющей управление ПЛБ, ЯВВ и коммутационными элементами. В отличие от жёстких, гибкие СФ-блоки разрабатываются на основе логических элементов ПЛИС. Они нацелены на использование особенностей структуры групп ПЛБ и специализированных коммутационных ресурсов, не доступных при логическом синтезе на основе стандартных логических элементов. К данным ресурсам могут относиться цепи ускоренного переноса, связывающие соседние ПЛБ, цепи для прямой связи выходов со входами данных LUT соседнего ПЛБ, а также цепи для прямой связи выхода триггера со входом данных или входом тактового сигнала соседнего триггера.

Оба типа СФ-блоков разрабатываются под конкретные вычислительные задачи, что позволяет оптимизировать их структуру и схемотехнику таким образом, чтобы характеристики схем, спроектированных на их основе, были выше, чем у схем, спроектированных на основе стандартных ПЛБ ПЛИС. Использование СФ-блоков приводит к уменьшению количества используемых ПЛБ и коммутационных ресурсов, а также к повышению быстродействия разрабатываемых схем [8].

В САПР X-CAD автоматический синтез функционального описания схемы с использованием СФ-блоков и их технологическое отображение в библиотечный базис ПЛИС реализуется с помощью ПО Yosys [9]. На первом этапе логического синтеза Yosys производит экстракцию СФ-блоков из функционального описания схемы и на основе полученных результатов формирует элемент из встроенного базиса параметризованных блоков [10]. Набор блоков, которые могут быть идентифицированы таким образом, ограничен. Он включает в себя полные сумматоры, полусумматоры, вычитатели, умножители, счётчики, сдвиговые регистры, блоки цифрового процессора обработки сигналов и блоки памяти. На втором этапе Yosys выполняет отображение полученного элемента в технологически-зависимый СФ-блок из переданной ему библиотеки, разработанной под конкретную архитектуру ПЛИС. Данная библиотека разрабатывается на языке Verilog с добавлением специализированных конструкций ПО Yosys. Каждый её элемент представляет собой СФ-блок ПЛИС, сконфигурированный под определённый режим работы. Как элементы внутреннего базиса,

так и элементы разработанной библиотеки для технологического отображения, не классифицируются в Yosys по типам и являются для него «черными ящиками» с определённым набором параметров.

Связь СФ-блоков, полученных в процессе логического синтеза, с СФ-блоками из схемотехнического описания ПЛИС, т. е. с подсхемами ПЛИС, реализуется в X-CAD при помощи библиотеки для топологического синтеза, основанной на разработанном интерфейсе и лингвистических средствах языка Tcl. С помощью данной библиотеки выполняется интеграция СФ-блоков в САПР для топологического синтеза и их программирование. В состав библиотеки входят как элементы из библиотеки для технологического отображения, так и СФ-блоки, для которых невозможна автоматическая экстракция из функционального описания, в связи с их отсутствием во внутреннем базисе Yosys.

СФ-блоки для отображения в подсхемы ПЛИС, в отличие от СФ-блоков из библиотеки технологического отображения, разделены на жёсткие и гибкие. Логический базис каждого семейства ПЛИС имеет в своем составе различный набор СФ-блоков. В базис семейства ПЛИС 5510ХС входят только гибкие СФ-блоки – сумматоры с ускоренным переносом, многоразрядные асинхронные счетчики и сдвиговые регистры, в состав 5510ТС, в зависимости от конкретной ПЛИС, входят как гибкие СФ-блоки, аналогичные тем, что представлены в семействе 5510ХС, так и жесткие – умножители, блоки памяти и цифровые процессоры обработки сигналов. Для добавления жёстких блоков в библиотеку используется имя подсхемы СФ-блока в ПЛИС, её информационные контакты, доступные при вызове экземпляра библиотечного элемента в проекте пользователя, а также конфигурационные контакты, которым, в соответствии с требуемым режимом работы, заданы постоянные значения логического 0 или 1 [11]. При добавлении гибких блоков используются экземпляры библиотечных ПЛБ, имена подсхем ПЛИС, их информационные и конфигурационные контакты. В библиотечном гибком СФ-блоке доступно определение ограничений топологического синтеза, к которым относится размещение ПЛБ относительно друг друга, соединение их между собой и назначение межсоединениям ПЛИС конкретных цепей из библиотечного СФ-блока. Корректное конфигурирование и назначение ограничений позволяет учесть особенности архитектуры жёстких блоков, а также особенности архитектуры ПЛИС, преимущества структуры группы ЛБ и специализированные коммутационные ресурсы при проектировании гибких СФ-блоков.

3. Топологический синтез

На этапе топологического синтеза в маршруте проектирования в базисе ПЛИС выполняется размещение элементов схемы и трассировка существующих соединений [12]. Традиционно процесс размещения включает в себя этапы глобального и детального размещения [13]. На глобальном размещении выполняется определение предварительного расположения элементов на кристалле, после чего на этапе детального размещения выполняется легализация, необходимая для назначения элементов на корректные посадочные площадки, и оптимизация полученного размещения по таким критериям, как длина соединений, задержки, потребляемая мощность и др. Программные модули размещения и трассировки являются собственной разработкой ИППМ РАН.

3.1 Кластеризация по группам логических блоков

В САПР X-CAD на этапе глобального размещения выполняется размещение групп программируемых логических блоков с учётом особенностей архитектуры целевой ПЛИС. Список соединений, полученный после этапа логического синтеза, представляется в виде гиперграфа $G=(V, E)$, где V – множество вершин (элементов схемы), E – множество гиперребер (соединений), после чего для формирования групп логических блоков используются специальные алгоритмы кластеризации гиперграфов.

Для кластеризации существуют различные программы и методы [14-16], среди которых наибольшей популярностью в академической среде обладает одна из самых старейших – hMETIS [17]. К сожалению, исходный код данной программы закрыт и последнее обновление было выпущено более двадцати лет назад. Среди программ с открытым исходным кодом [18-21] одним из наиболее эффективных и быстро развивающихся программ является программа KaHyPar [21]. В отличие от программ, использующих классические алгоритмы, в данной программе для получения оптимального разбиения используется комбинация алгоритмов. В программе KaHyPar пользователю доступны различные конфигурации запуска, которые позволяют выбрать требуемый алгоритм разбиения гиперграфа и целевую функцию, подходящую под конкретную задачу. Открытый исходный код программы, а также тот факт что на этапе кластеризации невозможна модификация списка соединений в следствии работы программы с абстрактным графом, данная программа была выбрана для интеграции в САПР X-CAD.

На вход KaHyPar подается гиперграф (список соединений), количество требуемых кластеров (групп логических блоков) и конфигурационный файл с настройками запуска алгоритмов. Выходной файл программы KaHyPar с кластеризованным гиперграфом конвертируется в специальный файл на языке Tcl, который загружается в САПР X-CAD. Так как гиперграф содержит вершины, представляющие как логические элементы, так и СФ-блоки из списка соединений, то при конвертации в Tcl-файл СФ-блоки удаляются из кластеров для получения только групп логических элементов.

Экспериментальные результаты, полученные в предыдущих работах [22], показали эффективность использования программы KaHyPar в маршруте проектирования в базисе ПЛИС. Однако, одним из недостатков KaHyPar является отсутствие учёта важных особенностей архитектуры групп программируемых логических блоков, например, количества доступных глобальных тактовых сигналов в пределах одной группы. Поэтому при применении программы KaHyPar в маршруте проектирования в базисе ПЛИС требуется дополнительная легализация полученного разбиения с учётом особенностей целевых архитектур ПЛИС.

3.2 Размещение логических элементов

Большинство современных ПЛИС имеет иерархическую архитектуру [12], поэтому для получения оптимального размещения обычно используются методы и алгоритмы, выполняющие размещение для каждого уровня иерархии по отдельности. В маршруте топологического синтеза в базисе ПЛИС в программе X-CAD на этапе размещения также применяется двухуровневый подход, описанный в работе [13]. Суть данного подхода заключается в разбиении размещения на глобальный и детальный этапы.

На этапе глобального размещения в X-CAD выполняется размещение групп программируемых логических блоков, полученных после кластеризации программой KaHyPar. Глобальный этап включает в себя два последовательных подэтапа: начальное размещение и оптимизация полученного размещения эвристическими методами. Начальный этап необходим для быстрой генерации предварительного размещения ПЛБ для его последующей оптимизации [23]. Существует множество различных методов и алгоритмов генерации начального размещения [24], среди которых популярны случайный и силовые методы [25]. В методе случайного размещения для каждого элемента из списка соединений последовательно генерируются случайные координаты посадочных мест. Данный подход малоэффективен, и не гарантирует детерминированных результатов. Поэтому в САПР X-CAD используется силовой алгоритм размещения [24]. В силовом алгоритме связи между размещаемыми элементами представляются в виде сил взаимного притяжения, которые определяют конечные координаты. Чем больше соединений между логическими элементами, тем выше сила притяжения и тем ближе друг к другу будут расположены элементы на кристалле. Благодаря тому, что сильно связанные элементы размещены ближе друг к другу,

достигается сокращение суммарной длины цепей и повышается скорость сходимости к оптимальному решению последующего алгоритма оптимизации размещения.

Для оптимизации начального размещения в X-CAD на глобальном этапе используются модифицированные алгоритмы размещения на основе метода имитации отжига [13]. В среде открытых и коммерческих САПР они завоевали наибольшую популярность. Преимуществами данного метода являются легкость программной реализации, простота учёта разных критериев оптимизации, возможность оптимизации нелинейных целевых функций. Основным отличием модифицированных алгоритмов является учёт архитектурных особенностей целевых ПЛИС, без которого невозможно получение легального размещения.

Этап детального размещения в X-CAD включает в себя те же шаги, что и этап глобального размещения. На первом шаге выполняется генерация начального размещения с помощью силового алгоритма для логических элементов в каждой группе с учётом результатов глобального размещения. На втором шаге выполняется оптимизация полученного размещения модифицированными алгоритмами на основе метода имитации отжига [26].

Помимо двухуровневого размещения в программе X-CAD поддерживается и плоское размещение логических элементов без предварительного этапа кластеризации [26]. В плоском варианте выполняется размещение логических элементов без учёта их расположения в группах на кристалле. Однако, чтобы не допустить слишком плотного размещения, которое может привести к нетрассируемому решению, вводится дополнительное ограничение на максимальное количество элементов в пределах групп. Экспериментальные результаты, полученные в предыдущих работах [22], показали, что в трассировке после плоского размещения используется в среднем меньше коммутационных элементов, чем после двухуровневого с алгоритмом KaHyPar, однако она работает заметно медленнее.

3.3 Трассировка межсоединений

Решение задачи трассировки в маршруте топологического синтеза в базисе ПЛИС представляет собой поиск детального отображения проектных межсоединений на имеющиеся на кристалле ПЛИС коммутационные ресурсы. Последние представлены совокупностью коммутационных магистралей и отдельных элементов, программируемых сигналами из общей конфигурационной памяти. Если на предыдущих этапах проектирования формируется часть прошивки (матрицы значений ячеек конфигурационной памяти), отвечающая за программирование логической части ПЛИС (ЯВВ, СФ-блоков и ПЛБ), то результат этапа трассировки является основой для формирования оставшейся части прошивки, определяющей программирование всей коммутационной сети ПЛИС. Обязательным требованием для успешного решения задачи трассировки является формирование целостных и непересекающихся путей для всех проектных межсоединений, исключающих короткие замыкания между ними, т.н. требование бесконфликтной разводки.

Работа модуля трассировки в составе разрабатываемой САПР X-CAD основывается на собственной реализации известного алгоритма трассировки Pathfinder [27], дополненного рядом модификаций. Алгоритм Pathfinder имеет доказанную высокую эффективность для решения задачи трассировки на ПЛИС. Он активно исследуется, развивается [28, 29] и в настоящее время применяется в составе крупнейших открытых САПР: OpenLane/OpenROAD [30], VPR [31], F4PGA [32].

Коммутационные ресурсы значительной части отечественных ПЛИС (например, серий 5510ХС, 5510ТС, схемы 5400ТС015, цифровой части 5400ТС194 и др.) могут быть формально описаны смешанным графом [33] $G=(V,E)$, вершины v_i, $v_i \in V$ которого соответствуют неразрывным коммутационным магистралям и контактам ПЛБ, СФ-блоков и ЯВВ, а ребра и дуги e_{ij}, $e_{ij} \in E$, $e_{ij}=(v_i, v_j)$ – коммутационным элементам между ними. Важные схемотехнические особенности коммутационных ресурсов этих схем снижают

эффективность применения оригинального алгоритма Pathfinder, существующих открытых решений и классической модели ориентированного графа для представления коммутационной сети, что потребовало внесения модификаций в алгоритм трассировки и разработки модели смешанного графа. В частности, такими особенностями являются применение широкого набора трассировочных элементов, в том числе, специализированных, дублирование и редуцирование коммутационных возможностей для повышения надежности схем ПЛИС и экономии площади кристалла в рамках используемой кремниевой технологии. С точки зрения доверенности собственная реализация алгоритма также позволяет гарантировать отсутствие возможности внесения сторонних изменений на этапе трассировки межсоединений.

Общий принцип работы алгоритма Pathfinder состоит в итерационной последовательной трассировке списка соединений проектируемой схемы с учётом результатов предшествующих итераций. Суммарное их количество определяется сложностью задачи трассировки. Она, в свою очередь, зависит от степени перегруженности коммутационных ресурсов, необходимых для реализации всех соединений между отдельными логическими блоками, блоками ввода-вывода и сложно-функциональными блоками схемы, расстановка которых была получена на этапе размещения. Перегруженными считаются те коммутационные магистрали и отдельные элементы, которые задействованы в трассировке более чем одного проектного межсоединения. К завершению трассировки все перегруженные области должны быть устранены, поскольку в процессе функционирования схемы на ПЛИС они способны вызвать короткое замыкание между разнопотенциальными участками цепей.

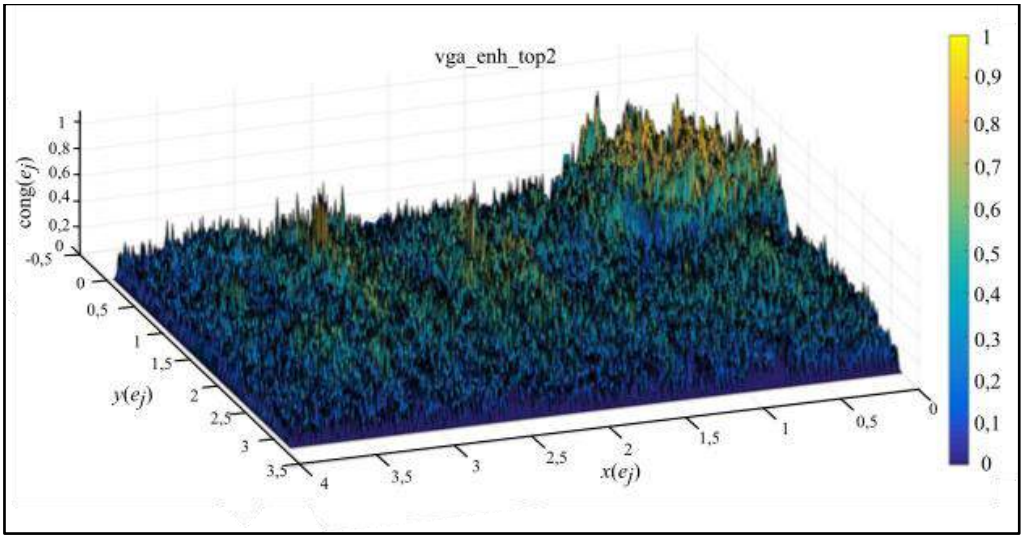


Рис. 7. Тепловая карта перегруженности коммутационных ресурсов на примере схемы vga_enh_top [31], x(ej), y(ej) — координаты трассировочного элемента ej.
Fig. 7. Congestion heatmap for routing resources for example circuit vga_enh_top [31], x(ej), y(ej) are coordinates of routing element ej.

Алгоритм Pathfinder и его модификации не требуют сведения трассировки межсоединений к задаче целочисленного линейного программирования с множеством ограничений для соблюдения требования бесконфликтной разводки. Необходимый эффект сходимости итерационного процесса к корректному трассировочному решению достигается учётом истории перегруженности трассировочных ресурсов. Графически перегруженность можно представить в виде тепловой диаграммы (рис. 7), где оси абсцисс и ординат образуют координатную сетку, в узлах которой расположены трассировочные элементы кристалла

ПЛИС, а по оси аппликат отмечается величина нормализованной перегруженности, которая рассчитывается как

$$cong(e_{ij}) = \frac{nets(e_{ij})}{max(nets(e_{ij}))},$$

где $nets(e_{ij})$ – количество проектных межсоединений, в разводке которых участвует перегруженный ресурс.

Для наиболее эффективной работы модуля трассировки в составе разрабатываемой САПР X-CAD произведено внесение модификаций в классическую версию алгоритма Pathfinder. В первую очередь, они направлены на детальный учёт особенностей коммутационных ресурсов новейших отечественных ПЛИС и реконфигурируемых систем на кристалле для достижения наилучших выходных характеристик проектируемых схем [35]. Например, к таким особенностям нужно отнести неклассическую структуру программируемых связей, которая вызвана технологическими ограничениями проектирования. Схемотехнически она реализована посредством разреженности коммутационной сети и внедрения широкого набора коммутационных элементов с комплексным управлением проводимостью. Во вторую очередь, модификации классического Pathfinder позволяют значительно ускорить [36] прохождение этапа трассировки без значительного ухудшения итоговых результатов.

4. Статический временной анализ

Инструменты статического временного анализа играют ключевую роль в оценке результатов топологического синтеза, позволяя проверить соответствие временных и мощностных характеристик схемы заданным ограничениям. В САПР X-CAD для выполнения статического временного анализа было интегрировано открытое программное обеспечение OpenSTA [37], имеющее широкий спектр возможностей. В отличие от аналогичных программных продуктов, таких как iSTA [38] или OpenTimer [39], данная программа включает в себя многие возможности, позволяющие качественно и эффективно проводить статический временной анализ. Основные достоинства OpenSTA заключаются в способности работать с иерархическим списком соединений и поддержке учёта вариаций на кристалле (от англ. On-Chip Variation, OCV). Кроме того, программа OpenSTA проверяет, что тактовый сигнал распространяется через блоки стробирования синхросигнала (англ. clock gating), и выполняет анализ мощности, что дает дополнительные возможности при работе с энергоэффективными схемами. Немаловажным является тот факт, что OpenSTA обладает широким набором команд управления, распознаваемых его собственным интерпретатором команд на языке Tcl. Благодаря этому, инструмент статического временного анализа может быть легко интегрирован в сторонние программные продукты, сохраняя свою функциональность.

Алгоритм работы с OpenSTA в составе САПР X-CAD состоит из нескольких этапов. Сначала задаются подключаемые Verilog файлы проекта и файлы библиотек в формате Liberty, после чего происходит сборка проекта. Далее пользователем задаются проектные ограничения и указывается требуемый формат результатов анализа. Так как работа с программой OpenSTA осуществляется посредством её собственных команд, то для взаимодействия с САПР X-CAD используется управляющий сценарий на языке Tcl, который на основе заданных временных ограничений создаёт командный файл, передаваемый программе OpenSTA. При этом ограничения можно задать как с помощью параметров консоли, так и с помощью сформированных файлов в формате Synopsys Design Constraints (SDC).

В маршруте проектирования ИС на основе ПЛИС, реализованном в САПР X-CAD, временной анализ проводится два раза: после логического и после топологического синтеза. Анализ, выполняющийся первым, называется предварительным. Анализ после размещения и трассировки называется итоговым. В ходе предварительного анализа анализируется список соединений после этапа технологического отображения. В процессе итогового анализа

список соединений дополняется коммутационными элементами и ёмкостями межсоединений, полученными в результате трассировки межсоединений.

Для формирования ёмкостей для каждого кристалла ПЛИС проводится паразитная экстракция и формируется специальный файл, в котором каждому межсоединению поставлена в соответствие паразитная ёмкость. При работе над проектом в САПР после трассировки генерируется файл, содержащий успешно трассированные цепи и связанные с ними ёмкости. Данный файл в ходе статического временного анализа передается программе OpenSTA. Процесс генерации и загрузки этого файла в программу полностью автоматизирован и не требует вмешательства пользователя.

Важным преимуществом OpenSTA является возможность выполнить статический временной анализ иерархического списка соединений. Данная функциональность необходима, поскольку в ходе технологического отображения проектируемой схемы, комбинационные и последовательностные элементы могут быть объединены в ПЛБ ПЛИС. Список соединений, содержащий подобные элементы, не может напрямую использоваться для статического временного анализа, так как они отсутствуют в библиотеках ПЛИС в явном виде. Однако, используя возможность OpenSTA работать с иерархическим представлением схемы и используя дополнительную библиотеку объединенных элементов становится возможным провести его временной анализ без генерации дополнительных списков соединений.

5. Заключение

Создание доверенных программных средств проектирования микроэлектроники позволяет избежать зависимости от зарубежных производителей коммерческих систем и способствует появлению современных решений в области формирования отечественной электронной компонентной базы. В данной работе рассмотрены основные особенности разработки отечественной доверенной САПР для проектирования в базисе гетерогенных ПЛИС. В частности, дан обзор основных используемых программных средств и алгоритмов, применяемых для построения доверенной САПР X-CAD. Рассмотрен полный маршрут проектирования ИС в базисе гетерогенных ПЛИС, включая логический синтез и технологическое отображение, этапы топологического синтеза и статического временного анализа. Показаны основные преимущества применяемых методов и подходов.

В процессе разработки доверенных программных средств и систем ИППМ РАН придерживается подхода, состоящего в сочетанном применении открытых программных средств проектирования и использовании собственных оригинальных разработок в области создания САПР ИС. Он не лишен недостатков, в частности, проявляющихся в сложности получения технической поддержки для открытых программных решений. Однако такая стратегия позволяет использовать наилучшие актуальные наработки научного сообщества в области САПР ИС, ускоряя процесс разработки и перенося наибольшую часть трудозатрат на разработку собственных решений для критически важных компонентов системы.

Список литературы / References

[1]. Wolf C., Glaser J. Yosys - A Free Verilog Synthesis Suite. [Электронный ресурс] // Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip). 2013. URL: https://yosyshq.net/yosys/files/yosys-austrochip2013.pdf (дата обращения: 25.10.2023).

[2]. Yosys Open SYnthesis Suite. Frequently Asked Questions [Электронный ресурс] // URL: https://yosyshq.net/yosys/faq.html (дата обращения: 26.10.2023).

[3]. 5400TC015 Программируемая логическая интегральная схема (ПЛИС) [Электронный ресурс] // URL: https://dcsoyuz.ru/products/pais/art/1727 (дата обращения: 25.10.2023).

[4]. ПЛИС емкостью 145 тыс. системных вентилей 5510XC3AT [Электронный ресурс] // URL: https://mikron.ru/products/high-rel-ic/programmiruemaya-logika-fpga/fpga/product/5510hs3at/ (дата обращения: 25.10.2023).

[5]. Tiunov I.V., Lipatov I.A., Zheleznikov D.A. Digital Circuits Resynthesis Approach for FPGAs Based on Logic Cell with Built-In Flip-Flop, Problems of advanced micro- and nanoelectronic systems development, 2019, pp. 33-36. DOI 10.31114/2078-7707-2019-3-33-36.

[6]. Тиунов И.В. Методы ресинтеза схем для ПЛИС на основе ячеек с разделенными выходами и обратной связью // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2020. №2. С. 50-56. DOI: 10.31114/2078-7707-2020-2-50-56. / Tiunov I.V. Resynthesis methods for FPGAs based on cells with separated outputs and built-in feedback // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2020. Issue 2. P. 50-56 (in Russian). DOI:10.31114/2078-7707-2020-2-50-56.

[7]. Хватов В. М., Гаврилов С. В. Формирование библиотек СФ-блоков в маршруте проектирования пользовательских схем на ПЛИС и РСнК // Известия высших учебных заведений. Электроника. – 2021. – Т. 26, № 5. – С. 387-398. DOI: 10.24151/1561-5405-2021-26-5-387-398. / Khvatov V.M., Gavrilov S.V. Complex functional block libraries formation in the design flow of user circuits on FPGA and RSoC. Proc. Univ. Electronics, 2021, vol. 26, no. 5, pp. 387–398 (in Russian). DOI: 10.24151/1561-5405-2021-26-5-387-398.

[8]. Khvatov V. M., Zheleznikov D. A., Gavrilov S. V. Analysis of the Programmable Soft IP-cores Implementation for FPGAs," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021, pp. 2681-2685.

[9]. Shah D., Hung E., Wolf C. et. al. Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs. 2019. pp. 1-40.

[10]. Yosys Manual, techmap - generic technology mapper. [Электронный ресурс] URL: https://yosyshq.readthedocs.io/projects/yosys/en/latest/cmd/techmap.html (дата обращения: 25.10.2023).

[11]. Хватов В. М., Гарбулина Т. В., Лялинская О. В. Методы формирования и верификации библиотек стандартных элементов в составе маршрута проектирования ИС на базе ПЛИС отечественного производства // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). – 2018. – № 1. – С. 57-62. DOI:10.31114/2078-7707-2018-1-57-62. / Khvatov V.M., Garbulina T.V., Lyalinskaya O.V. Formation and Verification of Standard Element Libraries in the Design Flow for the Domestic FPGAs // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2018. Issue 1. P. 57-62 (in Russian). DOI:10.31114/2078-7707-2018-1-57-62.

[12]. Hauck, S. Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation / S. Hauck, A Denon. –, 2008. – 944 p.

[13]. Фролова П.И., Чочаев Р., Иванова Г.А., Гаврилов С.В. Алгоритм размещения с оптимизацией быстродействия на основе матриц задержек для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2020. Выпуск 1. С. 2-7. DOI:10.31114/2078-7707-2020-1-2-7. / Frolova P.I., Chochaev R., Ivanova G.A., Gavrilov S.V. Timing-driven placement algorithm based on delay matrix model for reconfigurable system-on-chip // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2020. Issue 1. P. 2-7 (in Russian). DOI:10.31114/2078-7707-2020-1-2-7.

[14]. Miettinen P., Honkala M., Roos J. Using METIS and hMETIS Algorithms in Circuit Partitioning // Report CT49, Circuit Theory Laboratory, Helsinki University of Technology, 2006.

[15]. Devine K. D., Boman E.G., Riesen L.A., Catalyurek U.V., Chevalier C. Getting started with zoltan: A short tutorial. // In Combinatorial Scientific Computing №09061 in Dagstuhl Seminar Proceedings, 2009, p. 10.

[16]. Çatalyürek Ü., Aykanat C. PaToH (Partitioning Tool for Hypergraphs) // Encyclopedia of Parallel Computing, 2020. Springer US, Boston, MA, pp. 1479-1487. DOI: 10.1007/978-0-387-09766-4_93

[17]. Karypis G., Kumarh V. hMETIS∗ A Hypergraph Partitioning Package Version 1.5.3 [Электронный ресурс] // Minnesota. 1998. URL: https://course.ece.cmu.edu/~ee760/760docs/hMetisManual.pdf (дата обращения: 13.10.2023).

[18]. The first version of TritonPart: программа. / ABKGroup. URL: https://github.com/ABKGroup/TritonPart (дата обращения: 17.11.2023)

[19]. FREIGHT: Fast stREamInG Hypergraph parTitioning: программа. / KaHIP. Лицензия: MIT. URL: https://github.com/KaHIP/FREIGHT (дата обращения: 17.11.2023)

[20]. Mt-KaHyPar - Multi-Threaded Karlsruhe Graph and Hypergraph Partitioner: программа. / KaHyPar. Лицензия: MIT. URL: https://github.com/kahypar/mt-kahypar (дата обращения: 17.11.2023)

[21]. Schlag S., Heuer T., Gottesbüren L., et. al. High-Quality Hypergraph Partitioning // ACM J. Exp. Algorithmics Just Accepted, Association for Computing Machinery, New York, 2022. DOI: 10.1145/3529090.

[22]. Фролова П.И., Хватов В.М., Чочаев Р. Сравнительный анализ методов кластеризации и размещения схем для реконфигурируемых систем на кристалле // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2022. Выпуск 4. С. 63-70. doi:10.31114/2078-7707-2022-4-63-70. / Frolova P.I., Khvatov V.M., Chochaev R. Comparative Analysis of Clustering and Placement Methods for Reconfigurable System-on-Chips // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2022. Issue 4. P. 63-70. (in Russian). DOI:10.31114/2078-7707-2022-4-63-70.

[23]. Lin Z., Xie Y., Qian G., et. al. Late Breaking Results: An Analytical Timing-Driven Placer for Heterogeneous FPGAs* // 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020, pp. 1-2, DOI: 10.1109/DAC18072.2020.9218699.

[24]. Фролова П.И., Чочаев Р. Разработка и сравнительный анализ методов начального размещения на ПЛИС // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2021. Выпуск 3. С. 57-64. doi:10.31114/2078-7707-2021-3-57-64. / Frolova P.I., Chochaev R. Development and Comparative Analysis of Initial Placement Methods for FPGA // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2021. Issue 3. P. 57-64 (in Russian). DOI:10.31114/2078-7707-2021-3-57-64.

[25]. Eisenmann H., Johannes F. M. Generic global placement and floorplanning // Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175), 1998, pp. 269-274, DOI: 10.1145/277044.277119.

[26]. Гаврилов С.В., Железников Д.А., Чочаев Р.Ж. Разработка и сравнительный анализ методов решения задачи размещения для реконфигурируемых систем на кристалле // Изв. вузов. Электроника. 2020. Т. 25. № 1. С. 48–57. DOI: 10.24151/1561-5405-2020-25-1-48-57. / Gavrilov S.V., Zheleznikov D.A., Chochaev R.Z. Development and comparative analysis of placement methods for reconfigurable systems-on-a-chip. Proc. Univ. Electronics, 2020, vol. 25, no. 1, pp. 48–57 (in Russian). DOI: 10.24151/1561-5405-2020-25-1-48-57.

[27]. McMurchie L, Ebeling C. PathFinder: A negotiation-based performance-driven router for FPGAs. Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays, 1995, pp. 111-117.

[28]. D. Wang, J. Feng, K. Liu, W. Zhou, X. Hao and X. Zhang, "A Fast FPGA Connection Router Using Prerouting-Based Parallel Local Routing Algorithm," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 42, no. 11, pp. 3868-3880, Nov. 2023, DOI: 10.1109/TCAD.2023.3274950.

[29]. Mai, J., Wang, J., Di, Z., et.al. OpenPARF: An Open-Source Placement and Routing Framework for Large-Scale Heterogeneous FPGAs with Deep Learning Toolkit [Электронный ресурс] // arXiv preprint arXiv:2306.16665. 2023. URL: https://arxiv.org/pdf/2306.16665.pdf (Дата обращения: 20.10.2023).

[30]. OpenROAD's unified application implementing an RTL-to-GDS Flow [Электронный ресурс] // URL: https://github.com/The-OpenROAD-Project/OpenROAD (дата обращения: 30.10.2023).

[31]. Verilog to Routing – Open Source CAD Flow for FPGA Research: программа / VTR Development Team. Лицензия: MIT. URL: https://github.com/verilog-to-routing/vtr-verilog-to-routing (дата обращения: 17.11.2023)

[32]. F4PGA: официальный сайт. URL: https://f4pga.org/ (дата обращения: 17.11.2023)

[33]. Заплетина М. А. Решение задачи трассировки на ПЛИС с применением модели расширенного смешанного графа коммутационных ресурсов // Изв. вузов. Электроника. 2022. Т. 27. № 6. С. 774–786. DOI: 10.24151/1561-5405-2022-27-6-774-78. / Zapletina M. A. Solving the FPGA routing problem using the model of an extended mixed routing graph. Proc. Univ. Electronics, 2022, vol. 27, no. 6, pp. 774–786 (in Russian). DOI:10.24151/1561-5405-2022-27-6-774-786.

[34]. The OpenCores VGA/LCD Controller [Электронный ресурс] // URL: https://opencores.org/projects/vga_lcd (дата обращения: 30.10.2023).

[35]. Zapletina M. A., Gavrilov S. V. Pathfinder Algorithm Modification for FPGA Routing Stage. Russian Microelectronics, 51(7), pp. 573-578. DOI: 10.1134/S1063739722070125.

[36]. Zapletina M. A., Zheleznikov D. A., Gavrilov S. V. Improving Pathfinder Algorithm Perfomance for FPGA Routing // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021, pp. 2054-2057, DOI: 10.1109/ElConRus51938.2021.9396608.

[37]. Ajayi T., Blaauw D., Chan T.-B., et. al. OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain // Proc. Government Microcircuit Applications and Critical Technology Conference, 2019, pp. 1105-1110.

[38]. Li X., Tao S., Huang Z., et al. iEDA: An Open-Source Intelligent Physical Implementation Toolkit and Library [Электронный ресурс] // arXiv preprint arXiv:2308.01857. 2023. URL: https://arxiv.org/pdf/2308.01857.pdf (дата обращения: 10.10.2023).

[39]. Huang T. W., Wong M. D. F. OpenTimer: A high-performance timing analysis tool // 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). – IEEE, 2015. pp. 895-902. DOI: 10.1109/ICCAD.2015.7372666.

Информация об авторах / Information about authors

Сергей Витальевич ГАВРИЛОВ – доктор технических наук, профессор, директор Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС.

Sergey Vitalievich GAVRILOV – Dr. Sci. (Tech.), Professor, CEO of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation,VLSI computer-aided design.

Даниил Александрович ЖЕЛЕЗНИКОВ – кандидат технических наук, старший научный сотрудник отдела САПР ИС Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС.

Daniil Aleksandrovich ZHELEZNIKOV – Cand. Sci. (Tech.), Senior Research Scientist of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation, VLSI computer-aided design.

Мария Андреевна ЗАПЛЕТИНА – кандидат технических наук, научный сотрудник отдела САПР ИС Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, РСнК.

Mariia Andreevna ZAPLETINA – Cand. Sci. (Tech.), Research Scientist of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation, VLSI computer-aided design, FPGA, SoC FPGA.

Иван Викторович ТИУНОВ – младший научный сотрудник отдела САПР ИС Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, логический синтез, графические интерфейсы.

Ivan Victorovich TIUNOV – Junior Research Scientist of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation, VLSI computer-aided design, FPGA, logic synthesis, GUI.

Василий Михайлович ХВАТОВ – научный сотрудник отдела САПР ИС Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, РСнК.

Vasilii Mikhailovich KHVATOV – Research Scientist of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation, VLSI computer-aided design, FPGA, SoC FPGA.

Рустам Жамболатович ЧОЧАЕВ – инженер-исследователь отдела САПР ИС Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, РСнК.

Rustam Zhambolatovich CHOCHAEV – Research Engineer of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation, VLSI computer-aided design, FPGA, SoC FPGA.

Дмитрий Борисович ШОКАРЕВ – инженер-исследователь отдела САПР ИС Института проблем проектирования в микроэлектронике РАН. Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, РСнК.

Dmitry Borisovich SHOKAREV – Research Engineer of Institute for Design Problems in Microelectronics of RAS. Research interests: electronic design automation, VLSI computer-aided design, FPGA, SoC FPGA.

DOI: 10.15514/ISPRAS-2023-35(5)-9

Быстрый анализ статического IR drop эффекта на базе методов машинного обучения

[1] *Р.А. Соловьёв*, ORCID: 0000-0003-0312-452X <*roman.solovyev.zf@gmail.com* >
[1] *Д.В. Тельпухов*, ORCID: 0000-0001-9551-0748 <*telpukhov@ippm.ru* >
[2] *Е.Д. Демидов*, ORCID: 0009-0009-0253-0228 <*yevgeniy_demidov_1999@mail.ru*>
[1] *И.И. Шафеев*, ORCID: 0009-0005-2323-3725 <*ilya.tanc@gmail.com*>

[1] *Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН), 124365, Россия, г. Москва, г. Зеленоград, ул. Советская, д.3.*
[2] *Национальный исследовательский университет «МИЭТ», 124498, Россия, г. Москва, г. Зеленоград, пл. Шокина, д. 1.*

Аннотация. В статье рассматривается решение проблемы быстрого проведения статического анализа падения напряжения с использованием нейронной сети. Рассматривается методика генерации базы данных необходимой для обучения ML-модели. Описывается методика обучения ML-модели для анализа статического IR drop эффекта. Описывается алгоритм получения входных данные для обучения нейронной сети из SPICE представления. Предложенное решение задачи попало в ТОП3 конкурса ICCAD Contest 2023.

Ключевые слова: IR drop эффект; падение напряжения; ML-модель; нейронная сеть; машинное обучение; база данных.

Для цитирования: Соловьёв Р.А., Тельпухов Д.В., Демидов Е.Д., Шафеев И.И. Быстрый анализ статического IR drop эффекта на базе методов машинного обучения. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 127–144. DOI: 10.15514/ISPRAS–2023–35(5)–9.

Fast Analysis of Static IR Drop Effect Based on Machine Learning Methods

[1] *R.A. Solovyev*, ORCID: 0000-0003-0312-452X <*roman.solovyev.zf@gmail.com*>
[1] *D.V. Telpukhov*, ORCID: 0000-0001-9551-0748 <*telpukhov@ippm.ru* >
[2] *E.D. Demidov*, ORCID: 0009-0009-0253-0228 <*yevgeniy_demidov_1999@mail.ru*>
[1] *I.I. Shafeev*, ORCID: 0009-0005-2323-3725 <*ilya.tanc@gmail.com*>

[1] *Institute of Design Problems in Microelectronics, RAS (IPPM RAS), 3, Sovetskaya st., Moscow, Zelenograd, 124365, Russia.*
[2] *National Research University of Electronic Technology (MIET), 1, Shokina Square, Moscow, Zelenograd, 124498, Russia.*

Abstract. As part of the ICCAD Contest 2023 (Problem C) competition, the paper describes a methodology for applying ML models to perform static IR drop analysis. Methods for obtaining a database for training a neural network to solve this problem are given. We consider a technique for training an ML model to analyze the static IR-drop effect. The generation of input data for training a neural network from SPICE netlists is also discussed in this paper. This solution is ranked in the TOP 3 at the ICCAD Contest 2023 competition.

Keywords: IR drop analysis; ML-model; neural network; machine learning; database.

For citation: Solovyev R.A., Telpukhov D.V., Demidov E.D., Shafeev I.I. Fast analysis of static IR drop effect based on machine learning methods. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 127-144 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-9.

1. Введение

С уменьшением технологических норм проектирования увеличивается влияние различных негативных эффектов на разработанные микроэлектронные схемы. Одним из таких эффектов является IR drop (падение напряжения на шинах питания). IR drop эффект – это эффект, характеризующийся возникновением разности электрических потенциалов между двумя контактами проводника во время прохождения тока [1]. Эта разница потенциалов возникает за счёт падения напряжения при прохождении тока через участок металлической шины, имеющей сопротивление. Влияние на схему, которое оказывается IR drop эффектом, может привести к снижению производительности схемы из-за увеличения времени задержки переключения стандартных ячеек. Эффект падения напряжения может привести к функциональному сбою из-за нарушений синхронизации схемы, а именно времени установки и удержания [2].

В современном маршруте проектирования выделяют два типа IR drop эффекта: статический и динамический [3]. Статический IR drop представляет собой среднее падение напряжения в схеме, характеризующееся средним значением протекающего тока, который зависит от величины периода работы схемы. Основной вклад в статический IR drop вносит ток утечки в статическом состоянии ячеек. Динамическое падение напряжения по большей мере характеризуется высокой активностью переключения транзисторов и в меньшей степени зависит от величины периода.

Для упрощения анализа схемы используется метод статического анализа IR drop эффекта, который учитывает среднее количество переключений транзисторов за период тактового сигнала. Такой метод является менее точным, чем динамический, но при этом значительно сокращается время проведения анализа. Для проведения статического анализа падения напряжения топология схемы представляется в виде упрощённой эквивалентной модели схемы (рис. 1), которая включает в себя представление шин питания и земли в виде сетки резисторов, а также представление стандартных ячеек в виде источников постоянного тока.

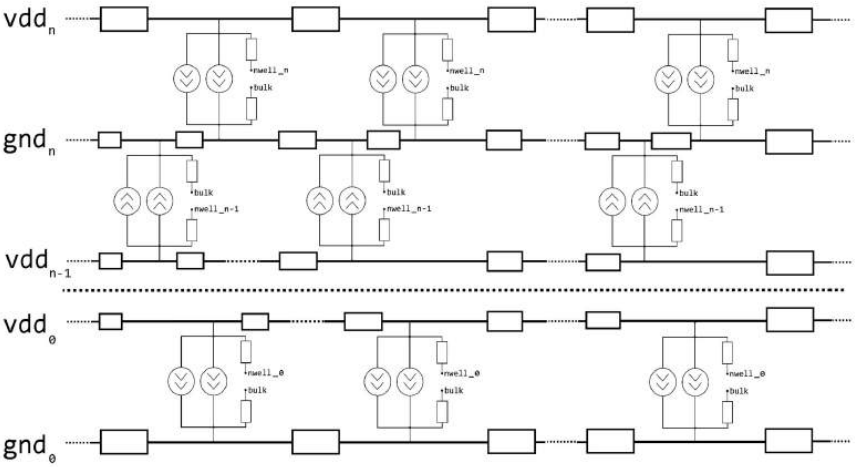


Рис. 1. Эквивалентная модель схемы.
Fig. 1. Equivalent circuit model.

Основные этапы построения эквивалентной модели и выполнения статического IR drop анализа:

- получение сетки резисторов, эквивалентной металлическим соединениям в схеме;
- вычисление средних значений потребляемой мощности каждой стандартной ячейки для получения источников постоянного тока и составления эквивалентной модели схемы;
- вычисление значений тока на каждом участке цепи путем решения системы линейных уравнений;
- вычисление значений напряжения на каждом участке цепи:

$$V = I * R$$

Целью моделирования эквивалентной модели схемы является вычисление напряжений во всех узлах сетки шин питания. Традиционно к решению этой задачи подходят с использованием метода решения системы линейных уравнений вида:

$$G * V = J, \text{ где}$$

G – матрица проводимостей;
V – вычисляемое значение напряжений в каждом узле;
J – матрица токов.

Проблема подобного подхода заключается во времени проведения расчётов, которое стремительно возрастает с увеличением узлов сетки питания в схеме. Одним из способов обойти вычислительную проблему при анализе статического IR drop эффекта является использование методов машинного обучения.

Ранее модели машинного обучения уже пытались применить в качестве средства для статического анализа падения напряжения [4, 5, 6, 7, 8]. В процессе исследования ранее предлагалось решение, определяющее статическое падение напряжение на всём кристалле на основе карты распределения мощности по сетке межсоединений и сетке шин питания [4]. Была разработана ML-модель, позволяющая быстро исследовать анализ падения напряжения, в несколько раз уменьшая время анализа по сравнению с коммерческими САПР [6]. Серьезной проблемой применения моделей машинного обучения является генерация базы данных для её обучения. Для полноценного обучения ML-модели необходимо анализировать все возможные варианты исследуемых схем. Отличия в исследуемых схемах могут начинаться от технологии, на которой они проектируются, и заканчиваться размерами и вариантами построения резистивной сети питания (power delivery network (PDN)).

В данной работе рассматривается:

1) архитектура и методика обучения нейронной сети;
2) методика сбора обучающих данных для увеличения датасета;
3) способ генерации двумерных данных из начального SPICE (Simulation Program with Integrated Circuit Emphasis) описания для шин питания;
4) сравнение скорости и точности работы предложенного метода в сравнении со SPICE моделированием.

Предложенное решение попало в ТОП3 конкурса ICCAD Contest 2023 (Problem C).

2. Конкурс ICCAD Contest 2023 (Problem C)

В 2023 году в рамках конференции ICCAD [9] проводился конкурс ICCAD Contest 2023 [10]. Конкурс проводится ежегодно на протяжении многих лет и на нем предлагаются для решения современные проблемы микроэлектроники и анализируются предлагаемые методы их решения. В качестве одной из трех задач этого года организаторы предложили для решения задачу по анализу статического эффекта падения напряжения на базе методов машинного обучения.

Глобальной задачей была разработка модели машинного обучения, которая позволяет за небольшое время определять статический IR drop эффект по всей площади схемы. Входными данными для обучения ML-модели выступают четыре матрицы для каждой схемы: матрица

источников тока, матрица эффективных расстояний источников напряжения, матрица плотности резистивной сетки и матрица значений IR drop. Подробное описание каждой из матриц будет рассмотрено далее.

В качестве обучающих данных организаторами конкурса были даны 10 реальных схем и 100 сгенерированных схем, информация о каждой из которых была представлена в виде четырех файлов: current_map.csv, eff_dist_map.csv, pdn_density_map.csv, ir_drop_map.csv. Также для каждой схемы было дано описание сети питания в SPICE формате.

В ходе предварительного исследования было обнаружено, что такого количества схем недостаточно для полноценного обучения ML-модели, поэтому в рамках работы был реализован маршрут для генерации дополнительных данных для обучения, как искусственно сгенерированных из представленных десяти схем, так и синтезированных из описаний RTL (Register Transfer Level) с помощью автоматизированного маршрута OpenROAD.

3. Предсказание результатов IR drop анализа с помощью методов машинного обучения

Распределение значений статического падения напряжения на цепях питания в микросхеме зависит от следующих трех факторов:

- расположение/распределение всех источников напряжения (силовых площадок) в схеме;
- топология сети земли и питания, значения сопротивлений каждого резистора (перехода/металлического слоя) в сети;
- распределение источников тока в схеме.

Каждый из обозначенных факторов связан с топологией и, следовательно, может быть представлен в виде изображения. Финальный результат IR drop анализа также является изображением, что позволяет использовать богатый опыт разработки различных моделей машинного обучения и нейронных сетей, работающих с изображениями. В работе был предложен маршрут для обучения и применения (инференса) моделей для анализа падения напряжения на цепях земли и питания (рис. 2).

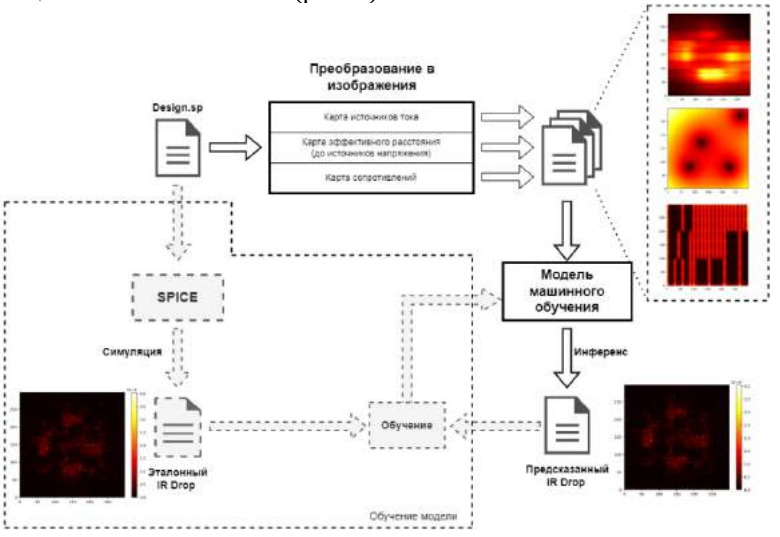


Рис. 2. Предложенный маршрут обучения и применения (инференса) моделей для предсказания результатов IR Drop анализа.

Fig. 2. A proposed route for training and applying (inferential) models for predicting IR Drop analysis results.

На вход разработанного маршрута поступает файл с описанием сети питания, который представляет собой SPICE файл с источниками внешнего напряжения, источниками тока, имитирующими стандартные ячейки и резисторами, которые определяют значения сопротивлений металлических проводников. В названиях цепей закодированы их координаты в топологии схемы, что позволяет полноценно воссоздать топологию сети питания (рис. 3).

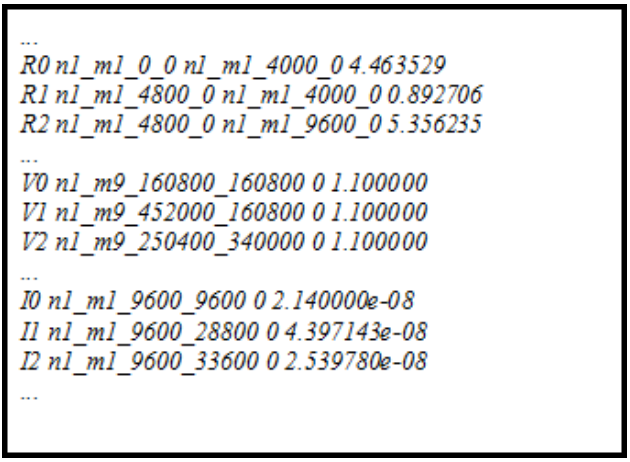


Рис. 3. Пример SPICE файла с описанием сети питания.
Fig. 3. Example SPICE file with power delivery network description.

Подаваемый на вход файл поступает на конвертер, который должен сформировать изображения, отражающие все три фактора, обозначенные в начале данного раздела. После этого, изображения поступают на вход предсказательной модели, которая выдаёт карту падений напряжения на цепях питания. Обучение модели сводится к тому, чтобы сгенерировать датасет, в котором в качестве эталонных значений карт IR drop будут выступать результаты SPICE симуляции большого набора схем, после чего модель будет учиться предсказывать значения IR drop во всех узлах схемы (рис. 2).

Подобный подход, связанный с конвертацией сети питания в изображения, уже был предложен в научных работах [4-8]. Например, в [4] три вышеуказанные характеристики представлены в виде трех распределений, где распределение источников тока моделируется как карта токов, топология металлических проводников моделируется как функция плотности (расстояние между полосами питания на единицу площади) в различных регионах, а распределение источников напряжения моделируется как карта эффективных расстояний, которая представляет собой расстояние от каждого узла схемы до узла с источником напряжения. В статье предлагалось использовать свёрточные нейронные сети и UNet-сети [11], позволяющие выполнять предсказание IR drop с помощью модели, обученной на разнообразном наборе таких данных (три входа и один выход).

В работе предлагается схожий подход, отличающийся архитектурой нейронной сети, преобразователями SPICE файла в изображения, и некоторыми аспектами постобработки результатов нейронной сети. В следующем разделе описаны принципы генерации изображений для обучения и инференса нейронной сети из SPICE файла, описывающего сеть питания чипа.

4. Конвертация SPICE описания сети питания в набор изображений

Все три предоставленных входных изображения для каждой схемы, как и выходной IR drop, имеют одинаковое разрешение, которое пропорционально реальному размеру кристалла. В конкурсе, размер изображения в пикселях равен размеру схемы в микрометрах. Один пиксель

отражает общую площадь кристалла в один квадратный микрометр. Однако, в предоставленном SPICE файле «рис. 3», содержащем информацию о сопротивлениях, источниках напряжения и источниках тока, координаты цепей масштабированы с использованием коэффициента 2000. Этот коэффициент необходимо учитывать при конвертации. Иными словами, один пиксель итоговых изображений содержит 2000 условных единиц по оси x и 2000 условных единиц по оси y.

4.1. Получение карты эффективного расстояния

Способ получения карты эффективного расстояния заключается в отображении реальной матрицы с источниками напряжения в матрицу, которая не только будет сжата по размерам, но также будет показывать, насколько далеко или близко находится каждая ячейка матрицы, это позволит улучшить понимание влияние источников на все области PDN. Сделать это можно, используя следующий алгоритм:

1. Изначально выбирается шаг отображения по осям X, Y – $stepX$ и $stepY$ соответственно. Зная размер исходной реальной матрицы, можно вычислить размеры сжатой матрицы:

$$w' = \left(\frac{w}{stepX}\right) + 1, \qquad h' = \left(\frac{h}{stepY}\right) + 1$$

2. Выполняется проход по всем ячейкам матрицы размера $(w',\ h')$. Для каждой ячейки вычисляется обратная сумма евклидового расстояния до всех источников напряжения:

$$S = \sum_{0}^{n} \frac{1}{\sqrt{(x_n - x)^2 + (y_n - y)^2}}$$, где

n – количество источников напряжения;

x, y – координаты рассматриваемой ячейки;

x_n, y_n – координаты каждого источника напряжения в схеме.

3. После вычисления суммы, каждой ячейке присваивается значение эффективного расстояния:

$$cell = S * \frac{(stepX + stepY)}{2},$$

следовательно, если точка близко к источнику напряжения, ее значение в матрице будет высоким, и наоборот.

4.2. Получение карты сопротивлений сети питания

В рамках конкурса для представления данных о сети земли и питания было предложено использовать карту плотности PDN, которая отражает топологию сети питания и значения сопротивления каждого резистора в этой сети. Она моделируется как функция плотности (расстояние между полосами питания на единицу площади) сетки питания в разных областях чипа.

Однако в нашей работе было принято решение использовать саму топологию сети питания. Получаемая карта непосредственно отражает распределение сопротивлений металлических проводников цепи питания по площади чипа. Это позволяет более точно моделировать IR drop и использовать модель на очень нерегулярных сетях питания. Отличие данной карты от карты плотности PDN изображено на рис. 4.

Основные шаги получения карты сопротивлений PDN:

1. Извлечение координат $x1, y1, x2, y2$ узлов и значений сопротивления для всех резисторов из SPICE файла с описанием схемы.

2. Создание пустой матрицы размером $x\_width\ X\ y\_width$, где x_width и y_width вычисляются исходя из размеров ядра и шага сетки $stepX$ и $stepY$, равного 2000.
3. Для каждого резистора добавляется его значение в ячейки матрицы, соответствующие координатам концов резистора, деленным на шаг сетки.
4. Применение размытия по Гауссу к матрице для сглаживания изображения.

Таким образом, в результате получается карта топологии PDN, отражающая распределение резисторов по площади чипа. Эта карта используется как один из входных признаков для обучения ML-модели.

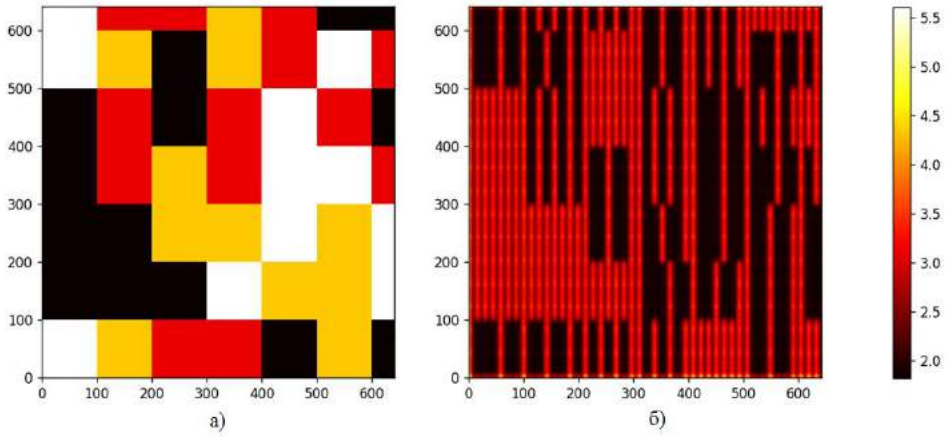


Рис. 4. Карта сети питания (карта плотности (а), карта сопротивлений (б)).
Fig. 4. Power network map (density map (a), resistance map (b)) .

4.3. Получение карты источников тока

Основная идея получения карты источников тока заключается в отображении реальной матрицы с источниками тока в некоторую сжатую версию, при этом требуется минимизировать количество возможной потерянной информации. Сделать это можно, используя следующий алгоритм:

1. Изначально выбирается шаг отображения по осям $x, y - stepX$ и $stepY$ соответственно. Зная размер исходной реальной матрицы, можно вычислить размеры сжатой матрицы:

$$w' = \left(\frac{w}{stepX}\right) + 1, \qquad h' = \left(\frac{h}{stepY}\right) + 1$$

2. Следующим шагом является обход реальной матрицы по ячейкам, которые хранят в себе информацию об источниках тока. Сначала вычисляется положение каждого источника тока в сжатой матрице:

$$x' = \left(\frac{x}{stepX}\right), \qquad y' = \left(\frac{y}{stepY}\right)$$

В случае, когда координаты источника тока в сжатом матричном пространстве соответствуют координатам ячейки, расположенной на границе этой матрицы, происходит процедура аддитивной интеграции: значение тока, генерируемое источником, суммируется с существующим значением в указанной ячейке. В другом случае, когда позиция источника тока находится внутри матрицы и не прилегает к её границам, возникает необходимость в анализе диффузии токового значения от

ячейки (x', y') к соседним ячейкам в пределах заданного диапазона в пределах одной ячейки.

3. Для определения влияния текущего источника тока на окружающие его ячейки, нужно рассмотреть области пересечения ячейки с центром в точке $(xstepX - 0.5, ystepY - 0.5)$ и размерами (1,1) с ячейками, окружающими ячейку (x', y') в радиусе одной ячейки. Значение площади пересечения ячеек затем умножается на числовое значение источника тока и складывается с пересекающейся ячейкой.
4. Полученная сжатая матрица затем проходит через фильтр Гаусса для достижения более однородного распределения значений в пустых ячейках.

5. Метрики качества

В рамках конкурса было предложено 2 метрики качества решения: MAE (Mean Absolute Error) и F1-score. Наиболее важной при оценке качества прогнозирования IR drop считалась метрика MAE, её вклад в общую метрику качества составлял 60%. Метрика F1 давала вклад 30%, а оставшиеся 10% вносила скорость вычислений. Метрика MAE оценивает общее качество решения на всей схеме. Метрика F1 – оценивает качество только на критических участках схемы с максимальным значением IR drop. Проектировщикам микросхем важно знать координаты на топологии, где падение напряжения максимально. Именно в критичных местах будет необходимо вносить изменения в структуру схемы, чтобы решить проблему большого падения напряжения. Все остальные места схемы с минимальным значением IR drop и качество решения там играют гораздо меньшую роль.

5.1. Метрика MAE

MAE или, по-другому, средняя абсолютная ошибка [12] – это среднее значение абсолютной разницы между прогнозом и фактическим значением, рассчитанное для каждого примера в наборе данных. Цель состоит в том, чтобы минимизировать метрику MAE.
Пример: возьмем матрицу 2x2 для наглядности. Предположим, что мы получили значения IR drop:

$$\begin{matrix} 2.1 & 6.4 \\ 3.8 & 0.1 \end{matrix}$$

При этом реальные значения IR drop:

$$\begin{matrix} 2.5 & 5.3 \\ 4.9 & 0.0 \end{matrix}$$

Тогда матрица MAE будет выглядеть следующим образом:

$$\begin{matrix} 0.4 & 1.1 \\ 1.1 & 0.1 \end{matrix}$$

Метрика MAE после усреднения будет равна:

$$MAE = 0.675$$

Это значение средней ошибки по всей матрице IR drop.

5.2. Метрика *F*1

*F*1 – это метрика бинарной классификации [13]. В этой задаче она использует 10 % максимального IR drop каждой схемы в качестве порога для выполнения классификации. Формулы для вычисления метрики:

$$F1 = 2 * Precision * \frac{Recall}{Precision + Recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}, \text{где}$$

TP (True positive) – количество совпадающих значений «1» между предсказанной матрицей и реальной;
FP (False positive) – количество несовпадающих значений «1» между предсказанной матрицей и реальной;
TN (True negative) – количество совпадающих значений «0» между предсказанной матрицей и реальной;
FN (False negative) – количество несовпадающих значений «0» между предсказанной матрицей и реальной.
На карте IR drop значением 1 (True) отмечаются участки, на которых IR drop выше, чем 90% от максимального для заданной схемы. Ставится задача найти такие участки как можно точнее. Чем точнее результат, тем выше будет метрика $F1$.
Рассмотрим пример из предыдущего раздела. Максимальное значение IR drop для матрицы равно 5.3. Если мы возьмем 90% от него, получим 4.77. Таким образом, все значения больше, чем 4.77 и в предсказанной матрице, и в реальной матрице станут равными 1, а остальные значения станут равными 0.
Таким образом, матрица горячих точек для предсказанной матрицы будет равна:

0 1
0 0

А для реальной матрицы:

0 1
1 0

Таким образом:

$$TP = 1,\ TN = 2,\ FP = 0,\ FN = 1$$

По формулам можно рассчитать:

$$Precision = 1 / (1 + 0) = 1,$$
$$Recall = 1 / (1 + 1) = 0.5. F1 = 2 * 1 * 0.5 / (1 + 0.5) = 0.67$$

6. Генерация датасета для обучения модели

Основная проблема, с которой сталкиваются разработчики нейросетевых моделей для решения задач в области физического синтеза – это отсутствие достаточного количества данных для обучения. Такая же проблема наблюдается и в данной задаче. Авторы конкурса предоставили в открытом доступе 100 сгенерированных схем для обучения и 10 реальных схем для отладки предлагаемого решения, а также ещё 10 реальных схем организаторы конкурса оставили закрытыми для внутреннего тестирования решений. Закрытые схемы нужны, чтобы быть уверенным, что разработанные модели не обучались на реальных схемах. Эти закрытые схемы использовались, чтобы сформировать финальную таблицу лидеров конкурса. Десяти реальных схем и ста сгенерированных схем явно недостаточно для обучения хорошей модели. Поэтому потребовалось искать и готовить дополнительные данные для обучения. Было обнаружено, что тестовые схемы были синтезированы с помощью технологии Nangate45 [14], поскольку сгенерированная сетка питания на данной технологии была схожа с предоставленными данными от организаторов. Организаторы рекомендовали использовать для обучения схемы, сгенерированные с помощью генеративной нейронной сети BeGAN [15-16]. В работе были сгенерированы новые схемы из уже существующих, а также синтезированы топологии нескольких реальных схем на

технологии Nangate45 с помощью автоматизированного маршрута физического синтеза OpenROAD [17-18].

6.1. Алгоритм получения карты падения напряжения

Результатом проведения этапа физического синтеза является топологическое представление схемы, представленное в общепринятом формате DEF (Design Exchange Format). Далее с помощью программы PDNSim, входящей в состав OpenROAD, генерировался SPICE нетлист. Моделируя нетлист с помощью SPICE симулятора, были получены значения напряжений в каждом узле резистивной сетки, которые в конечном итоге конвертировались в эталонные значения падения напряжения в каждом узле.

6.2. Алгоритм генерации новых схем из заданных

Для расширения набора данных для обучения на основе уже имеющихся схем, было использовано несколько подходов:

1. Перемещение источников тока на новые позиции.
2. Увеличение количества источников тока.
3. Изменение числового значения источников тока.

Для генерации новых схем, описанные выше подходы, применялись последовательно. Если все возможные перемещения источников тока уже были выполнены, начинается добавление новых источников. Как только все узлы на нижнем слое металлизации подключены к источникам тока, начинается изменение их значений до достижения сопоставимого с другими схемами по порядку значения падения напряжения.
Все модификации схемы происходят на основе входного SPICE описания, выходным форматом также является SPICE. В процессе генерации для расчета значений падения напряжения используются стандартные методы решения в схемотехнике – метод узловых потенциалов (МУП) [19].
Главное преимущество этого метода заключается в том, что он позволяет генерировать множество новых вариантов схем, которые также можно решить по методу узловых потенциалов, при этом, не изменяя резистивную сетку.

6.3. Получение новых тестов на реальных схемах

Для увеличения набора данных, необходимых для обучения ML-модели, также было принято решение о генерации дополнительного набора реальных схем. Разработка схем для обучения происходила на технологии Nangate45 [14].
Основными этапами в получении новых данных для обучения нейронной сети были:

1. Проведение физического синтеза схем, содержащихся в тестовом наборе для технологии Nangate45, а также схем из набора ISCAS85 [20] с помощью открытого маршрута проектирования OpenROAD.
2. Генерация множества вариантов реальных схем (функционирующих схем), на основе синтезированных, путём изменения резистивной сети питания по принципу удаления некоторых шин питания. Удаление некоторого ограниченного числа вертикальных и горизонтальных шин питания верхнего уровня в схеме не приведёт к функциональным сбоям, что позволяет использовать данную схему для обучения.
3. Генерация случайного числа источников напряжения в схеме с расположением их в допустимых координатах на топологии, что также позволяет считать новые схемы реальными.

4. Получение SPICE нетлистов для сгенерированного набора схем и их моделирование для получения тепловых карт со значениями IR drop в каждом узле схемы в виде «.csv» файла: ir_drop_map.csv.

На примере одной из схем набора ISCAS85 на рис. 5 показаны топологические представления до модернизации сетки питания и после, а на рис. 6 показаны соответствующие им карты падения напряжения.

Таким образом, в результате генерации данных были получены наборы схем, где каждая схема описывается в виде четырех «.csv» файлов, необходимых для обучения ML-модели нейронной сети.

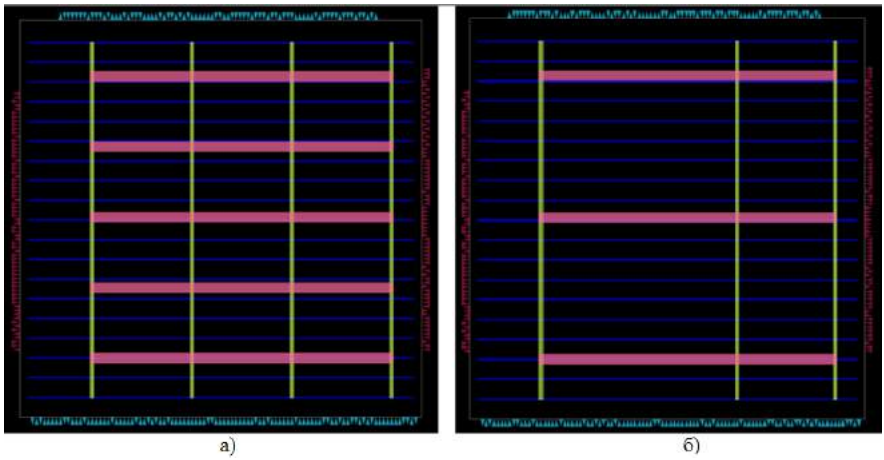


Рис. 5. Топологическое представление схемы c7552 из набора тестовых схем ISCAS85. Изображение до преобразований сети питания (а), изображение после удаления некоторых шин питания верхнего уровня (б).

Fig. 5. Topological representation of the c7552 circuit from the ISCAS85 test circuit set. Image before power supply network conversion (a), image after removing some of the top-level power supply rails(b).

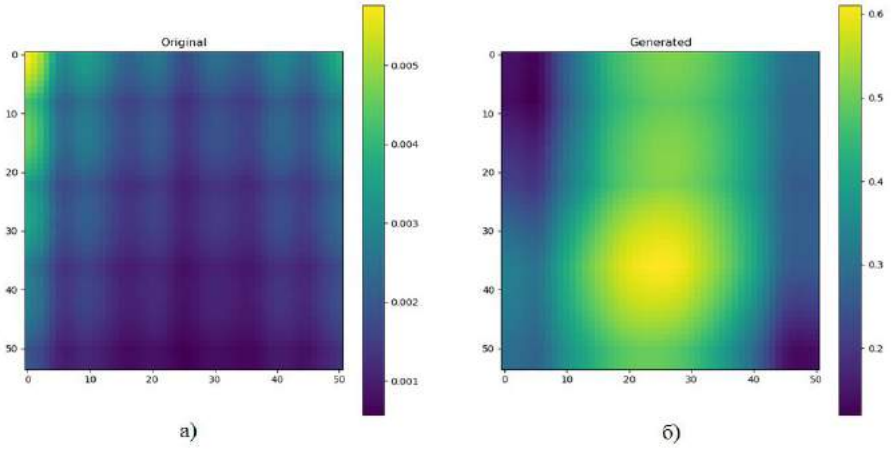


Рис. 6. IR drop для оригинальной сети питания схемы c7552 (а) и для модифицированной сети питания схемы c7552 (б).

Fig.6. IR drop for the original c7552 circuit power supply network (a) and for the modified c7552 circuit power supply network (b).

7. Нейросетевая модель

Изначально для решения задачи необходимо было определиться с архитектурой нейронной сети. Были протестированы Unet [11] и FPN [21] архитектуры с разными кодировщиками. В качестве кодировщиков использовали свёрточные нейронные сети из семейства EfficientNet [22], а также Multi-Axis Vision Transformers – MaxVit [23]. Выбор этих архитектур определялся доступностью кода для обучения, высокими метриками при решении задач классификации и предыдущим опытом успешного использования при решении задач сегментации. Сравнение на валидационном наборе данных показало, что MaxVit имеет выше метрику MAE, чем EfficientNetB5. Также требовалось выбрать размер входа для нейронной сети. Изменение размеров исходных изображений приводит к потере информации, поэтому изображения должны подаваться пиксель в пиксель. Но не обязательно подавать всё изображение, а достаточно подавать какую-то небольшую часть. Максимальный размер схемы для обучения в пикселях был $930\ X\ 930$. Современные нейронные сети для классификации изображений на базе трансформеров редко принимают на вход изображения более $512\ X\ 512$ пикселей, а типовой размер входа обычно $224\ X\ 224$ пикселей. Поэтому мы пробовали разные размеры изображений от $128\ X\ 128$ до $512\ X\ 512$. В наших экспериментах максимальное качество достигалось на больших размерах. По всей видимости, это связано с тем, что таким образом нейронная сеть имеет больше информации для предсказания значений IR drop. Если изображение меньше указанного размера, оно дополняется нулями до заданного размера.

В качестве аугментаций данных использовалось:

1) выбор случайного патча нужного размера из изображения;
2) случайный поворот на 0/90/180/270 градусов;
3) случайное отражение по вертикали/горизонтали.

На этапе использования (инференса) при анализе заданной схемы, она сначала преобразовывалась в изображение и затем методом плавающего окна анализировалась на предмет значения IR drop. Окно имело размер $512\ X\ 512$ пикселей, то есть равное входу нейронной сети, шаг можно было варьировать. При этом, чем меньше шаг, тем выше точность, но ниже скорость работы. Также для улучшения точности был использован метод Test time Augmentation [24]. Изображение вращается на 90 градусов и отражается во всех 8 возможных положениях, и результат усредняется. Можно использовать только часть положений из 8, что тоже дает возможность находить некоторый баланс между точностью и скоростью работы.

Во время обучения мы столкнулись с проблемой, что метрика F1 для тестовых примеров иногда равна нулю, но при этом значение метрики MAE низкое. Это связано с особенностью расчёта метрики F1. Если значение 90% от максимума равно N, то если мы предскажем значение меньше N, то получим False Negative, а если больше N, то True Positive и тем самым улучшим метрику F1. Однако с точки зрения MAE метрики ничего не поменяется. Для больших значений IR drop выгоднее предсказывать значение IR drop с некоторым запасом в большую сторону и избегать заниженных значений IR drop. Поэтому loss-функция была модифицирована следующим образом:

$$\text{если } pred < true, \text{то } loss = K * MAE, \text{если } pred >= true, \text{то } loss = MAE,$$

коэффициент K в нашем случае был равен 2.

Дополнительно на этапе инференса значения более чем 90% от максимального IR drop домножались на поправочный коэффициент 1.05. Это также позволило улучшить метрику F1 на валидации и при этом почти не ухудшить метрику MAE.

Изначально мы столкнулись с проблемой, что обучение не сходится, если подавать значения в абсолютных величинах. По всей видимости, это связано с тем, что погрешность вычислений начинает превышать искомые значения или же с использованием смешанной точности (mixed

precision) во время обучения. Поэтому мы сделали масштабирование значений как входных данных, так и искомого IR drop: источники тока домножались на 10^8, карта эффективных расстояний на 100 и карта плотности PDN на 100. Значения IR drop домножались на 10^5.

8. Результаты

Для тестирования скорости и точности работы были использованы данные, предоставленные организаторами ICCAD 2023. Это 10 реальных схем (см. табл. 1). Сравнивалась скорость и точность работы SPICE симулятора NGSPICE [25] и предложенной нейросетевой модели.

Табл. 1. Размеры тестовых схем
Table 1. Test circuit sizes

Название тестовой схемы	Размер, um^2	Число резисторов
testcase1	298 *X* 298	24087
testcase2	298 *X* 298	22327
testcase3	930 *X* 930	228439
testcase4	930 *X* 930	214889
testcase5	641 *X* 641	110465
testcase6	641 *X* 641	102525
testcase11	204 *X* 204	10859
testcase12	204 *X* 204	10407
testcase17	566 *X* 566	84044
testcase18	566 *X* 566	80038

Поскольку обучение нейронной сети проводилось с использованием K-Fold валидации со значением параметра k=5, то по результатам было получено 5 моделей. Были предсказаны значения по схеме Out of Fold, то есть для каждой из пяти моделей были предсказаны значения для схем, которые не принимали участия в тренировке. Результаты тестирования приведены в табл. 2. Из приведённой выше таблицы, время расчёта падения напряжения с использованием нейронной сети даже для больших схем меньше секунды. Основное время занимает преобразование SPICE описания в 2D-матрицу. Эта операция может быть оптимизирована, что даст дополнительный прирост по скорости. В данной работе оптимизация не проводилась. Результаты сравнения реальной карты IR drop с предсказанной для нескольких тестовых схем приведены на рис. 7-9.

Предложенное решение попало в тройку лидеров конкурса ICCAD Contest 2023 (Problem C).

9. Выводы

Из результатов исследования видно, что метод использования свёрточных нейросетей для предсказания IR drop имеет высокую точность (средняя абсолютная ошибка имеет значения, не превышающие $15 * 10^{-5}$). Для больших схем метод с использованием нейронных сетей в десятки раз (10-30 раз) превосходит по скорости классические методы на базе моделирования. С ростом размера схем отрыв по скорости увеличивается. Дополнительно

данный метод в будущем можно легко адаптировать для предсказания эффекта электромиграции. Однако существует ряд проблем, которые ещё нужно решить для применения метода на практике:

1. В настоящее время разработанная модель поддерживает только одну технологию – Nangate45 и скорее всего не справится с расчетом IR drop на других технологиях. Для решения этой проблемы требуется добавлять тестовые данные из других технологий. В этом случае может быть получено универсальное решение для этой задачи.
2. Требуется расширять публичный набор данных для решения этой задачи. Нужен либо набор реальных тестовых схем, либо сразу результаты физического синтеза (генерации PDN) для них в рамках различных технологий.
3. Как правило, современные СБИС состоят не только из стандартных ячеек, но и из множества макроблоков, соединяющихся между собой. В данном решении никак не учитывается возможность обработки схем, содержащих в себе макроблоки.
4. В данном решении поддерживается только один алгоритм генерации PDN, встроенный в маршрут OpenROAD.

Табл. 2. Результаты сравнения скорости NGSPICE и нейросетевой модели
Table 2. Results of speed comparison between NGSPICE and neural network model

Название тестовой схемы	**Время, с**			**Ускорение по сравнению с NGSPICE (x раз)**	**Метрика MAE * 10^{-5}**	**Метрика *F*1**
	NgSPICE	**Преобразование SPICE нетлиста в матрицы**	**Рассчёт нейро сети**			
testcase1	5.8	0.9	0.2	5.2	9.30	0.62
testcase2	4.7	0.9	0.1	4.7	14.31	0.87
testcase3	1230.3	45.4	0.8	26.6	4.84	0.75
testcase4	966.9	45.2	0.8	21	10.18	0.63
testcase5	170.3	11.9	0.4	13.8	6.10	0.81
testcase6	125.9	11.9	0.4	10.1	10.21	0.60
testcase11	1.1	0.5	0.4	1.2	10.59	0.52
testcase12	1.1	0.5	0.4	1.2	14.27	0.62
testcase17	83.6	7.3	0.6	10.5	3.41	0.69
testcase18	75.8	7.4	0.7	9.3	4.39	0.79

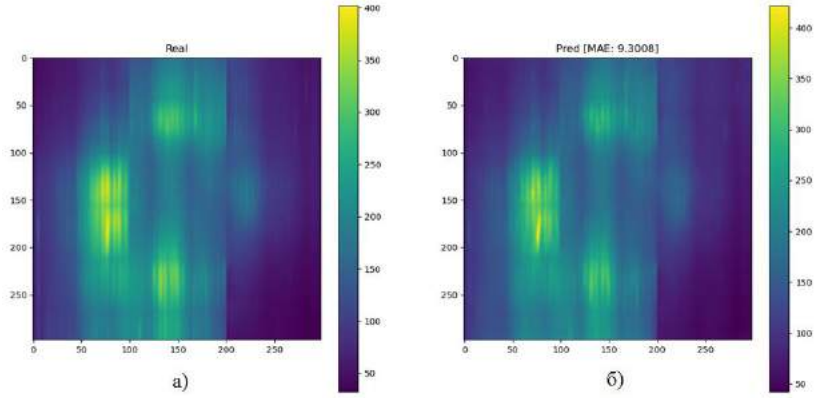


Рис. 7. Тестовая схема 1. Реальный IR drop (а), предсказанный IR drop (б) (значения IR drop нужно домножить на 10^{-5}).

Fig. 7. Testcase 1. Real IR drop (a), predicted IR drop (b) (IR drop values need to be multiplied by 10^{-5}).

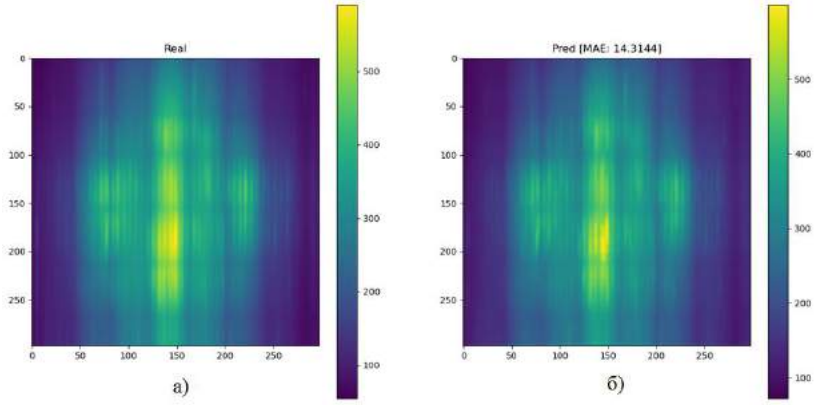


Рис. 8. Тестовая схема 2. Реальный IR drop (а), предсказанный IR drop (б) (значения IR drop нужно домножить на 10^{-5}).

Fig. 8. Testcase 2. Real IR drop (a), predicted IR drop (b) (IR drop values need to be multiplied by 10^{-5}).

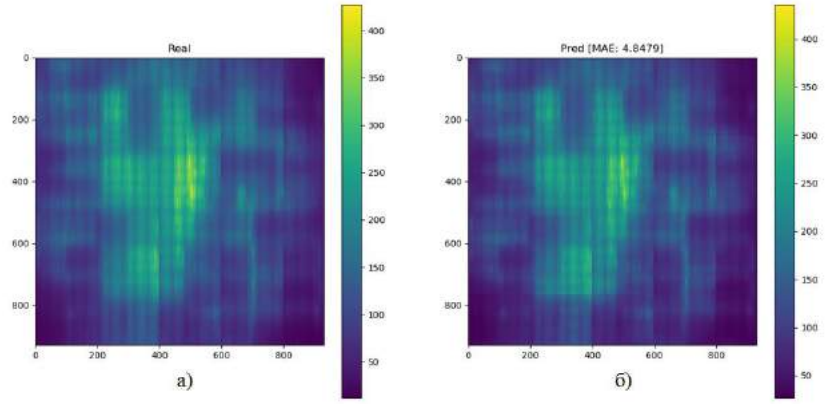


Рис. 9. Тестовая схема 3. Реальный IR drop (а), предсказанный IR drop (б) (значения IR drop нужно домножить на 10^{-5}).

Fig. 9. Testcase 3. Real IR drop (a), predicted IR drop (b) (IR drop values need to be multiplied by 10^{-5}).

Список литературы / References

[1]. Интернет-ресурс https://siliconvlsi.com/what-is-ir-drop/ (дата обращения – 27.11.2023).
[2]. Интернет-ресурс https://vlsi-backend-adventure.com/ir_analysis.html (дата обращения – 27.11.2023).
[3]. Интернет-ресурс https://teamvlsi.com/2020/07/ir-analysis-in-asic-design-effects-and.html (дата обращения – 27.11.2023).
[4]. Chhabria V. A., Ahuja V., Prabhu A., Patil N., Jain P., and Sapatnekar S. S. Thermal and IR Drop Analysis Using Convolutional Encoder-Decoder Networks. Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), 2021.
[5]. Chia-Tung Ho and Andrew B Kahng IncPIRD: Fast Learning Based Prediction of Incremental IR Drop. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019.
[6]. Zhiyao Xie, Haoxing Ren, Brucek Khailany, Ye Sheng, Santosh Santosh, Jiang Hu, and Yiran Chen PowerNet: Transferable Dynamic IR Drop Estimation via Maximum Convolutional Neural Network. Asia and South Pacific Design Automation Conference (ASP-DAC), 2020.
[7]. Chi-Hsien Pao, An-Yu Su, and Yu-Min Lee XGBIR: An xgboost-based IR drop predictor for power delivery network. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020
[8]. Chhabria V. A., Zhang Y., Ren H., Keller B., Khailany B., and Sapatnekar S. S. MAVIREC: ML-Aided Vectored IR-Drop Estimation and Classification. Proceedings of Design, Automation, and Test in Europe (DATE), 2021.
[9]. Интернет-ресурс https://2023.iccad.com/ (дата обращения – 27.11.2023).
[10]. Интернет-ресурс http://iccad-contest.org/ (дата обращения – 27.11.2023).
[11]. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18 (pp. 234-241). Springer International Publishing.
[12]. Интернет-ресурс https://skine.ru/articles/533011/ (дата обращения – 27.11.2023).
[13]. Интернет-ресурс https://academy.yandex.ru/handbook/ml/article/metriki-klassifikacii-i-regressii (дата обращения – 27.11.2023).
[14]. Интернет-ресурс https://eda.ncsu.edu/freepdk/freepdk45/ (дата обращения – 27.11.2023).
[15]. Интернет-ресурс https://github.com/UMN-EDA/BeGAN-benchmarks (дата обращения – 27.11.2023).
[16]. Chhabria V. A. et al. BeGAN: Power grid benchmark generation using a process-portable GAN-based methodology. 2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD). – IEEE, 2021. – С. 1-8.
[17]. Интернет-ресурс https://github.com/The-OpenROAD-Project (дата обращения – 27.11.2023).
[18]. Ajayi T., Blaauw D. Openroad: Toward a self-driving, open-source digital layout implementation tool chain. Proceedings of Government Microcircuit Applications and Critical Technology Conference. – 2019.
[19]. Интернет-ресурс https://electroandi.ru/toe/metod/metod-uzlovykh-potentsialov.html (дата обращения 27.11.2023).
[20]. Интернет-ресурс http://vscripts.ru/w/Схемы_ISCAS85 (дата обращения 27.11.2023).
[21]. Kirillov, A., He, K., Girshick, R., & Dollár, P. (2017). A unified architecture for instance and semantic segmentation. In *CVPR*.
[22]. Tan M., Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. International conference on machine learning. – PMLR, 2019. – С. 6105-6114.
[23]. Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., & Li, Y. (2022, October). Maxvit: Multi-axis vision transformer. European conference on computer vision (pp. 459-479). Cham: Springer Nature Switzerland.
[24]. Shanmugam D. et al. Better aggregation in test-time augmentation //Proceedings of the IEEE/CVF international conference on computer vision. – 2021. – С. 1214-1223.
[25]. Интернет-ресурс https://ngspice.sourceforge.io/ (дата обращения 27.11.2023).

Информация об авторах / Information about authors

Роман Александрович СОЛОВЬЁВ – чл.-кор. РАН, доктор технических наук, главный научный сотрудник Института Проблем Проектирования в Микроэлектронике РАН. Сфера научных интересов: автоматизация проектирования цифровых СБИС, нейронные сети и машинное обучение, система остаточных классов, физический синтез интегральных схем.

Roman Aleksandrovich SOLOVYEV – Dr. Sci. (Tech.), member-corr. RAS, chief researcher at the Institute of Design Problems in Microelectronics RAS. Area of scientific interests: automation of digital VLSI design, neural networks and machine learning, residue number system, physical synthesis of integrated circuits.

Дмитрий Владимирович ТЕЛЬПУХОВ – доктор технических наук, заместитель директора по научной работе Института Проблем Проектирования в Микроэлектронике РАН. Сфера научных интересов: автоматизация проектирования цифровых СБИС, логический синтез, радиационно-стойкое проектирования, машинное обучение и нейронные сети, система остаточных классов.

Dmitry Vladimirovich TELPUKHOV – Dr. Sci. (Tech.), Deputy Director for Scientific Work of the Institute of Design Problems in Microelectronics RAS. Research interests: automation of digital VLSI design, logic synthesis, radiation-resistant design, machine learning and neural networks, residual class system.

Евгений Денисович ДЕМИДОВ – аспирант 2 курса «Национального исследовательского университета «МИЭТ», инженер-разработчик отдела разработки средств автоматизированного проектирования СБИС в компании ООО «Альфачип». Тема научно-исследовательской работы: «Разработка средств анализа IR drop эффекта с применением методов машинного обучения». Сфера научных интересов: автоматизация проектирования цифровых СБИС, специальные виды анализа СБИС, машинное обучение и нейронные сети.

Evgeny Denisovich DEMIDOV – 2nd year postgraduate student of National Research University "MIET", development engineer computer-aided design tools of VLSI development department at Alphachip Company. Theme of scientific research work: «Development of tools for analyzing IR drop effect using machine learning methods». Research interests: automation of digital VLSI design, special types of VLSI analysis, machine learning and neural networks.

Илья Ильич ШАФЕЕВ – студен 2 курса магистратуры «Национального исследовательского университета «МИЭТ», инженер-исследователь в Институте Проблем Проектирования в Микроэлектронике РАН. Сфера научных интересов: автоматизация проектирования цифровых СБИС, специальные виды анализа СБИС, машинное обучение и нейронные сети.

Ilya Ilyich SHAFEEV – 2nd year master’s student of National Research University "MIET", research-engineer of the Institute of Design Problems in Microelectronics RAS. Research interests: automation of digital VLSI design, special types of VLSI analysis, machine learning and neural networks.

DOI: 10.15514/ISPRAS-2023-35(5)-10

CC BY

Применение нейронных сетей для сегментации изображений в задаче быстрой трассировки интегральных схем

Р.А. Соловьев, ORCID: 0000-0003-0312-452X <roman.solovyev.zf@gmail.com>
Т.М. Кадирлиев, ORCID: 0009-0004-7872-5164 <perrysquaredz@gmail.com>
Д.В. Тельпухов, ORCID: 0000-0001-9551-0748 <telpukhov@ippm.ru>

Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН), 124365 Москва, Зеленоград, ул. Советская, дом 3

Аннотация. В работе исследуются возможности применения нейросетевых методов для решения задачи глобальной трассировки интегральных схем. Разработан алгоритм генерации обучающей выборки на основе волнового алгоритма Ли, позволяющий синтезировать трехмерные матрицы с препятствиями и точками, которые нужно соединить. Для обучения выбрана полносверточная нейронная сеть U-Net, эффективная для семантической сегментации изображений. Проведена оценка качества результатов на тестовой выборке. Показано значительное сокращение времени трассировки по сравнению с волновым методом, однако доля маршрутов без разрывов составила лишь 37%. Предложены пути улучшения обучающей выборки и адаптации подхода под реальные условия с использованием файлов DEF и GUIDE. В работе продемонстрирован потенциал нейросетевых методов для ускорения задачи трассировки, однако требуется продолжение исследований для повышения качества и надежности результатов. Работа полезна для специалистов в области проектирования интегральных схем и машинного обучения.

Ключевые слова: машинное обучение; нейронные сети; волновой алгоритм; глобальная трассировка;

Для цитирования: Соловьев Р.А., Кадирлиев Т.М., Тельпухов Д.В. Применение нейронных сетей для сегментации изображений в задаче быстрой трассировки интегральных схем. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 145–156. DOI: 10.15514/ISPRAS–2023–35(5)–10.

Application of Neural Networks for Image Segmentation in the Problem of Fast Global Routing

R.A. Soloviev, ORCID: 0000-0003-0312-452X <roman.solovyev.zf@gmail.com>
T.M. Kadirliev, ORCID: 0009-0004-7872-5164 <perrysquaredz@gmail.com>
D.V. Telpukhov, ORCID: 0000-0001-9551-0748 <telpukhov@ippm.ru>

Institute of Design Problems in Microelectronics, RAS (IPPM RAS), 3, Sovetskaya st., Moscow, Zelenograd, 124365, Russia.

Abstract. The paper explores the possibilities of using neural network methods to solve the problem of global routing for VLSI ASIC design. An algorithm has been developed for generating a training dataset based on the Lee algorithm, which allows one to synthesize three-dimensional matrices with obstacles and points that need to be connected. The U-Net fully convolutional neural network, effective for semantic segmentation of images, was selected for training. The quality of the results was assessed using a validation data. A significant reduction in routing time compared to the Lee algorithm was shown, but the share of unbroken routes was only 37%. Ways to improve the training dataset and adapt the approach to real conditions using DEF and GUIDE files are

proposed. In general, the work demonstrated the potential of neural network methods to speed up the global routing task, but continued research is required to improve the quality and reliability of the results. The work is useful for specialists in the field of integrated circuit design and machine learning.

Keywords: machine learning; neural networks; lee algorithm; global routing.

For citation: Solovyev R.A., Kadirliev T.M., Telpukhov D.V. Application of Neural Networks for Image Segmentation in the Problem of Fast Global Routing Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 145-156 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-10.

1. Введение

Возрастающая сложность современных цифровых интегральных схем обуславливает новые задачи и повышающиеся требования в области средств автоматизации проектирования. Одним из способов улучшения качества получаемых результатов является применение новых быстрых алгоритмов и моделей, которые позволяют сократить время проектирования, а значит – открывают возможности для более широкого исследования пространства возможных решений поставленной задачи.

Одним из наиболее трудоемких этапов процесса проектирования интегральных схем является трассировка – этап, следующий за размещением элементов схемы и заключающийся в соединении полученных логических элементов проводниками. Эффективная и быстрая трассировка критически важна для обеспечения корректной работы проектируемого устройства.

Учитывая, что на трассировку приходится значительная доля временных затрат всего процесса физического синтеза, актуальной задачей представляется разработка подходов, позволяющих ускорить данный этап. Одним из перспективных направлений является применение методов машинного обучения, базирующихся на выявлении зависимостей между входными и выходными данными.

В данной статье предлагается подход к созданию и оценке качества работы нейронной сети, способной имитировать работу волнового алгоритма трассировки. Полученные результаты позволяют оценить потенциал использования предложенного подхода для ускорения процесса трассировки интегральных схем в реальных маршрутах проектирования.

2. Реализация волнового алгоритма

В качестве базового алгоритма трассировки в данном исследовании был выбран волновой алгоритм ввиду двух его ключевых преимуществ: во-первых, данный алгоритм гарантирует нахождение пути между двумя заданными точками (при условии его существования); во-вторых, найденный им путь является кратчайшим. Указанные свойства являются критически важными для формирования корректного набора данных, необходимого для обучения нейронной сети, несмотря на то, что волновой алгоритм не относится к числу наиболее высокоскоростных [1].

На начальном этапе алгоритма формируется трехмерное пространство поиска, в котором генерируются препятствия и случайным образом выбираются стартовая и финишная ячейки. Далее от стартовой ячейки осуществляется переход в соседние ячейки в рамках трёхмерной окрестности фон Неймана (без учёта диагональных ячеек). При этом проверяется, являются ли данные ячейки свободным пространством и не были ли они помечены как посещённые на предыдущих итерациях алгоритма.

На этапе распространения "волны" в успешно прошедшие проверку ячейки заносится значение, равное номеру текущей итерации (количеству "шагов" от стартовой ячейки). Данные ячейки становятся новыми источниками, относительно которых выполняются последующие итерации описанной проверки. Процесс повторяется до тех пор, пока

очередная распространяющаяся "волна" не достигнет финишной ячейки либо не будет обнаружено свободного пространства для дальнейшего распространения.

Восстановление кратчайшего пути выполняется в обратном направлении: на каждой итерации выбирается соседняя ячейка с маркировкой на единицу меньше текущей, что позволяет последовательно приближаться к стартовой ячейке. Алгоритм завершает работу при достижении стартовой ячейки [1].

Результатом выполнения данного подхода будет являться матрица, содержащая ячейки, обозначающие путь между стартом и финишем. Для простоты визуализации, на рис. 1 представлен пример работы данного подхода в двумерном пространстве.

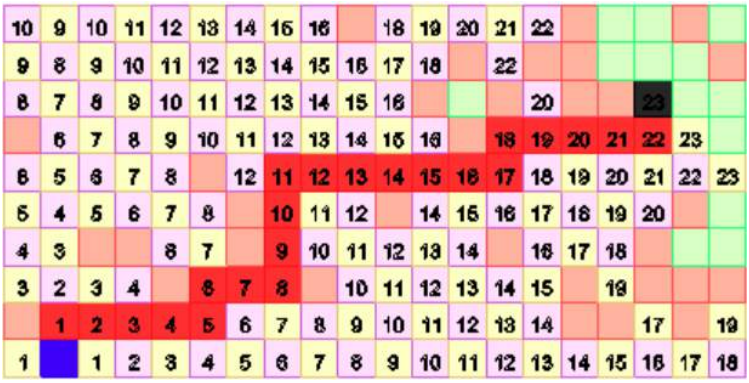


Рис.1. Все шаги волнового алгоритма в двумерном пространстве с выделенным восстановленным путем.

Fig.1. All steps of Lee's algorithm in a two-dimensional space with a highlighted reconstructed path.

3. Создание тренировочного набора данных

3.1 Входные данные

В качестве информации, которая подается на вход нейронной сети, используется специально созданный синтетический датасет, который является набором данных для обучения нейронной сети. Датасет при данном подходе состоит из пар задача-решение, которые используются для построения зависимостей, на основе которых нейронной сетью производятся предсказания [2]. Входом для задачи является трехмерная матрица с размерами 64x64x3 (3 – количество слоев металлизации) на которой размещаются препятствия, а также стартовая и финишная ячейки, которые используются волновым алгоритмом для построения пути.

Алгоритм генерации датасета можно условно разбить на 3 части:

- Создание препятствий;
- Размещение стартовой и финишной ячейки;
- Построение пути.

Генерация препятствий основывается на использовании дождевых матриц (рис. 2). В качестве первого этапа формируются двумерные матрицы заданного размера, в которых в случайно выбранных столбцах генерируются линии различной длины. Для каждого из трех слоев металлизации создаются две такие матрицы: одна с вертикальными линиями и вторая, полученная транспонированием первой и содержащая горизонтальные линии. Затем указанные пары матриц объединяются в единую матрицу, включающую препятствия обоих направлений для данного слоя металлизации.

Так как задачей в данном проекте является матрица 64x64x3, то аналогичный процесс повторяется для каждого из трех слоев металлизации.

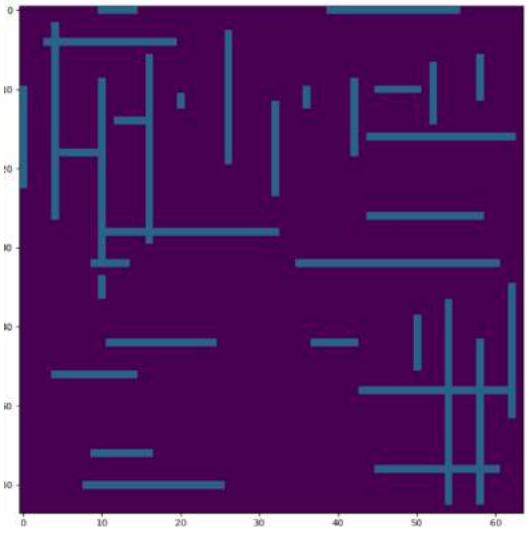


Рис.2. Пример матрицы препятствий для одного слоя металлизации.
Fig.2. Example of a matrix with obstacles in one metallization layer.

Сгенерированные с помощью дождевых матриц препятствия обозначаются значением «-1», чтобы их наличие не нарушало работы волнового алгоритма, который работает только с положительными значениями. По той же причине стартовая и финишная ячейки обозначаются как «-2» и «-3», а ячейки пустого пространства заданы нулями.

Размещение стартовой и финишной ячеек осуществляется путём генерации случайным образом трёх координат (X, Y и Z) в диапазоне размерностей матрицы для каждой ячейки. Далее для выбранных ячеек дополнительно проверяется отсутствие наложения на имеющиеся препятствия, с повторной генерацией координат при необходимости.

Построение пути производится описанным ранее волновым алгоритмом, на вход которого поступает трехмерная матрица с размещенными препятствиями и ячейками старта и финиша. В качестве результата работы алгоритма формируется матрица такой же размерности, заполненная нулевыми значениями. На позиции ячеек, соответствующих координатам пути, найденного с помощью волнового алгоритма, присваивается маркерное значение (в данном случае – "1"). Пример исходной и результирующей матриц показан на рис. 3.

4. Описание архитектуры нейронной сети

4.1 Общие сведения о подходе обучения нейронной сети

В работе использовался подход "обучение с учителем" (supervised learning), заключающийся в построении нейронной сетью функциональной зависимости между входными данными (представляющими задачу) и соответствующими им выходными данными (решениями) [3].

Цель обучения в рамках данного подхода состоит в минимизации среднеквадратичного отклонения между выходами нейронной сети и эталонными решениями по всей совокупности объектов тренировочной выборки. Иными словами, в процессе обучения происходит подбор оптимальных параметров нейронной сети, позволяющих максимально точно аппроксимировать зависимость между входными и выходными данными.

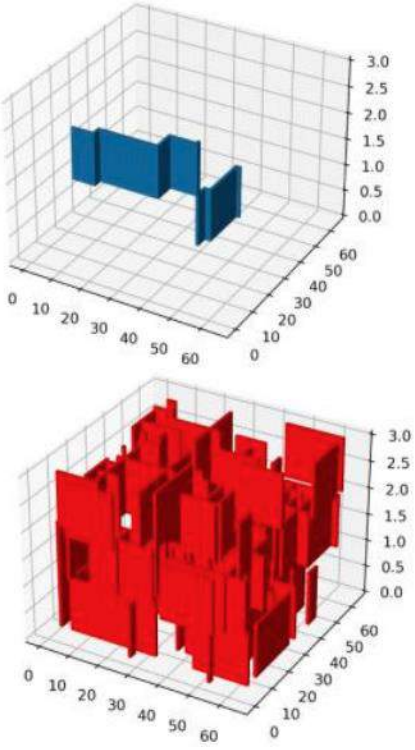


Рис.3. Матрица с маршрутом, построенным волновым алгоритмом (сверху) и исходная матрица с препятствиями (снизу), красным цветом изображены препятствия.
Fig.3. Matrix with the path created by Lee algorithm (top) and initial matrix with the obstacles (bottom), obstacles are marked with red color.

4.2 Архитектура Unet

Сеть Unet (см. рис. 4), является полностью сверточной нейронной сетью, разработанной для биомедицинской сегментации изображений. Ее архитектура основана на парадигме кодировщик-декодировщик с пропускающими соединениями (skip connections), которые позволяют точно локализовать объекты [4].

Архитектура сети Unet состоит из двух основных частей: энкодера и декодера. Энкодер следует типичной структуре сверточной нейронной сети и состоит из повторяющихся сверточных и подвыборочных слоев [4]. Этот путь направлен на захват контекста и извлечение высокоуровневых признаков из входного изображения. С другой стороны, декодер повышает размеры карт признаков для восстановления сегментированного изображения.

Уникальным аспектом сети Unet являются пропускающие соединения, которые объединяют соответствующие слои между сжимающим и расширяющими путями. Эти соединения позволяют сети объединять низкоуровневые и высокоуровневые признаки, обеспечивая точную локализацию и детальную сегментацию, что является важным аспектом для поставленной нами задачи, так как в ее основе лежит взаимодействие с объектами малых размеров.

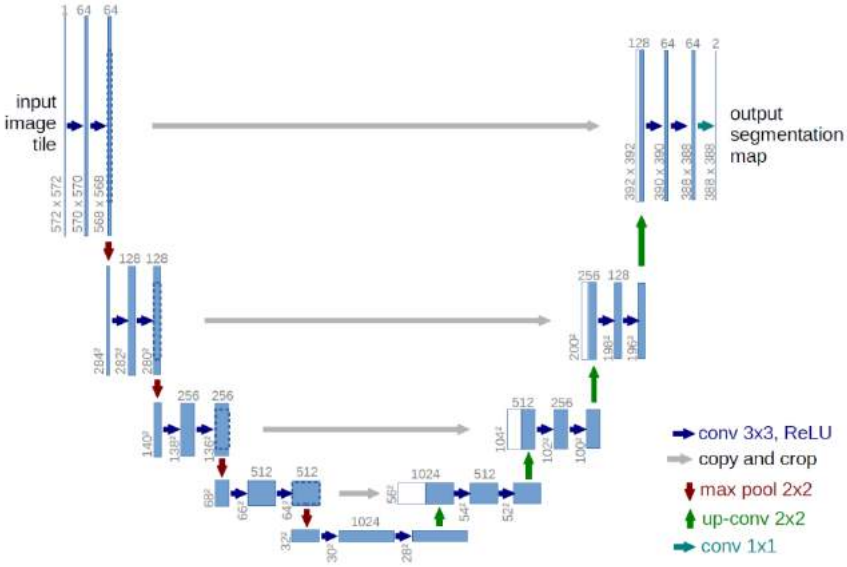


Рис.4. Архитектура Unet.
Fig.4. Unet architechture.

4.3 Подача данных в Unet

Используя тот факт, что обычные изображения рассматриваются как двумерные матрицы, у которых каждый цвет подается в отдельный входной канал, можно применить схожий подход для датасета описанного в разделе 3.1, но вместо цвета каждый канал Unet будет получать один слой металлизации, как для задачи, так и для решения.

Следовательно, результатом работы нейронной сети будет являться матрица с размерами идентичными исходным и содержащая только предсказание пути, полученное с помощью нейронной сети Unet.

5. Оценка работы нейронной сети

В качестве метрики функции потерь в предложенном методе используется коэффициент игральной кости Dice [5] – одна из наиболее часто применимых метрик оценки результатов сегментации свёрточных нейронных сетей.

$$D = \frac{2\sum_i^N p_i\, g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

В формуле коэффициента значения p_i и g_i являются парами соответствующих значений для предсказанных (predicted) и реальных значений (ground truth). В случае, когда нужно обнаружить путь, данные значения будут либо «0», либо «1» (если пиксель является путем, то значение будет равно 1). Соответственно знаменатель – это сумма всех пикселей пути, как предсказанного, так и решения, соответственно числитель – это сумма корректно предсказанных пикселей, поэтому сумма увеличивается только если оба значения p_i и g_i совпадают (оба равны 1). Следовательно, данный коэффициент лежит в диапазоне от 0 до 1, где 1 – абсолютное совпадение множеств, 0 – отсутствие каких-либо пересечений между множествами. Так как функция потерь должна стремиться к нулю, то вид самой функции потерь, используемой нейронной сетью во время работы будет 1-D. Соответственно функция валидации будет являться просто самим коэффициентом D, так как она должна показывать процент совпадений между предсказаниями и предоставленными решениями.

6. Тестирование

На этапе тренировки была использована часть датасета размерностью 90 тысяч элементов и функция потерь 1-Dice, а на этапе валидации часть датасета размерностью 10 тысяч элементов и коэффициент Dice.

Также для полученных предсказаний было необходимо провести постобработку, заключающуюся в переопределении значений матриц предсказаний. Необходимо было выбрать порог, превышение которого свидетельствовало бы о наличии пути, а недостижение порога – его отсутствию.

Для тестирования разработанной модели на новом наборе данных помимо проверки на схожесть между предсказанной матрицей и матрицей пути, было необходимо проверить предсказанный путь на разрывы. Это необходимо так как даже при большой корреляции с волновым алгоритмом, отсутствие одной ячейки может означать отсутствие пути между стартовой и финишной ячейкой. Для оценки матриц на разрывы был использован рекурсивный алгоритм:

1) Алгоритм инициализируется в стартовой ячейке.
2) По аналогии с волновым алгоритмом проверяются 6 соседних ячеек, но не на наличие свободного пространства, а на наличие пути, то есть «1».
3) Если такие ячейки найдены, в ячейку, вокруг которой происходил поиск путей, записывается «0» для предотвращения цикличного бесконечного вызова функции самой себя.
4) В обнаруженных в пункте 2 ячейках, для каждой (если их несколько), вызываются функции проверки 6 соседних ячеек.
5) Пункты 2,3,4 повторяются до тех пор, пока не будет достигнута финишная ячейка, что даст на выходе значение «1», либо пока на очередном вызове пункта 2 не будет обнаружено ни одной ячейки – что будет обозначать разрыв пути.

Помимо проверки на разрывы было проведено сравнение длин путей, полученных в результате работы, как волнового алгоритма, так и предсказаний нейронной сети. Для каждого подхода средняя длина была найдена сложением всех единиц во всех матрицах и делением полученного значения на размер датасета.

7. Анализ полученных результатов

Размер датасета, созданного для тренировки нейронной сети равен 100 тысячам элементов с разбиением 90% на тренировочную часть и 10% на валидационную часть. Процесс тренировки нейронной сети длился 40 эпох.

График коэффициента Dice в режиме валидации, то есть средний процент пересечения между предсказанными матрицами и матрицами решения (рис. 5).

Рассмотрим визуализацию успешного предсказания пути с помощью нейронной сети (см. рис. 6).

Также рассмотрим и неудачное предсказание, полученное нейронной сетью (см. рис. 7). На нем можно заметить, как при наличии более комплексных препятствий начинают возникать трудности с построением пути при необходимости прохода через несколько слоев металлизации.

Для оценки работы нейронной сети на предсказание путей без разрывов было создано 3 датасета различной размерности, содержащих 1000, 5000 и 10000 матриц, по ним был вычислен средний процент успешно проложенных путей. Для каждого из них также была проведена оценка на время работы по сравнению с волновым алгоритмом и оценка средней длины путей для всех 16000 матриц, см. табл. 1.

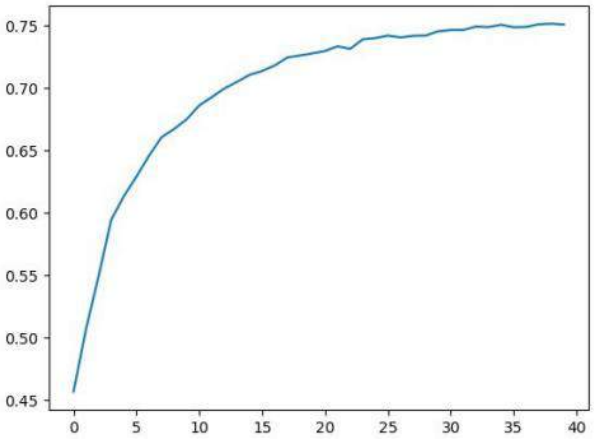


Рис.5. График коэффициента Dice на этапе валидации.
Fig.5. Graph of dice coefficient for validation stage.

По результатам, представленным в таблице, можно сделать вывод, что в среднем скорость работы нейронной сети выше скорости работы волнового алгоритма в 57 раз.

Также можно отметить, что вычислительная сложность волнового алгоритма $O(N^2)$, то есть, чем больше схема, тем в среднем длиннее будут пути и выполнение алгоритма будет занимать гораздо больше времени.

После применения рекурсивного алгоритма на 16 тысяч матриц количество путей без разрывов, то есть корректно построенных путей, было равно 37%.

Средняя длина пути для волнового алгоритма: 45,23.

Средняя длина пути для предсказания: 43,45.

Разница в 4% между длинами пути объясняется наличием разрывов в полученных предсказаниях.

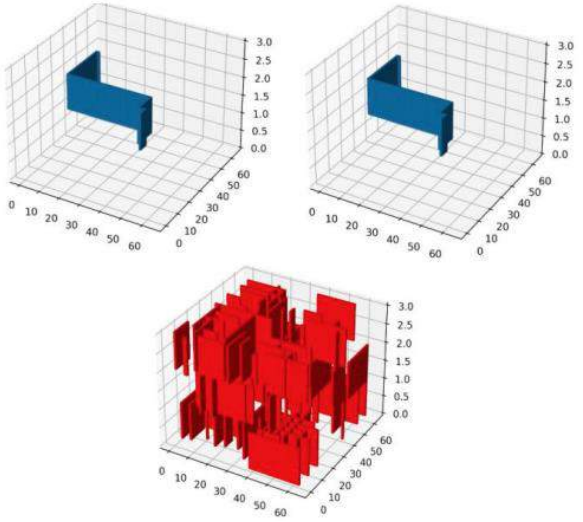


Рис.6. Путь, построенный волновым алгоритмом (слева сверху), пример удачного предсказания (справа сверху) и исходная матрица с препятствиями (снизу), красным цветом изображены препятствия.
Fig.6. Path created by Lee algorithm (top left), example of a successful prediction (top right) and source matrix with obstacles (bottom), obstacles are marked with red color.

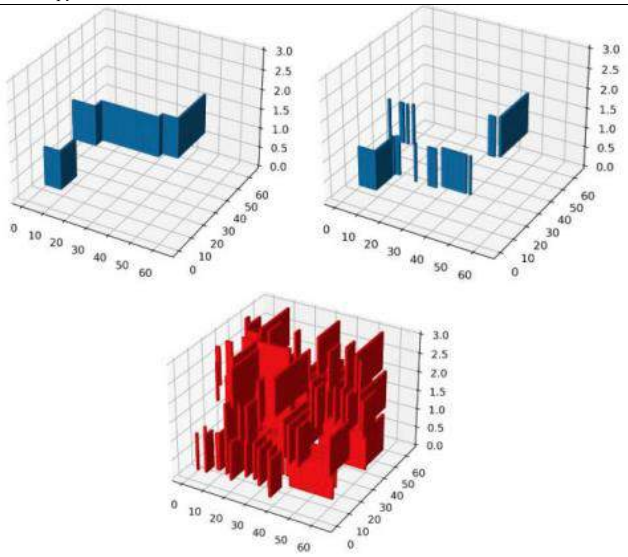


Рис.7. Путь, построенный волновым алгоритмом (слева сверху), пример неудачного предсказания (справа сверху) и исходная матрица с препятствиями (снизу), красным цветом изображены препятствия.
Fig.7. Path created by Lee algorithm (top left), example of an unsuccessful prediction (top right) and source matrix with obstacles (bottom), obstacles are marked with red color.

Табл. 1. Оценка скорости работы двух методов решения задачи трассировки
Table 1. Evaluation of the completion time for two methods of solving the global routing task

Количество матриц	1000	5000	10000
Время работы волнового алгоритма	140 секунд	728 секунд	1477 секунд
Время работы нейронной сети	3 секунды	12 секунд	23 секунды

8. Возможности практического применения

8.1 Общие сведения об архитектуре OpenLane

Для полноценного применения на практике данную нейронную сеть будет необходимо внедрить в один из существующих открытых маршрутов проектирования интегральных схем. Для этого было выбрано открытое программное обеспечение OpenLane [6].

OpenLane включает в себя множество этапов (см. рис. 8), которые позволяют получить из исходного RTL описания и библиотеки стандартных ячеек, набор данных необходимых для производства интегральных схем (GDSll, LEF и т.п.).

Официальная документация делит работу программы на 7 основных этапов: Synthesis, Floorplaning, Placement, CTS, Routing, Tapeout, Signoff.

Рассмотрим в подробностях три интересующих нас момента:

CTS (Clock Tree Synthesis) – данный этап необходим для равномерного распределения синхросигнала между всеми частями дизайна, что позволяет минимизировать смещение и задержки. Важной для нас деталью здесь является то, что в результате выполнения CTS мы будем иметь состояние интегральной схемы до этапа трассировки.

Routing – этап, включающий в себя глобальную и детальную трассировку, определение паразитных сопротивлений и емкостей, а также оценку задержек и целостности сигналов. Из

данного этапа мы будем извлекать информацию о расположении проводников, полученных в результате работы FastRoute – алгоритма для глобальной трассировки.

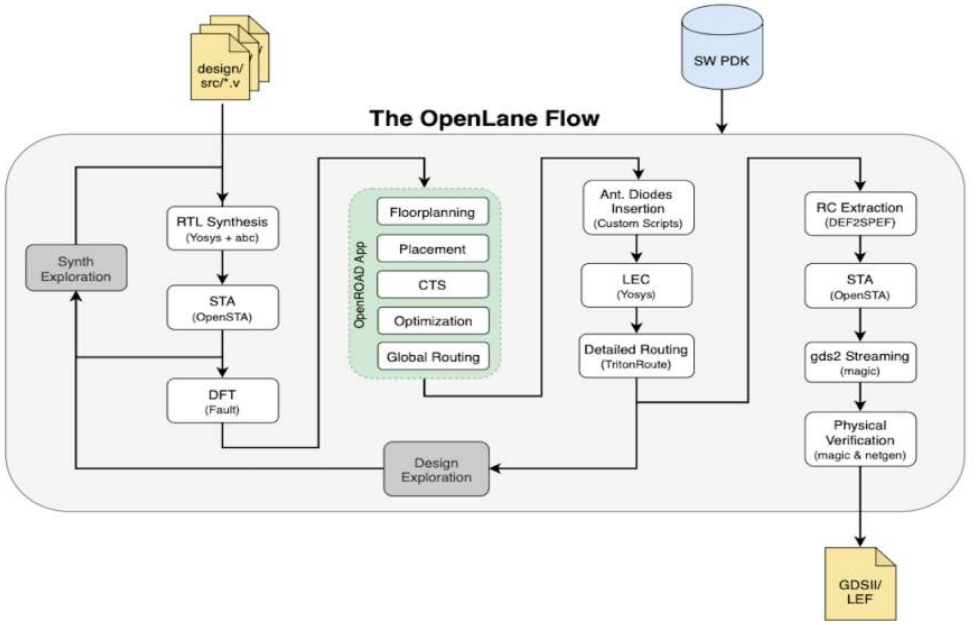


Рис.8. Маршрут синтеза от RTL до GDS – OpenLane.
Fig.8. Synthesis flow from RTL to GDS – OpenLane.

8.2 Выборка файлов

Рассмотрим сами файлы, выбранные нами ранее из различных этапов работы OpenLane. Из CTS будет использоваться DEF (Design Exchange Format) файл – он хранит в себе логическую и физическую информацию о дизайне [7] – то есть внутренние соединения, информацию о группах и контактах, физические ограничения (локация, ориентация и геометрия компонентов). Если посмотреть на то, как выглядят препятствия, сгенерированные дождевыми матрицами (рис. 9), то можно заметить сходства между изображениями.

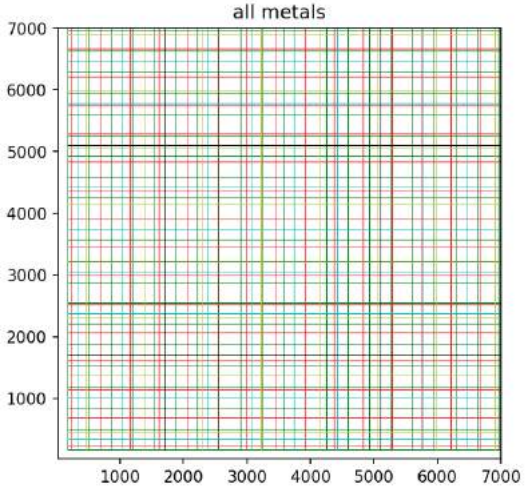


Рис.9. Треки извлеченные из DEF файла для всех слоев металлизации.
Fig.9. Tracks extracted from the DEF file for all metallization layers.

Из Routing будет взят GUIDE файл – в нем содержится информация о путях, проложенных в результате глобальной трассировки. Они разбиты на множество прямоугольных фигур на различных слоях металлизации, и хранятся в виде координат для двух противолежащих углов прямоугольных фигур и слоя металлизации для данной части пути. Данные, полученные из этого файла, будут являться результатом работы программы FastRoute – алгоритма глобальной трассировки, в отличие от рассматриваемого в данной статье волнового алгоритма.

В итоге, рассмотрев информацию, полученную из DEF и GUIDE файлов, можно сделать вывод о том, что подход, который использовался в данной работе, может быть применен и для информации, которую можно получить из реального маршрута проектирования. Для создания датасета с реальными данными будет необходимо провести следующие шаги:

1) Выбрать случайную группу контактов, относящихся к одному пути.
2) Удалить из матрицы полученной из GUIDE файла путь, относящийся к данным контактам.
3) Скомбинировать матрицу с выбранной группой контактов, матрицу с удаленным путем и матрицу с состоянием интегральной схемы до трассировки.
4) В результате выполнения этих действий для различных схем, будет сформирован датасет, на основе которого будет обучена модель сегментации на основе UNET.

9. Выводы

Предложенный подход обозначил достоинства и недостатки применения нейронных сетей для задачи глобальной трассировки интегральных схем, в целом показав свою применимость и перспективность. К преимуществам предложенного подхода можно отнести высокое быстродействие уже обученной модели: в 57 раз быстрее по сравнению с методом волнового алгоритма. При этом количество корректно построенных путей оказалось невысоким: всего 37%. Однако решение данной проблемы возможно следующими способами:

1) улучшение датасета за счёт расширения методики генерации исходных матриц;
2) включение признака разводимости в функцию потерь при обучении модели;
3) выбор более эффективной архитектуры нейронной сети.

Анализ современных маршрутов проектирования свидетельствует о возможности адаптации предложенного подхода к реальным открытым маршрутам проектирования, что вкупе с обозначенными возможностями для усовершенствования обуславливает широкие перспективы для использования нейронных сетей в задаче глобальной трассировки интегральных схем.

Список литературы / References

[1]. F. Rubin, "The Lee Path Connection Algorithm," in IEEE Transactions on Computers, vol. C-23, no. 9, pp. 907-914, Sept. 1974, doi: 10.1109/T-C.1974.224054.

[2]. James, Gareth & Witten, Daniela & Hastie, Trevor & Tibshirani, Robert. (2013). An Introduction to Statistical Learning: With Applications in R., Springer, New York, NY https://doi.org/10.1007/978-1-4614-7138-7.

[3]. O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[4]. Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28.

[5]. Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Jorge Cardoso, M. (2017). Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In: Cardoso, M., et al. Deep

Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. DLMIA ML-CDS 2017 2017. Lecture Notes in Computer Science, vol 10553. Springer, Cham. https://doi.org/10.1007/978-3-319-67558-9_28

[6]. Ghazy, A.A., & Shalan, M. (2020). OpenLANE: The Open-Source Digital ASIC Implementation Flow. in Proc. Workshop on Open-Source EDA Technol. (WOSET), 2020, Art. no. 21.

[7]. LEF/DEF 5.8 Language Reference. Доступно по ссылке: https://corio-lis.lip6.fr/doc/lefdef/lefdefref/DEFSyntax.html, дата обращения — апрель 2023.

Информация об авторах / Information about authors

Тимур Маратович Кадирлиев – бакалавр, инженер-исследователь в Институте Проблем Проектирования в Микроэлектронике РАН. Сфера научных интересов: машинное обучение и нейронные сети, работа с большими данными, веб разработка, автоматизация проектирования цифровых СБИС.

Timur Maratovich Kadirliev – university graduate, engineer-researcher of the Institute of Design Problems in Microelectronics RAS. Research interests: machine learning and neural networks, data science, web development, automation of digital VLSI design.

Дмитрий Владимирович ТЕЛЬПУХОВ – доктор технических наук, заместитель директора по научной работе Института Проблем Проектирования в Микроэлектронике РАН. Сфера научных интересов: автоматизация проектирования цифровых СБИС, логический синтез, радиационно-стойкое проектирования, машинное обучение и нейронные сети, система остаточных классов.

Dmitry Vladimirovich TELPUKHOV – Dr. Sci. (Tech.), Deputy Director for Scientific Work of the Institute of Design Problems in Microelectronics RAS. Research interests: automation of digital VLSI design, logic synthesis, radiation-resistant design, machine learning and neural networks, residual class system.Роман Александрович СОЛОВЬЁВ – чл.-кор. РАН, доктор технических наук, главный научный сотрудник Института Проблем Проектирования в Микроэлектронике РАН. Сфера научных интересов: автоматизация проектирования цифровых СБИС, нейронные сети и машинное обучение, система остаточных классов, физический синтез интегральных схем.

Roman Aleksandrovich SOLOVYEV – Dr. Sci. (Tech.), member-corr. RAS, chief researcher at the Institute of Design Problems in Microelectronics RAS. Area of scientific interests: automation of digital VLSI design, neural networks and machine learning, residue number system, physical synthesis of integrated circuits.

DOI: 10.15514/ISPRAS-2023-35(5)-11

Оптимизация алгоритма деления чисел в системе остаточных классов на основе функции ядра Акушского

[1] *В.В. Луценко*, ORCID: 0000-0003-4648-8286 <*vvlutcenko@ncfu.ru*>
[2,3] *М.Г. Бабенко*, ORCID: 0000-0001-7066-0061 <*mgbabenko@ncfu.ru*>
[3,4] *А.Н. Черных,* ORCID: 0000-0001-5029-5212 <*chernykh@cicese.mx*>
[2] *М.А. Лапина*, ORCID: 0000-0001-8117-9142 <*mlapina@ncfu.ru*>

[1] *Северо-Кавказский центр математических исследований, Северо-Кавказский федеральный университет,*
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.
[2] *Северо-Кавказский федеральный университет,*
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.
[3] *Институт системного программирования РАН им. В.П. Иванникова,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[4] *Центр научных исследований и высшего образования Энсенада,*
В.С. 22860, Мексика.

Аннотация. Система остаточных классов широко применяются в криптографии, цифровой обработке сигналов, системах обработки изображений и других областях, где требуется выполнение операций деления. Однако, операция деления является наиболее сложной с точки зрения вычислений в системе остаточных классов. В статье представлен оптимизированный алгоритм деления, основанный на функции ядра Акушского. Показано, что предложенный алгоритм по скорости вычислений эффективней, чем классическое итерационное деление.

Ключевые слова: система остаточных классов; функция ядра Акушского; модулярная арифметика; немодульные операции; итерационное деление.

Для цитирования: Луценко В.В., Бабенко М.Г., Черных А.Н., Лапина М.А. Оптимизация алгоритма деления чисел в системе остаточных классов на основе функции ядра Акушского. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 157–168. DOI: 10.15514/ISPRAS–2023–35(5)–11.

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

Optimization of a Number Division Algorithm in the Residue Number System Based on the Akushsky Core Function

[1] *V.V. Lutsenko* ORCID: 0000-0003-4648-8286 <*vvlutcenko@ncfu.ru*>
[2,3] *M.G. Babenko* ORCID: 0000-0001-7066-0061 <*mgbabenko@ncfu.ru*>
[3,4] *A.N. Tchernykh*, ORCID: 0000-0001-5029-5212 <*chernykh@cicese.mx*>
[2] *M.A. Lapina*, ORCID: 0000-0001-8117-9142 <*mlapina@ncfu.ru*>

[1] *North-Caucasus Center for Mathematical Research, North-Caucasus Federal University, 1, Pushkin st., Stavropol, 355017, Russia.*
[2] *North-Caucasus Federal University, Stavropol, 1, Pushkin st., Stavropol, 355017, Russia.*
[3] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*
[4] *CICESE Research Center, Ensenada, Baja California, 22860, Mexico.*

Abstract. Residue number systems find wide application in cryptography, digital and image signal processing, and other domains necessitating division operations. Nevertheless, division is the most computationally intensive activity in residue number systems. An optimized division algorithm based on the Akushsky core function is presented in this paper. The suggested method exhibits superior computational efficiency when compared to the conventional iterative division.

Keywords: residue number system; Akushsky core function; modular arithmetic; non-modular operation; iterative division.

For citation: Lutsenko V.V., Babenko M.G., Tchernykh A.N., Lapina M.A. Optimization of a number division algorithm in the residue number system based on the Akushsky core function. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 157-168 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-11.

Acknowledgements. The research was supported by the Russian Science Foundation Grant No. 19-71-10033, https://rscf.ru/en/project/19-71-10033/.

1. Введение

В последнее десятилетие система остаточных классов (СОК) стала объектом повышенного внимания в сфере вычислительной техники. Это нетрадиционное представление чисел, основанное на арифметике остаточных классов, стало фундаментом для многочисленных исследований и практических применений в различных областях, включая цифровую обработку сигналов, системы обработки изображений, помехоустойчивое кодирование, криптографию, квантовые автоматы, нейрокомпьютеры, системы с массовым параллелизмом операций, облачные вычисления и многие другие [1–7]. Система остаточных классов используется в архитектуре математических сопроцессоров криптосистем RSA [14].

Одной из наиболее сложных операций в СОК является деление чисел. Сложность этой операции в СОК привела к необходимости разработки оптимизированных алгоритмов, способных обеспечивать быстрые результаты при выполнении деления, особенно в случаях, где требуются большие объемы вычислений.

Существующие алгоритмы деления в СОК обычно можно разделить на две категории: алгоритмы, использующие умножение [8], и алгоритмы, использующие вычитание [9]. Большинство алгоритмов, основанных на умножении, требуют предварительного вычисления обратной величины делителя, что влечет за собой дополнительные вычислительные затраты. В то время как алгоритмы, основанные на вычитании, могут обойтись без обратной величины делителя, они, в свою очередь, могут использовать другие немодульные операции, что также снижает эффективность.

Цель данной статьи – исследовать и усовершенствовать алгоритм деления чисел в системе остаточных классов с использованием функции ядра Акушского.

Статья имеет следующую структуру. В разделе 2 рассматривается система остаточных классов. В разделе 3 представлены основы функции ядра Акушского. Затем, в разделе 4 исследуется метод итерационного деления на основе функции ядра. В разделе 5 представлен способ оптимизации итерационного деления. Наконец, в разделе 6 проведен анализ эффективности предложенного метода. В заключении суммируются полученные результаты.

2. Система остаточных классов

Представление числа в СОК основано на китайской теореме об остатке (КТО). Пусть $p_1, p_2, \dots, p_n$ – взаимно простые модули, и $P = p_1 \cdot p_2 \cdot \dots \cdot p_n$ – их произведение, называемое динамическим диапазоном. Для каждого числа X, существует набор остатков $x_1, x_2, \dots, x_n$, где $0 \le x_i < p_i$, эти остатки образуют представление числа в СОК [10]. Математически это можно выразить следующим образом:

$$x_i \equiv X(mod\ p_i). \tag{1}$$

Таким образом, число X записывается в СОК в следующей форме:

$$X = (x_1, x_2, \dots, x_n), \tag{2}$$

Вычеты (1) x_i можно вычислить по формуле:

$$x_i = X - \left\lfloor \frac{X}{p_i} \right\rfloor \cdot p_i, \tag{3}$$

Для выполнения операций над числами в СОК, таких как сложение и умножение, операции выполняются независимо над остатками по каждому модулю. Вычисления в СОК выполняются по формуле:

$$X * Y = (x_1 * y_1, x_2 * y_2, \dots, x_n * y_n). \tag{4}$$

где $*$ обозначает арифметические операции: сложение $(+)$, вычитание $(-)$ или умножение $(\cdot)$. Каждый модуль СОК взаимно прост с другими модулями, т.е. выполняется условия: $\text{НОД}(p_i, p_j) = 1$, для всех $i \ne j$.

3. Функция ядра Акушского

Поиск позиционных характеристик, которые позволили бы уменьшить вычислительную сложность алгоритма сравнения чисел в СОК привели исследователей Акушского И.Я., Бурцева В.М. и Пака И.Т. к построению новой конструкции, названной функцией ядра Акушского [11]. Задается функция ядра Акушского следующей формулой:

$$C(X) = C_X = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{p_i} \right\rfloor. \tag{5}$$

где целые числа w_i – постоянные определяемые выбором точки интерполяции. Константы w_i задают вес каждого из частных $\left\lfloor \frac{X}{p_i} \right\rfloor$ в формуле (5) тем самым задавая функцию ядра и придавая ей различные свойства.

Веса w_i, участвующие в формуле (5), предоставляют нам значительную гибкость и могут быть подобраны с учетом конкретной цели. Эти коэффициенты w_i играют ключевую роль в определении уникальных функций ядра и могут быть настроены в соответствии с требованиями задачи, над которой мы работаем. Алгоритм определения оптимальных весов для функции Акушского представлен в статье [12].

Основное свойство функции ядра заключается в том, что ее максимальный диапазон изменяется и может быть значительно меньше числа P в зависимости от выбора весов. Например, мы можем использовать произвольное значение $C(P)$ в качестве $C(P)$, при

условии, что оно обладает необходимыми свойствами для решения нашей конкретной задачи. Это значение называется диапазоном функции ядра и определяется выражением.

$$C(P) = C_P = \sum_{i=1}^{n} w_i \cdot P_i. \quad (6)$$

где $P_i = \frac{P}{p_i}$.

Учитывая, что $P_i \equiv 0(mod\ p_i)$ для любого $i \neq j$, константы w_i этой функции могут быть определены из соотношения

$$w_i \equiv C(P) \cdot P_i^{-1}(mod\ p_i). \quad (7)$$

При этом необходимо учитывать, что (7) задает некий класс вычетов для каждого i, и числа w_i могут оказаться отрицательными или положительными в каждом конкретном случае.

Значение функции ядра $C(X)$, заданной весами $w_1, w_2, \dots, w_n$, при условии $0 \leq C(X) \leq C(P), X \in [0, P)$, можно вычислить с использованием формулы

$$C(X) = \left| \sum_{i=1}^{n} c_i \cdot x_i \right|_{C_P}. \quad (8)$$

где $c_i = C(B_i)$ и $B_i = P_i \cdot |P_i^{-1}|_{p_i}$, $|P_i^{-1}|_{p_i}$ это мультипликативная инверсия P_i по модулю p_i.

4. Итерационное деление

Рассмотрим применение алгоритма деления с использованием ядерной позиционной характеристики.

Алгоритм деления базируется на особой простоте выполнения элементарных операций деления на 2 и на основание СОК, а также на определения факта переполнения при сложении двух чисел [13].

Рассмотрим процесс итерационного деления делимого $A = (\alpha_1, \alpha_2, \dots, \alpha_n)$ с ядром C_A на делитель $B = (\beta_1, \beta_2, \dots, \beta_n)$ с ядром C_B поэтапно.

Первый этап – если $\beta_1 = 0$, то делим B на p_1 в противном случае делим B на 2, получаем B_1. Делим B_1 на p_1, если $\beta_1' = 0$, и на 2, в противном случае получаем B_2 и т.д. до $B_k = 1$.

Параллельно с этим делим A на p_1, если $\beta_1 = 0$, и на 2, в противном случае получаем A_1. Делим A_1 на p_1, если $\beta_1' = 0$, и на 2, в противном случае получаем A_2 и т.д. до A_k.

Второй этап – вычисляется первая невязка:

$$A - BA_k = Q^{(1)}. \quad (9)$$

Третий этап в соответствии с первым этапом деления делителя B вычисляется так:

$$Q_1, Q_2, \dots, Q_k^{(1)}. \quad (10)$$

При этом если имеется возможность запомнить соответствующие делители $d_1, d_2, \dots, d_k$ делителя B, который является как бы ведущим в рассматриваем процессе деления, то не надо повторять деление B, и содержание этапа сводится лишь к делению $Q^{(1)}$.

Четвертый этап – вычисляется вторая невязка:

$$Q_1 - Q_k^{(1)} \cdot B = Q^{(2)}. \quad (11)$$

Эти этапы повторяются до получения нулевого промежуточного частного $Q_i^{(l+1)} = 0$ $(i = 1,2, \dots, k)$. Тогда искомое частное:

$$Z = \frac{A}{B} = A_k + Q_k^{(1)} + Q_k^{(2)} + \cdots + Q_k^{(l)}, \quad (12)$$

где

$$Q_k^{(l+1)} = 0. \quad (13)$$

Таким образом, алгоритм итерационного деления чисел A и B будет иметь вид:

Алгоритм. Алгоритм итерационного деления чисел A и B.

Input: $A \xrightarrow{RNS} (\alpha_1, \alpha_2, \dots, \alpha_n)$,

$B \xrightarrow{RNS} (\beta_1, \beta_2, \dots, \beta_n)$,

$p_1, p_2, \dots, p_n, w_1, w_2, \dots, w_n$,

$P = p_1 \cdot p_2 \cdot \dots \cdot p_n, P_i = \frac{P}{p_i}, C_P = \sum_{i=1}^{n} w_i \cdot P_i$ для всех $i = \overline{1,n}$,

$c_i = C(B_i), B_i = P_i \cdot |P_i^{-1}|_{p_i}$ для всех $i = \overline{1,n}$,

Output: $Z = A/B$.

1. **Function** basic_division(X, k)

1.1. Initialize X_k, C_{X_k} to empty list;

1.2. **If** $x_1 = 0$ **then**

1.2.1. Append the $(x_1, x_2, \dots, x_n)/p_1$ to the list X_k;

1.2.2. Append the $C_{X_1} = C(X_1)$ to the list C_{X_k};

1.3. **Else**

1.3.1. Append the $(x_1, x_2, \dots, x_n)/2$ to the list X_k;

1.3.2. Append the $C_{X_1} = C(X_1)$ to the list C_{X_k};

1.4. **For** $i \coloneqq 2$ **to** k **do:**

1.4.1. **If** $x'_1 = 0$ **then**

1.4.1.1. Append the X_i/p_1 to the list X_k;

1.4.1.2. Append the $C_{X_i} = C(X_i)$ to the list C_{X_k};

1.5.2. **Else**

1.5.2.1. Append the $X_i/2$ to the list X_k;

1.5.2.2. Append the $C_{X_i} = C(X_i)$ to the list C_{X_k};

1.5. **return** X_k, C_{X_k}

2. Initialize A_k, C_{A_k}, B_k, C_{B_k} to empty list;

3. **If** $\beta_1 = 0$ **then**

3.1. Append the $(\beta_1, \beta_2, \dots, \beta_n)/p_1$ to the list B_k;

3.2. Append the $C_{B_1} = C(B_1)$ to the list C_{B_k};

3.3. Append the $(\alpha_1, \alpha_2, \dots, \alpha_n)/p_1$ to the list A_k;

3.4. Append the $C_{A_1} = C(A_1)$ to the list C_{A_k};

4. **Else**

4.1. Append the $(\beta_1, \beta_2, \dots, \beta_n)/2$ to the list B_k;

4.2. Append the $C_{B_1} = C(B_1)$ to the list C_{B_k};

4.3. Append the $(\alpha_1, \alpha_2, \dots, \alpha_n)/2$ to the list A_k;

4.4. Append the $C_{A_1} = C(A_1)$ to the list C_{A_k};

5. **while** $B_k \neq 1$ **do:**

5.1. **If** $\beta'_1 = 0$ **then**

5.1.1. Append the B_i/p_1 to the list B_k;

5.1.2. Append the $C_{B_i} = C(B_i)$ to the list C_{B_k};

5.1.3. Append the A_i/p_1 to the list A_k;

5.1.4. Append the $C_i = C(A_i)$ to the list C_{A_k};

5.2. **Else**

5.2.1. Append the $B_i/2$ to the list B_k;

5.2.2. Append the $C_{B_i} = C(B_i)$ to the list C_{B_k};

5.2.3. Append the $A_i/2$ to the list A_k;

5.2.4. Append the $C_i = C(A_i)$ to the list C_{A_k};

5. Initialize $Q^{(k)}, Q_k^{(l)}$ to empty list;

6. Append the $Q^{(1)} = A - BA_k$ to the list $Q^{(k)}$;

7. $Q_k^{(1)}, C_{Q_k^{(1)}} =$ basic_division$\left(Q^{(1)}, k\right)$

8. Append the $Q_k^{(1)}$ to the list $Q_k^{(l)}$;

9. Append the $Q^{(2)} = Q_1 - Q_k^{(1)} \cdot B$ to the list $Q^{(k)}$;

10. **while** $\boldsymbol{Q_i^{(l)} \neq 0}$ **do:**

10.1. $Q_k^{(i)}, C_{Q_k^{(i)}} =$ basic_division$\left(Q^{(i)}, k\right)$

10.2. Append the $Q_k^{(i)}$ to the list $Q_k^{(l)}$;

10.3. Append the $Q^{(i)} = Q_i - Q_k^{(i)} \cdot B$ to the list $Q^{(k)}$;

11. $Z = A_k + \sum_{i=1}^{l} Q_k^{(i)}$;

End.

Пример 1. Рассмотрим СОК с основаниями $p_1 = 7, p_2 = 9, p_3 = 11$, с системой весов $w_1 = -1, w_2 = -1, w_3 = 3$, ядром диапазона $C_P = 13$.

Пусть дано делимое $A = (5,7,10)$ с ядром $C_A = 7$ и делитель $B = (3,6,10)$ с ядром $C_B = 0$.

Первый этап:

$$B_1 = (3,6,10)/2 \approx (2,5,9)/2 = (1,7,10)\ c\ C_{B_1} = -1;$$
$$B_2 = (1,7,10)/2 \approx (0,6,9)/2 = (0,3,10)\ c\ C_{B_2} = -2;$$
$$B_3 = (0,6,9)/p_1 = (0,6,9)/7 = (3,3,3)\ c\ C_{B_3} = 0;$$
$$B_4 = (3,3,3)/2 \approx (2,2,2)/2 = (1,1,1)\ c\ C_{B_4} = 0.$$

Параллельно выполняем операции над делимым A:

$$A_1 = (5,7,10)/2 \approx (4,6,2)/2 = (2,3,10)\ c\ C_{A_1} = 2;$$
$$A_2 = (2,3,10)/2 \approx (1,2,9)/2 = (4,1,10)\ c\ C_{A_2} = 0;$$
$$A_3 = (4,1,10)/7 = (0,6,6)/7 = (1,6,4)\ c\ C_{A_3} = 0;$$
$$A_4 = (1,6,4)/2 \approx (0,5,3)/2 = (0,7,7)\ c\ C_{A_4} = -1.$$

Первое промежуточное частное $A_4 = (0,7,7)$с $C_{A_4} = -1$.

Второй этап – вычисляется первый невязка.

Для этого найдем $A_4 \times B = (0,7,7) \cdot (3,6,10) = (0,6,4)$ с истинным ядром $C_{\mathrm{A_4B}} = 11$.

Вычисляем функции четности A_4, B и $A_4 \cdot B$. Получаем $\varphi(A_4) = 1, \varphi(B) = 1, \varphi(\mathrm{A_4} \cdot B) = 1$. Так как $\varphi(A_4) \cdot \varphi(B) = \varphi(A_4 B)$, то, следовательно, $A_4 B < P$. Найдем невязку $Q^{(1)} = \mathrm{A} - A_4 B = (5,7,10) - (0,6,4) = (5,1,6)$ с расчетным ядром $C_{Q^{(1)}} = 9$.

Так как $\bar{C}_{Q^{(1)}} \neq C_{Q^{(1)}}$, то истинная невязка равна $Q_n^{(1)} = P - Q^{(1)} = (2,8,5)$ с ядром $C_{Q_n^{(1)}} = 3$ и знаком минус.

Третий этап – повторение первого этапа для $Q_n^{(1)}$:

$$Q_1^{(1)} = (2,8,5)/2 = (1,4,8)\ c\ C_{Q_1^{(1)}} = 0;$$
$$Q_2^{(1)} = (1,4,8)/2 \approx (0,3,7)/2 = (0,6,9)\ c\ C_{Q_2^{(1)}} = -1;$$
$$Q_3^{(1)} = (0,6,9)/7 = (6,6,6)\ c\ C_{Q_3^{(1)}} = 0;$$
$$Q_4^{(1)} = (6,6,6)/2 = (3,3,3)\ c\ C_{Q_4^{(1)}} = 0.$$

Второе промежуточное частное $Q_4^{(1)} = (3,3,3)$ с $C_{Q_4^{(1)}} = 0$.

Четвертый этап – вычисляется вторая невязка.

Следуя второму этапу, получим $Q_4^{(1)} \cdot \mathrm{B} = (3,3,3) \times (3,6,10) = (2,0,8)$ с истинным ядром $C_{Q_4^{(1)}\,\mathrm{B}} = 3$. Здесь $\varphi\left(Q_4^{(1)}\right) = 1, \varphi\left(Q_4^{(1)} \cdot \mathrm{B}\right) = 1$ и $\varphi\left(Q_4^{(1)}\right) \cdot \varphi(\mathrm{B}) = \varphi\left(Q_4^{(1)} \cdot B\right)$, т. е. $Q_4^{(1)} \cdot B < P$.

Найдем невязку с учетом знака минус у $Q_n^{(1)}Q^{(2)} = Q_n^{(1)} \cdot B - Q_4^{(1)} = (2,0,8) - (2,8,5) = (0,1,3)$ с расчетным ядром $\bar{C}_{Q^{(2)}} = 1$ и истинным ядром $C_{Q^{(2)}} = 1$.

Так как $\bar{C}_{Q^{(2)}} = C_{Q^{(2)}}$, то получена истинная невязка.

$$Q_1^{(2)} = (0,1,3)/2 \approx (6,0,2)/2 = (3,0,1) \text{ с } C_{Q_1^{(2)}} = 1;$$

$$Q_2^{(2)} = (3,0,1)/2 \approx (2,8,0)/2 = (1,4,0) \text{ с } C_{Q_2^{(2)}} = 1;$$

$$Q_3^{(2)} = (0,6,9)/7 \approx (0,3,10)/7 = (3,3,3) \text{ с } C_{Q_3^{(2)}} = 0;$$

$$Q_4^{(2)} = (6,6,6)/2 \approx (2,2,2)/2 = (1,1,1) \text{ с } C_{Q_4^{(2)}} = 0.$$

Третье промежуточное частное $Q_4 = (1,1,1)$ с $C_{Q_4^{(2)}} = 0$.

На этом деление можно закончить, так как Q_4 будет обязательно равно нулю.

Составим частное:

$\frac{A}{B} = \frac{(5,7,10)}{(3,6,10)} = (0,7,7) - (3,3,3) + (1,1,1) = (5,5,5)$ с расчетным ядром $C_{A/B} = -1 - 0 + 0 - (-1) = 0$.

Действительно,

$$\frac{A}{B} = \frac{(5,7,10)}{(3,6,10)} = \frac{439}{87} = 5\frac{4}{84}.$$

В следующем разделе рассмотрим возможность оптимизации расчета функции ядра.

5. Оптимизация итерационного деления

Как видно из предыдущего раздела, функция ядра играет важную роль при выполнении итерационного деления, возникает необходимость определения знака числа, а также уточнения значения ядра в случае возникновения так называемых критических ядер [11]. Для оптимизации итерационного деления докажем следующую теорему.

Теорема 1. $C\left(\frac{X}{2}\right) = \frac{C(X) - \sum_{|x_i|_2=1}^{n} w_i}{2}$.

Доказательство.

Так как значение функции ядра Акушского вычисляется по формуле (6), тогда:

$$C\left(\frac{X}{2}\right) = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{2p_i} \right\rfloor.$$

Рассмотрим функцию ядра в следующем виде:

$$C(X) = \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \cdot \frac{X}{2p_i} \right\rfloor = \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \cdot \left(\left\lfloor \frac{X}{2p_i} \right\rfloor + \left\{ \frac{X}{2p_i} \right\} \right) \right\rfloor$$

$$= \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \left\lfloor \frac{X}{2p_i} \right\rfloor + 2\left\{ \frac{X}{2p_i} \right\} \right\rfloor = 2\sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{2p_i} \right\rfloor + \sum_{i=1}^{n} w_i \cdot \left\lfloor 2\left\{ \frac{X}{2p_i} \right\} \right\rfloor$$

$$= 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1}^{n} w_i \cdot \left\lfloor 2\frac{|X|_{2p_i}}{2p_i} \right\rfloor = 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{|X|_{2p_i}}{p_i} \right\rfloor,$$

так как $|X|_2 = 0$ и $|X|_{p_i} = x_i$, тогда:

$$C(X) = 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{|x_i|_{2p_i} + x_i}{p_i} \right\rfloor = 2 \cdot C\left(\frac{X}{2}\right) + \sum_{\substack{i=1 \\ |x_i|_2=1}}^{n} w_i.$$

Отсюда следует, что:

$$C\left(\frac{X}{2}\right) = \frac{C(X) - \sum^{n}{}_{\substack{i=1 \\ |x_i|_2=1}} w_i}{2}.$$

Теорема доказана.

Предложенный подход позволяет увеличить скорость вычисления функции ядра Акушского, а значит увеличить эффективность выполнения итерационного деления в СОК.

6. Оценка эффективности

Для подтверждения свойств предложенной оптимизации метода деления мы реализуем его на языке Python и сравним производительность с классическим итерационным делением. Эксперименты проводились в операционной системе Windows 10 Home Edition на базе компьютера с процессором Intel Core i7-7700HQ 2,80 ГГц, оперативной памятью DDR4 8 ГБ 1196 МГц и твердотельным накопителем SSD 512 ГБ.

Эксперимент заключается в следующем, исследование проводится в два этапа:

Этап А – исследование производительности 4 наборов модулей, размерностью от 8 до 32 бит;

Этап Б – исследование производительности 8 комплектов, от 3 до 10 модулей, размерность каждого модуля 8 бит.

При проведении двухэтапного моделирования были получены временные характеристики каждого метода. В каждом из результатов было получено среднее значение после 100 итераций. Результаты отражены в табл. 1 и 2. Время указано в секундах.

Табл. 1. Результат этапа А (Бит - размер одного набора в битах)
Table 1. Result of stage A (Bit is the size of one set in bits)

Bit	Итерационное деление	Модифицированное итерационное деление
8	0.0000543	0.0000521
16	0.0000628	0.0000598
24	0.0000646	0.0000629
32	0.0000918	0.0000877

Из таблицы видно, что предложенный нами метод, использующий для расчета функции ядра форму, предложенную в разделе 5, в среднем на 4% быстрее. Уменьшение вычислительных затрат приводит к экономии энергии и снижению энергопотребления, что делает его перспективным для вычислительных систем. Кроме того, эффективность алгоритма может привести к снижению требований к аппаратному обеспечению, что еще больше повышает его пригодность для использования в средах с ограниченными ресурсами.

7. Заключение

В данной статье мы исследовали оптимизацию алгоритма деления чисел в системе остаточных классов с использованием функции ядра Акушского. Был разработан более эффективный и быстрый алгоритм расчета функции ядра Акушского, для использования в итерационном делении.

Мы провели эксперименты и показали, что наш метод улучшает производительность итерационного деления. В будущих исследованиях планируется реализовать аппаратную

реализацию предлагаемого алгоритма, что позволит дать более точную оценку времени, потреблении энергии и площади.

Алгоритмы деления в системе остаточных классов, в том числе предложенная нами модификация итерационного деления имеют ограничение, связанное с тем, что результатом деления может быть только целое число. Однако, нами ведутся поиски метода, позволяющего разрешить данную проблему.

Результаты нашей статьи могут быть использованы в приложениях где используется криптосистема RSA, в системах обработки изображений и других областях, где необходимы высокопроизводительные вычисления.

Табл. 2. Результат этапа Б (p[n] - длина набора модулей, где n - количество модулей в наборе)
Table 2. Result of stage B (p[n] is the length of the set of modules, where n is the number of modules in the set)

p[n]	Итерационное деление	Модифицированное итерационное деление
3	0.0000728	0.0000701
4	0.0001046	0.0001022
5	0.0001251	0.0001045
6	0.0001452	0.0001413
7	0.0001586	0.0001459
8	0.0001663	0.0001621
9	0.0002076	0.0001920

Список литературы / References

[1]. Mohan P. V. A., Mohan P. V. A. Residue Number Systems. Cham, Switzerland: Birkhäuser, 2016. – pp. 16-24.

[2]. Chervyakov N., Babenko M., Tchernykh A., Kucherov N., Miranda-López V., Cortés-Mendoza J. M. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. Future Generation Computer Systems, vol. 92, 2019, pp. 1080-1092.

[3]. Kasianchuk, M. M., Yakymenko, I. Z., Nykolaychuk, Y. M. (2021). Symmetric cryptoalgorithms in the residue number system. Cybernetics and Systems Analysis, 57(2), 2021, pp. 329-336.

[4]. Tchernykh, A., Schwiegelsohn, U., Talbi, E. G., Babenko, M. (2019). Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. Journal of Computational Science, vol. 36, 2019, P. 100581.

[5]. Червяков, Н. И., Коляда, А. А., Ляхов, П. А., Бабенко, М. Г., Лавриненко, И. Н., Лавриненко, А. В. Модулярная арифметика и ее приложения в инфокоммуникационных технологиях. М.: ФИЗМАТЛИТ, 2017, 440 с. / Chervyakov, N. I., Kolyada, A. A., Lyakhov, P. A., Babenko, M. G., Lavrinenko, I. N., Lavrinenko, A. V. Modular arithmetic and its applications in info-communication technologies. Moscow: Fizmatlit, 2017, 440 p. (in Russian).

[6]. Molahosseini A. S., Sorouri S., Zarandi A. A. E. Research challenges in next-generation residue number system architectures. Computer Science & Education (ICCSE), 7th International Conference, 2012, pp. 1658–1661.

[7]. Червяков Н.И., Ляхов П.А., Оразаев А.Р. Компьютерная оптика, том. 42, вып. 4, 2018 г., стр. 667-678: DOI: 10.18287/2412-6179-2018-42-4-667-678./ Chervyakov N.I., Lyakhov P.A., Orazaev A.R. Computer Optics, 2018, vol. 42, issue 4, pp. 667-678 (in Russian). DOI: 10.18287/2412-6179-2018-42-4-667-678.

[8]. Червяков Н.И., Лавриненко И. Н. Модулярные методы и алгоритмы деления на основе спуска Ферма и итераций Ньютона. Инфокоммуникационные технологии, том. 7, вып. 4, 2009 г., стр. 9–12. / Chervyakov N.I., Lavrinenko I.N. Modular methods and algorithms of division based on Fermat's descent and Newton's iterations. Info-communication Technologies, 2009, vol. 7, issue 4, pp. 9-12 (in Russian).

[9]. Червяков Н.И. Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов. Инфокоммуникационные технологии, том. 9, вып. 4, 2011 г., стр. 4–12. / Cherviakov N.I. Methods, algorithms and technical implementation of the main problem operations performed in the residual class system. Info-communication Technologies, 2011, vol. 9, issue 4, pp. 4-12 (in Russian).

[10]. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. М., Советское радио, 1968, 440 с. / Akushsky I. Ya., Yuditsky D. I. Computer arithmetic in residual classes. Moscow, Soviet Radio, 1968, 440 p. (in Russian).

[11]. Акушский И. Я., Бурцев В. М., Пак И. Т. О новой позиционной характеристике непозиционного кода и ее приложении. Теория кодирования и оптимизация сложных систем. Алма-Ата, Наука, КазССР, 1977, стр. 8–16. / Akushsky, I.Y., Burtsev, V.M., Pak, I.T. (1977) About the New Positional Characteristic of the Non-Positional Code and Its Application. In Theory of Coding and Optimization of Complex Systems, Alma-Ata, Nauka, KazSSR, 1977, pp. 8–16 (in Russian).

[12]. Shiriaev, E., Kucherov, N., Babenko, M., Lutsenko, V., Al-Galda, S. Algorithm for Determining the Optimal Weights for the Akushsky Core Function with an Approximate Rank. Applied Sciences, 13(18), 2023, 10495. https://doi.org/10.3390/app131810495.

[13]. Акушский И. Я., Бурцев В. М., Пак И. Т. Алгоритмы деления с использованием ядерной характеристики. Теория кодирования и оптимизация сложных систем. Алма-Ата, Наука, КазССР, 1977, стр. 26–33. / Akushsky, I.Y., Burtsev, V.M., Pak, I.T. (1977) Division Algorithms Using Core Characteristics. In Theory of Coding and Optimization of Complex Systems, Alma-Ata, Nauka, KazSSR, 1977, pp. 26–33 (in Russian).

[14]. Diffie W., Hellman M. E. New Directions in Cryptography. IEEE Transactions on Information Theory, 22(18), pp. 644–654. doi:10.1109/TIT.1976.1055638

Информация об авторах / Information about authors

Владислав Вячеславович ЛУЦЕНКО – аспирант, кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: высокопроизводительные вычисления, система остаточных классов, умный город, нейронные сети, интернет вещей.

Vladislav Vyacheslavovich LUTSENKO – postgraduate student, Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. Research interests: high-performance computing, residue number system, smart city, neural networks, Internet of Things.

Михаил Григорьевич БАБЕНКО – доктор физико-математических наук, заведующий кафедры вычислительной математики и кибернетики факультета математики и компьютерных наук имени профессора Н.И. Червякова ФГАОУ ВПО «Северо-Кавказский федеральный университет». Сфера научных интересов: облачные вычисления, высокопроизводительные вычисления, система остаточных классов, нейронные сети, криптография.

Mikhail Grigoryevich BABENKO – Dr. Sci (Phys.-Math.), Head of the Department of Computational Mathematics and Cybernetics, Faculty of Mathematics and Computer Science named after Professor N.I. Chervyakov, North Caucasus Federal University. His research interests include cloud computing, high-performance computing, residue number systems, neural networks, cryptography.

Андрей Николаевич ЧЕРНЫХ получил степень кандидата наук в Институте точной механики и вычислительной техники РАН. В настоящее время он является профессором Центра научных исследований и высшего образования в Энсенаде, Нижняя Калифорния, Мексика. В научном плане его интересуют многоцелевая оптимизация распределения ресурсов в облачной среде, проблемы безопасности, планирования, эвристики и метаэвристики, энергосберегающие алгоритмы, интернет вещей.

Andrei TCHERNYKH received his Cand. Sci. (Phys.-Math.) degree at the Institute of Precise Mechanics and Computer Engineering of the Russian Academy of Sciences. Now he is holding a full professor position in computer science at CICESE Research Center, Ensenada, Baja California, Mexico. He is interesting in grid and cloud research addressing multi-objective resource optimization, both, theoretical and experimental, security, uncertainty, scheduling, heuristics and meta-heuristics, adaptive resource allocation, energy-aware algorithms and Internet of Things.

Мария Анатольевна ЛАПИНА – кандидат физико-математических наук, доцент, доцент кафедры информационной безопасности автоматизированных систем ФГАОУ ВО «Северо-Кавказский федеральный университет». Сфера научных интересов: цифровые технологии, управление информационной безопасностью, процессный подход, образовательный процесс, криптография.

Maria Anatolievna LAPINA – Cand. Sci. (Phys.-Math.), Associate Professor, Associate Professor of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: digital technologies, information security management, process approach, educational process, cryptography.

DOI: 10.15514/ISPRAS-2023-35(5)-12

CC BY

Проверка программ на соответствие стандарту MISRA C с использованием инфраструктуры Clang

[1] Р.А. Бучацкий, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>
[1,2] Я.А. Чуркин, ORCID: 0009-0000-5044-4249 <yan@ispras.ru>
[1] К.А. Чибисов, ORCID: 0009-0008-4371-5475 <chibisov@ispras.ru>
[1] М.В. Пантилимонов, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>
[1,3] Е.В. Долгодворов, ORCID: 0000-0001-6962-6448 <krym4s@ispras.ru>
[1,3] А.В. Вязовцев, ORCID: 0009-0007-0826-2186 <andrey.vyazovtsev@ispras.ru>
[1] А.Г. Волохов, ORCID: 0009-0003-7797-4508 <alexeyvoh@ispras.ru>
[1,3] В.В. Трунов, ORCID: 0009-0001-9457-2610 <vtrunov@ispras.ru>
[4] Г.О. Миракян, ORCID: 0009-0001-2028-3496 <mirakyan.gayane@student.rau.am>
[1,3] К.Н. Китаев, ORCID: 0009-0004-9469-8100 <kitaev.konstantin@ispras.ru>
[1,2] А.А. Белеванцев, ORCID: 0000-0003-2817-0397 <abel@ispras.ru>

[1] Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[2] Московский государственный университет имени М.В. Ломоносова,
119991, Россия, Москва, Ленинские горы, д. 1.
[3] Московский физико-технический институт,
141701, Московская область, г. Долгопрудный, Институтский переулок, д. 9.
[4] Российско-Армянский (Славянский) университет,
РА, г. Ереван, 0051, ул. О.Эмина 123.

Аннотация. MISRA C – это сборник правил и рекомендаций по программированию на языке C, который является фактическим стандартом в отраслях, где безопасность играет ключевую роль. Стандарт разработан консорциумом MISRA (Motor Industry Software Reliability Association) и включает в себя набор рекомендаций, которые позволяют использовать язык C для разработки безопасного, надежного и переносимого программного обеспечения. MISRA широко применяется во многих отраслях с высокими требованиями к надежности, включая аэрокосмическую, оборонную, автомобильную и медицинскую.

Мы разработали статические детекторы для проверки кода на соответствие рекомендациям стандарта безопасного кодирования MISRA C 2012. Средство проверки кода основано на компиляторной инфраструктуре LLVM/clang. В данной статье описываются стратегии, лежащие в основе проектирования и реализации детекторов. На тестовых примерах MISRA C предложенные детекторы с высокой точностью определяют соответствие или нарушение рекомендациям. Также детекторы показывают большее покрытие и лучшую скорость работы, чем Cppcheck, популярный статический анализатор с открытым исходным кодом.

Ключевые слова: MISRA; статический анализ; символьное выполнение; LLVM; Clang; Clang-Tidy; статический анализатор Clang.

Для цитирования: Бучацкий Р.А., Чуркин Я.А., Чибисов К.А., Пантилимонов М.В., Долгодворов Е.В., Вязовцев А.В., Волохов А.Г., Трунов В.В., Миракян Г.О., Китаев К.Н., Белеванцев А.А. Проверка программ на соответствие стандарту MISRA C с использованием инфраструктуры Clang. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 169–192. DOI: 10.15514/ISPRAS–2023–35(5)–12.

Checking Programs for Compliance with MISRA C Standard Using the Clang Framework

[1] *R.A. Buchatskiy, ORCID: 0000-0001-8522-1811 <ruben@ispras.ru>*
[1,2] *Y.A. Churkin, ORCID: 0009-0000-5044-4249 <yan@ispras.ru>*
[1] *K.A. Chibisov, ORCID: 0009-0008-4371-5475 <chibisov@ispras.ru>*
[1] *M.V. Pantilimonov, ORCID: 0000-0003-2277-7155 <pantlimon@ispras.ru>*
[1,3] *E.V. Dolgodvorov, ORCID: 0000-0001-6962-6448 <krym4s@ispras.ru>*
[1,3] *A.V. Vyazovtsev, ORCID: 0009-0007-0826-2186 <andrey.vyazovtsev@ispras.ru>*
[1] *A.G. Volokhov, ORCID: 0009-0003-7797-4508 <alexeyvoh@ispras.ru>*
[1,3] *V.V. Trunov, ORCID: 0009-0001-9457-2610 <vtrunov@ispras.ru>*
[4] *G.H. Mirakyan, ORCID: 0009-0001-2028-3496 <mirakyan.gayane@student.rau.am>*
[1,3] *K.N. Kitaev, ORCID: 0009-0004-9469-8100 <kitaev.konstantin@ispras.ru>*
[1,2] *A.A. Belevantsev, ORCID: 0000-0003-2817-0397 <abel@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the RAS,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.
[3] *Moscow Institute of Physics and Technology,*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.
[4] *Russian-Armenian (Slavonic) University,*
0051, Republic of Armenia, Yerevan, Hovsep Emin str. 123.

Abstract. MISRA C is a collection of rules and recommendations for C programming language that is the de facto standard in industries where security plays the key role. The standard was developed by the MISRA (Motor Industry Software Reliability Association) consortium and includes a set of recommendations that allow the C language to be used to develop safe, reliable and portable software. MISRA is widely used in many industries with high reliability requirements, including aerospace, defense, automotive and medical.

We have developed static checkers to check code for compliance with MISRA C 2012 secure coding standard. The developed checkers are based on the LLVM/clang compiler infrastructure. This paper describes the strategies underlying the design and implementation of checkers. Using MISRA C 2012 example suite, the proposed checkers determine compliance or violation of the recommendations with high accuracy. The checkers also show greater coverage and better performance than Cppcheck, a popular open-source static analyzer.

Keywords: MISRA; static analysis; symbolic execution; LLVM; Clang; Clang-Tidy; Clang static analyzer.

For citation: Buchatskiy R.A., Churkin Y.A., Chibisov K.A., Pantilimonov M.V., Dolgodvorov E.V., Vyazovtsev A.V., Volokhov A.G., Trunov V.V., Mirakyan G.H., Kitaev K.N., Belevantsev A.A. Checking programs for compliance with MISRA C standard using the Clang framework. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 169-192 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-12.

1. Введение

Программное обеспечение (ПО) давно прошло стадию применения исключительно в исследовательских целях и заняло ключевую роль в современном мире. По мере увеличения размера и сложности разрабатываемого кода, вопросы надежности и безопасности стали важными проблемами при разработке ПО. Как следствие, проверка программ на отсутствие ошибок и уязвимостей, которые могут привести к отказу систем и даже человеческим жертвам, стала более сложной.

Язык `C` часто применяется при разработке ПО в критических системах (например, в контроллерах реактивных двигателей и медицинских системах). Несмотря на популярность он имеет плохую репутацию в контексте безопасного кодирования:

- синтаксис языка C предрасположен к совершению ошибок программистом, например, использование оператора присваивания (=) вместо оператора проверки на равенство (==) в конструкциях `if` и `while`;
- философия языка C предполагает, что программисты знают, что они делают: любая совершенная ошибка в коде может остаться незамеченной. Область, в которой C особенно слаб в этом отношении – это проверка типов. Например, является допустимой запись числа с плавающей запятой в целое, которое используется для представления значений `true` или `false`.

Использование языка `C` в системах, связанных с безопасностью, требует значительной осторожности. С целью обеспечения контроля над языками `C/C++` организации начали разрабатывать и принимать стандарты безопасного кодирования, например, "Motor Industry Software Reliability Association C" (MISRA C) [1], его дальнейшие редакции MISRA C 2004, MISRA C++ 2008, MISRA C 2012, "Computer Emergency Response Team C" (CERT C) [2], "AUTomotive Open System ARchitecture" (AUTOSAR) [3]. Эти стандарты содержат правила и рекомендации по разработке безопасного, защищенного и надежного программного обеспечения и стали обязательными в соответствующих областях.

Для проверки соответствия кода безопасным стандартам кодирования применяются различные техники, из них выделим:

- проверка кода вручную: опытные разработчики или специалисты по безопасности просматривают исходный код, чтобы выявить потенциальные проблемы безопасности или нарушения стандартов безопасного кодирования. Этот процесс может занять много времени, но он эффективен для выявления многих типов уязвимостей;
- автоматический анализ кода: анализ с применением средств статического и/или динамического анализа. Автоматический анализ признан более эффективным, чем ручной, хотя бы потому, что он легко интегрируется в процесс разработки;
- аудиты соответствия: проводятся сторонними аудиторами или внутренними группами безопасности, чтобы гарантировать, что программное обеспечение разработано в соответствии со стандартами безопасного кодирования. Аудиты включают проверку документации, политик, процедур и кода для выявления любых несоответствий.

Данная работа посвящена разработке статических детекторов кода для программ на языке `C` для проверки их соответствия безопасному стандарту кодирования MISRA C 2012. Детекторы реализованы на основе компиляторной инфраструктуры LLVM [4] с использованием инструментов Clang-Tidy [5] и Clang Static Analyzer (CSA) [6].

Целью данной работы является реализация статических детекторов, проверяющих код на языке `C` на соответствие стандарту MISRA C 2012. Структура работы включает в себя обзор существующих решений (раздел 2), анализ стандарта MISRA C 2012 (раздел 3), разбор функциональных возможностей Clang-Tidy и CSA (раздел 4), примеры реализованных детекторов и трудностей, с которыми пришлось столкнуться при их разработке (раздел 5), тестирование (раздел 6) и заключение.

2. Обзор существующих решений

В настоящее время для проверки программ на соответствие стандартам безопасного кодирования в большинстве своем используются статические анализаторы, предоставляемые по коммерческим лицензиям, но есть небольшая часть с открытым исходным кодом.

В данном разделе приведено краткое описание существующих популярных открытых и коммерческих решений, приведена сводная таблица, отражающая поддержку стандартов безопасного кодирования в рассмотренных анализаторах, а также приведено сравнение анализаторов с открытым исходным кодом.

2.1 Анализаторы с открытым исходным кодом

Clang-Tidy и **CSA** – статические анализаторы с открытым исходном кодом, разрабатываемые как часть проекта LLVM [4] для анализа языков C/C++/Objective-C.

Clang-Tidy делает анализ на уровне абстрактного синтаксического дерева (AST) [7] программы и на уровне препроцессора, используя для этого инфраструктуру компилятора Clang. Clang-Tidy имеет поддержку множества стандартов кодирования (cert, cppcoreguidelines, hicpp и т.п.) и активную поддержку сообщества, которая добавляет поддержку как общепринятых стандартов кодирования, так и популярные практики написания кода.

CSA в настоящее время используется как дополнение к Clang-Tidy и запускается с использованием его инфраструктуры. Для анализа кода CSA использует символьное исполнение, что позволяет обнаружить ошибки в программах классов: "*деление на ноль*", "*использование ресурсов после освобождения*", "*недостижимый код*" и т.д.

Cppcheck [8] – статический анализатор с открытым исходным кодом для языков C/C++. Cppcheck очень популярная утилита из-за своей простоты и доступности. Несмотря на это, безопасные стандарты кодирования поддерживаются в нем либо по коммерческой лицензии, либо как расширение для исходного анализатора на языке Python, что негативно сказывается на качестве анализа и производительности данного анализатора.

SonarQube [9] – открытая платформа для анализа исходного кода на языках C/C++, C#, JavaScript, Python, Ruby, PHP, Swift, Ruby. Данный анализатор написан на языке Java и в первую очередь интегрируется с экосистемой данного языка. Данный анализатор имеет частичную поддержку стандартов MISRA C/C++ которые доступны только по коммерческой лицензии.

2.2 Коммерческие анализаторы

Coverity [10] – коммерческий анализатор, используемый для поиска уязвимостей в коде на языках C, C++, C#, Java, JavaScript, TypeScript, и Objective-C. Данный анализатор имеет интеграцию с интегрированными средами разработки (IDE) такими как Visual Studio, Intellij IDEA и Eclipse. Он покрывает большинство стандартов безопасного кодирования.

Klocwork [11] – коммерческий анализатор, используемый в первую очередь для поиска уязвимостей в исходном коде на языках C, C++, C#, Java, JavaScript и Python. В нем реализовано большинство стандартов безопасного кодирования.

PVS-Studio [12] – коммерческий анализатор, используемый в первую очередь для поиска уязвимостей в исходном коде на языках C, C++, C# и Java. Данный анализатор может использоваться некоммерческими проектами с открытым исходным кодом без лицензионного сбора. Он полностью поддерживает основные стандарты безопасного кодирования такие как MISRA C 2012 и AUTOSAR C++14.

2.3 Сравнение анализаторов

В табл. 1 представлено сравнение статических анализаторов по критерию поддержки стандартов безопасного кодирования. Из таблицы видно, что большинство анализаторов, поддерживающих стандарты безопасного кодирования, предоставляются по коммерческим лицензиям, что, в свою очередь, накладывает ограничения на их использование, не предоставляет легкой возможности для оценки качества реализованных детекторов независимыми специалистами по безопасности, а также расширения функционала детекторов и настройки под конкретный проект.

*Табл. 1. Поддержка стандартов безопасного кодирования в анализаторах C/C++ кода (символом * отмечены анализаторы с открытым исходном кодом, символом $ поддержка по коммерческой лицензии)*
*Table 1. C/C++ coding standards support in code analyzers (the * symbol indicates open source analyzers, the $ symbol indicates support under a commercial license)*

Стандарт / Анализатор	MISRA C++ 2008	MISRA C 2012	CERT	AUTOSAR
Coverity	Да	Да	Да	Да
Klocwork	Да	Да	Нет	Да
PVS-Studio	Частичная	Да	Да	Да
SonarQube*	Частичная$	Частичная$	Нет	Нет
Cppcheck*	Нет	Частичная	Частичная	Частичная$
Clang-Tidy / CSA*	Нет	Нет	Да	Нет

Статические анализаторы с открытым исходным кодом как правило предоставляют поддержку стандартов безопасного кодирования по коммерческим лицензиям. Так, например, анализатор SonarQube не имеет поддержки никаких безопасных стандартов кодирования в своей открытой версии, что сильно замедляет написание стандартов кодирования с нуля, так как сокращается возможность переиспользования кода в нем.

Cppcheck, который имеет частичную поддержку рекомендаций MISRA C 2012 и CERT C, использует для анализа AST и лексический анализ, чего в свою очередь недостаточно для проверки правил, требующих моделирование выполнения программы и оценки значения выражений. Также данный анализатор использует расширения на языке Python, что негативно сказывается на производительности анализа при большом количестве детекторов.

Clang-Tidy и CSA несмотря на то, что не имеют поддержки MISRA в отличие от Cppcheck, имеют активную поддержку сообщества, более мощные инструменты анализа, символьное исполнение и анализ на уровне AST, также реализуют множество стандартов кодирования, которые в некоторых местах брали за основу стандарты MISRA C/C++.

В данной работе за основу реализации детекторов для рекомендаций MISRA C 2012 были выбраны инструменты Clang-Tidy и CSA из компиляторной инфраструктуры LLVM.

3. Стандарт безопасного кодирования MISRA

Первая редакция стандарта MISRA C была опубликована в 1998 году [13] с целью удовлетворить потребность, возникшую в автомобильной промышленности, и отраженную в рекомендациях MISRA по разработке программного обеспечения для транспортных средств [14]. Первая редакция имела большой успех и стала применяться за пределами автомобильного сектора. Более широкое использование стандарта в отрасли и сообщаемый опыт разработчиков ПО побудили к серьезной переработке MISRA C, в результате которой в 2004 году было опубликовано второе издание.

В 2013 году было опубликовано третье издание – MISRA C 2012. В MISRA C 2012 внесены многочисленные улучшения по сравнению с предыдущим изданием: расширена поддержка стандарта языка C; в дополнение к C90 добавлена поддержка C99; рекомендации и правила были определены более четко и точно, и, как следствие, значительно возросла вероятность того, что разные инструменты статического анализа могут дать одинаковые результаты.

В настоящее время стандарт MISRA C 2012 включает ***175*** *рекомендаций* (с учетом расширений Amendment 1 [15] и 2 [16]). Каждая рекомендация классифицируется как *директива* или *правило*.

Правило – это рекомендация, для которой предоставлено полное описание требования. Должна быть возможность проверить соответствие исходного кода правилу без необходимости использования какой-либо дополнительной информации. Инструменты статического анализа должны быть способны проверять соответствие правилам.

Директива – это рекомендация, для которой невозможно дать полное описание, необходимое для проведения проверки на соответствие. Для проведения проверки необходима дополнительная информация, которая может быть предоставлена в проектной документации или спецификациях требований к исходному коду. Инструменты статического анализа могут помочь в проверке соблюдения директив, но разные инструменты могут по-разному интерпретировать то, что представляет собой несоответствие кода директиве.

Каждому правилу в свою очередь присвоена своя категория:

- *Обязательное* – правило должно выполняться без каких-либо исключений в любом коде, заявляющий соответствие данным стандартам безопасного кодирования.
- *Необходимое* – правило должно выполняться, но код может иметь задокументированное формальное отклонение.
- *Рекомендательное* – данное правило не обязательно для реализации, чтобы заявить соответствие кода безопасному стандарту кодирования.

При реализации инструментов статического анализа приоритет расставляется согласно значимости категорий: обязательные, затем необходимые, и лишь потом рекомендательные.

Стандарт MISRA C 2012 также предоставляет информацию о том, как должна осуществляться проверка на соответствие кода рекомендации: автоматически или полуавтоматически. Из этого следует, что рекомендации, которые проверяются полуавтоматически, должны предоставлять настройки своих эвристик.

Стоит отметить, что стандарт MISRA активно применяется многими международными компаниями в автомобильной индустрии: General Motors, Ford Motor Company, Volvo Cars, Bosch, Renault, BMW Group.

4. Статический анализ в LLVM

LLVM [4] – компиляторная инфраструктура с открытым исходным кодом, предоставляющая большое число переиспользуемых компонентов для разработки компиляторов. LLVM предоставляет широкие возможности для анализа программ, написанных на языках C/C++/Objective-C посредством компилятора Clang.

Clang содержит в себе множество C/C++ библиотек, позволяющих работать с исходным кодом на разных стадиях компиляции, среди них можно выделить следующие:

- `libclangLex` – библиотека для работы с лексическим анализатором, которая в свою очередь включает работу с препроцессором;
- `libclangAST` – библиотека для работы с AST;
- `libclangASTMatchers` – библиотека, содержащая подготовленные, часто используемые шаблоны для поиска в AST;

- `libclangAnalysis` – библиотека с часто используемыми утилитами для анализа кода на базе AST. В ней содержатся абстракции для работы с графом вызовов, базовый анализ достижимости кода, изменяемости переменных.

Данные библиотеки используются как фундамент для статических анализаторов кода на базе LLVM: Clang-Tidy и CSA.

4.1 Clang-Tidy

Clang-Tidy [5] – инструмент статического анализа, разработанный для проверки C/C++/Objective-C кода на соответствие различным стандартам кодирования и общепринятым практикам, детекторы которых сгруппированы как модуль `ClangTidyModule`.

`ClangTidyModule` – набор детекторов, имеющих общий класс, например *cert*. Детекторы не изолированы внутри модуля и могут наследовать реализации из других `ClangTidyModule`. Такое переиспользование возможно, так как каждый детектор должен реализовать интерфейс `ClangTidyCheck`, который представлен на листинге 1.

```
class ClangTidyCheck {
public:
  virtual void registerMatchers (ast_matchers::MatchFinder *Finder) {}
  virtual void check (const ast_matchers::MatchFinder::MatchResult &Result) {}
  virtual void registerPPCallbacks (const SourceManager &SM,
                                    Preprocessor *PP,
                                    Preprocessor *ModuleExpanderPP) {}
}
```

Листинг 1. Интерфейс класса ClangTidyCheck.
Listing 1. ClangTidyCheck class interface.

- `registerMatchers` – отвечает за регистрацию шаблонов поиска в AST, которые записывает в `Finder`;
- `check` – это обратный вызов, как результат обработки шаблонов из `registerMatchers`. Данный метод вызывается для каждого совпадения в AST и передает интересующие узлы через параметр `Result`;
- `registerPPCallbacks` – функция, аналогичная `registerMatchers`, но для препроцессора. Обратные вызовы добавляются через методы *PP*.

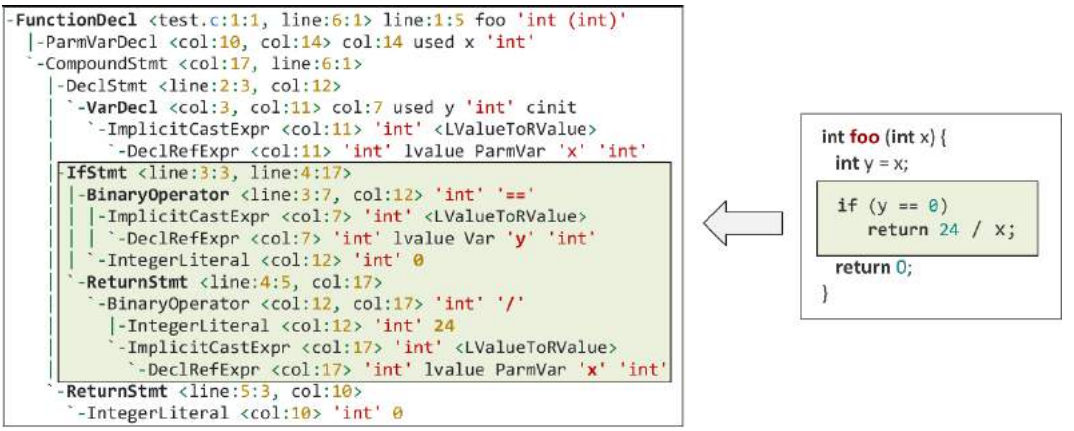
Детекторы для Clang-Tidy можно реализовать на уровне препроцессора с помощью `PPCallbacks` или на уровне AST с помощью шаблонов поиска (AST Matchers) [17].

Абстрактное синтаксическое дерево (AST) Clang [7] строится после обработки исходного кода препроцессором и представляет собой структуру, создаваемую фронтендом компилятора и служащую промежуточным представлением программы, используемым Clang. Генерация бинарного кода осуществляется на основе AST.

AST Clang содержит полную информацию об исходном коде программы: каждый узел дерева хранит местоположение в исходном коде до и после обработки препроцессором. Это делает AST пригодным для использования в качестве наиболее простой основы для анализа кода. На листинге 2 показан пример AST представления, полученного с помощью компилятора *clang*. Левая часть – это внутреннее представление дерева для соответствующего кода справа.

Шаблоны поиска AST в Clang-Tidy [17] – это интерфейс для декларативного определения участков кода, которые нужно найти, и выполнения определенных действий над каждым найденным шаблоном. В качестве примера шаблона поиска AST рассмотрим следующее определение: `ifStmt(has(binaryOperator(hasOperatorName("=="))))).bind("if").`

В данном примере выполняется поиск в AST оператора `"if"`, который содержит в условии бинарный оператор сравнения `"=="` (листинг 2). Команда сопоставления `bind(...)` присваивает имя узлу AST, к которому она применяется, для дальнейшего использования.



Листинг 2. Пример AST представления для кода на языке C.
Listing 2. AST representation of function written in C language.

В Clang-Tidy также доступен анализ на уровне графа потока управления (CFG), который построен «на уровне исходного кода» программы и состоит из указателей на операторы AST, упорядоченные в порядке потока управления (т. е. в порядке исполнения).

Существенным ограничением Clang-Tidy является невозможность одновременного анализа сразу нескольких единиц трансляции. В некоторых ситуациях информации только на уровне AST или только в рамках одной единицы трансляции бывает недостаточно.

4.2 Clang Static Analyzer

Clang Static Analyzer [6] – статический анализатор для проверки C/C++/Objective-C кода, использующий метод символьного исполнения. Этот метод предполагает присвоение символьных значений переменным программы и разбиение всех возможных состояний программы на классы. Для анализа используется структура данных под названием *разобранный граф* (далее РГ) (exploded graph) – это один из известных способов организации чувствительного к путям анализа. Анализатор интерпретирует все возможные пути исполнения в графе потока управления. Каждый путь в данном направленном графе – множество узлов вида `(ProgramPoint, ProgramState)`.

`ProgramPoint` – участок программы до и после оператора в потоке управления.

`ProgramState` – абстрактное состояние программы, содержащее текущие значения переменных, оценку значений для выражений, ограничения на значения неизвестных переменных, области памяти, отображающие содержимое переменных, а также нетипизированное хранилище данных – `GenericDataMap` (далее GDM), принадлежащее пользовательским детекторам, а также позволяющее хранить составные типы данных, необходимые для анализа, например, список указателей на одну и ту же память в детекторе, который анализирует алиасы и так далее. Состояния меняются между `ProgramPoint`.

Ребра в РГ между узлами означают, что `ProgramPoint` первого узла корректирует его `ProgramState` до `ProgramState` из второго узла и переводит программу в точку выполнения второго узла.

В CSA, также, как и Clang-Tidy анализ происходит в рамках одной единицы трансляции. В ней ищутся функции, являющиеся корневыми, то есть, теми, которые не вызываются в других

местах, в рамках этой единицы трансляции. Для каждой такой корневой функции инициализируется начальное состояние `ProgramState`, который, затем, в результате анализа операторов и выражений меняется. Анализатор симулирует выполнение условных выражений, таких как оператор `"if"`, путем создания новых узлов и ребер в разобранном графе для истинного и ложного результата с учетом текущего абстрактного состояния.

В CSA применяется кэширование узлов графа для снижения сложности его обхода. Если будет сгенерирован новый узел с тем же состоянием и программной точкой, что у существующего узла, путь кэшируется, и переиспользуется существующий узел.

Взаимодействие анализатора с конечными детекторами осуществляется посредством паттерна "Посетитель" (*Visitor*) и использования идиомы языка `C++` *Curiously Recurring Template Pattern* (CRTP) [18], часть интерфейса которого представлена на листинге 3.

```
class CSAChecker: public Checker<check::PreCall, check::PostCall,
                                 check::Location, check::Bind,
                                 check::BranchCondition, check::PointerEscape,
                                 check::EndAnalysis> {
public:
  void checkPreCall(const CallEvent &Call, CheckerContext &C) const {}
  void checkPostCall(const CallEvent &Call, CheckerContext &C) const {}
  void checkLocation(SVal Loc, bool IsLoad, const Stmt *S,
                     CheckerContext &) const {}
  void checkEndAnalysis(ExplodedGraph &G,
                        BugReporter &BR,
                        ExprEngine &Eng) const {}
}
```

Листинг 3. Пример интерфейса детектора CSA.
Listing 3. CSA checker interface example.

- `checkPreCall` – вызывается сразу перед входом в функцию;
- `checkPostCall` – вызывается сразу после выхода из функции;
- `checkLocation` – вызывается при каждом доступе области памяти `Loc`;
- `checkEndAnalysis` – вызывается при завершении анализа, результирующий граф находится в `G`.

Каждый детектор в процессе такого обхода может добавлять ребра в РГ посредством обновления состояния `ProgramState` в контексте анализатора (`CheckerContext`). Если детектор обнаруживает ошибку в коде, то сообщение о ней будет содержать путь анализатора до этой ошибки: какие ветки и условия он выбирал, ограничения на переменные, участвующие в пути, сколько итераций сделал в цикле, и т.д.

В анализаторе также реализована модель памяти, позволяющая запоминать конкретные и символьные значения определенных областей памяти (*memory regions*) и получать к ним доступ в любой момент анализа.

Частью CSA являются предсказатели (*assumers*), позволяющие накладывать ограничения на значения переменных в процессе символьного выполнения, например – `if ( i != 0 ) func(i);`, в данном примере из диапазона значений переменной `i` будет выколота точка `0`, в `ProgramPoint`, соответствующей точки выполнения после оператора `if ( i != 0 )`. Это позволяет реализовывать детекторы, например, проверяющие переполнение типа. Стоит отметить, что предсказатели действуют достаточно консервативно, например, если о символе ничего не известно, то диапазон его значений будет определен как полный диапазон значений типа. Анализатору можно подсказывать о диапазонах значений переменных с помощью определенных методов, например, `assumeSymNE` из модуля *RangeConstraintManager*, который предполагает неравенство символа определенному переданному значению. Методы

данного типа изменяют информацию о диапазонах значений символов в состоянии и возвращают обновленное состояние.

Также ядром CSA поддерживаются различные типы диагностик: упрощенная диагностика (`BasicBugReport` – только сама диагностика и подсказки); чувствительная к путям диагностика (`PathSensitiveBugReport` – сама диагностика, подсказки и путь от корневой функции к месту, в котором она была выдана).

CSA, используя анализ графа потока управления и символьное выполнение, эмулирует вызов вложенных функций. Это позволяет осуществлять межпроцедурный анализ, чего нельзя сказать про Clang-Tidy.

В заключение стоит отметить, что для анализа на уровне AST, не требующего межпроцедурность и символьное выполнение, Clang-Tidy является хорошим выбором, в силу простоты реализации детекторов и скорости анализа. В ином случае, стоит обратить внимание на CSA. Также стоит отметить, что помимо межпроцедурного, в CSA была добавлена поддержка межмодульного анализа, но на данный момент она плохо масштабируема на большие программы.

5 Разработка детекторов для MISRA C

За основу реализации детекторов для рекомендаций MISRA C 2012 была выбрана инфраструктура открытых анализаторов Clang-Tidy и CSA, работа каждого из которых была разобрана в разделе 4.

В рамках данной работы были реализованы детекторы для 139 рекомендаций MISRA C 2012 из которых 122 детектора для Clang-Tidy и 17 для CSA. Из оставшихся 36 рекомендаций 27 покрываются уже имеющимися в Clang детекторами или диагностиками (в том числе ошибками компиляции), 3 рекомендации явно требуют анализа в рамках множества единиц трансляций (что не возможно реализовать с помощью Clang), а 6 рекомендаций являются директивами и сильно зависят от структуры конкретного проекта. Совокупное покрытие стандарта MISRA C 2012 разработанными детекторами и уже имеющимися в Clang диагностиками составляет 95%.

В Clang-Tidy у каждого детектора есть уникальное имя. Детекторы для запуска можно выбрать с помощью опции `-checks=`, которая задает разделенный запятыми список включаемых и отключаемых (с префиксом `-`) детекторов. С помощью Clang-Tidy также можно запускать детекторы CSA. Ниже представлена команда для запуска реализованных детекторов Clang-Tidy и CSA для проверки кода на соответствие стандарту MISRA C 2012:

```
clang-tidy -checks="-*, misra-c-2012*, clang-analyzer-misra*" main.c
```

Здесь опция `-checks=` отключает все детекторы по умолчанию (`-*`), включает все детекторы Clang-Tidy (`misra-c-2012*`) и детекторы CSA (`clang-analyzer-misra*`).

В случае обнаружения детектором несоответствия кода стандарту MISRA C 2012 выдается предупреждение в виде диагностики с соответствующим сообщением и с указанием места в исходном коде.

5.1 Пример детекторов на Clang-Tidy

Рассмотрим примеры реализации детекторов для правил MISRA C 2012 с использованием инфраструктуры Clang-Tidy для которых будет достаточно поиска шаблонов на AST и лексического анализа. Большинство реализованных детекторов для правил MISRA C 2012 не встретило сложностей, пример такого детектора представлен в разделе 5.1.1. Из сложностей можно отметить аккуратную работу с шаблоном поиска `findAll`, которая может существенно снижать производительность на больших проектах. Для решения этой проблемы был предложен определенный подход, который описан в разделе 5.1.2.

5.1.1 Правило 15.2

Правило 15.2 из стандарта MISRA C 2012 звучит следующим образом:

> «Оператор *goto* должен переходить к метке, объявленной позже в той же функции»

На листинге 4 приведен пример кода на языке C с конструкциями как нарушающими данное правило, так и соответствующими ему:

```
void rule15_2() {
    int J = 0;
L1:
    ++J;
    if (10 == J) {
        goto L2; // Нарушения нет
    }
    goto L1;     // Нарушение - переход к метке, объявленной до оператора goto.
L2:
    ++J;

L3: goto L3;   // Нарушение - переход к метке, объявленной до оператора goto.
}
```

Листинг 4. Иллюстрация различных аспектов правила MISRA C 2012 15.2.
Listing 4. Code markup according to the MISRA C 2012 Rule 15.2.

Рассмотрим возможную реализацию детектора для данного правила с использованием Clang-Tidy. Для реализации данного правила достаточно анализа AST представления для поиска узлов оператора «goto».

В разделе 4.1 был описан интерфейс детекторов Clang-Tidy. Реализация методов `check` и `registerMatchers` детектора для данного правила представлена на листинге 5:

```
1   void LabelAfterGotoCheck::registerMatchers(MatchFinder *Finder) {
2     Finder->addMatcher(gotoStmt().bind("goto"), this);
3   }
4
5   void LabelAfterGotoCheck::check(const MatchFinder::MatchResult &Result) {
6     const auto *MatchedGoto = Result.Nodes.getNodeAs<GotoStmt>("goto");
7     const auto &SM = Result.SourceManager;
8
9     SourceLocation GotoLoc = MatchedGoto->getBeginLoc();
10    SourceLocation LabelLoc = MatchedGoto->getLabel()->getBeginLoc();
11    // Check that `goto` location is presumed before `label` location
12    if (SM->getFileLoc(GotoLoc) <= SM->getFileLoc(LabelLoc))
13      return;
14
15    diag(GotoLoc, "Оператор goto должен переходить к метке, "
16                  "объявленной позже в той же функции");
17    diag(LabelLoc, "Метка объявлена здесь", DiagnosticIDs::Note);
18  }
```

Листинг 5. Возможная реализация детектора для правила MISRA C 2012 15.2 в Clang Tidy.
Listing 5. One of the possible MISRA C 2012 Rule 15.2 checker implementations in Clang Tidy.

Функция `registerMatchers` регистрирует шаблон поиска `gotoStmt`. Функция `check` получает шаблон, найденный в AST дереве в виде экземпляра класса `GotoStmt`, находит смещение оператора `goto` в файле и смещение соответствующей метки (строки 9-10) и сравнивает их (строка 12). В случае, если смещение метки находится до смещения оператора `goto`, то выдается предупреждение (строки 15-17).

5.1.2 Правило 8.9

Далее приведем пример правила с более сложной реализацией детектора. Рассмотрим правило 8.9 из стандарта MISRA C 2012, которое звучит следующим образом:

> «Объект должен быть объявлен в теле функции, если его использование встречается только в одной функции»

Возможный вариант кода, нарушающего правило 8.9, представлен на листинге 6:

```
int X; // Нарушение - переменная используется только в функции 'foo'
extern char Y; // Нарушение - переменная используется только в функции 'foo'

static int Z; // Нарушения нет - переменная используется как в функциях 'foo' и 'bar'
char A; // Нарушения нет - переменная не используется ни в одной функции

void foo() {
  int I = 0; // Нарушения нет - переменная используется только в функции 'foo'
             // и определена в ней же
  for (I = 0; I < X; ++I, ++Z, ++Y) {
  ...
  }
}

void bar() {
  Z++;
}
```

Листинг 6. Иллюстрация различных аспектов правила MISRA C 2012 8.9.
Listing 6. Code markup according to the MISRA C 2012 Rule 8.9.

Первоначальная реализация детектор для этого правила представлена на листинге 7.
В “наивной реализации” для каждого глобального объявления переменной (`VarDecl`) ищутся все ее использования (`DeclRefExpr`) во всей единице трансляции (параметр `*Result.Context->getTranslationUnitDecl()`) с помощью функции `match`, с использованием шаблона поиска `findAll` (строки 9-14), который ищет все использования вложенного в него шаблона поиска. Такая конструкция может существенно снижать производительность на проектах с большим количеством глобальных переменных. Каждое использование рассматривается вместе с функцией, в которой оно находится. Далее производится подсчет количества функций, в которых используется рассматриваемая глобальная переменная, если такая функция всего одна, то выдается предупреждение.
При тестировании разработанного детектора на реальных проектах был выявлен ряд проблем, связанных с производительностью на больших единицах трансляции и ошибок в диагностике глобальных переменных, использование которых встречается вне функций. Данный аспект не был учтен при первоначальном проектировании детектора.
Обновленная реализация детектора представлена на листинге 8.
В обновленной реализации отбор всех функций, в которых наблюдается использование глобальных переменных, теперь происходит при регистрации шаблонов поиска (метод `registerMatchers`, строки 7-10). Был использован шаблон `forEachDescendant`, необходимый для связывания *“родительского объявления”* с объявлением каждой глобальной переменной, которая в его определении была использована. Соответственно метод `check` будет вызываться для каждой пары *“родительское объявление - глобальная переменная”*. Таким родительским объявлением может быть объявление глобальной переменной с инициализатором или определение функции. В методе `check` происходит лишь сбор информации в словарь `VDToFDs`, представляющий собой отображение определений глобальных переменных в множество определений функций, в которых они были использованы. Проверка количества использований глобальных переменных в функциях и выдача предупреждающей была перенесена в конец обработки единицы трансляции (метод

`onEndOfTranslationUnit`, строки 33-50), когда информация обо всех глобальных переменных и функциях была собрана.

```
1  void ObjectBlockScopeCheck::registerMatchers(MatchFinder *Finder) {
2    Finder->addMatcher(varDecl(unless(common::isLocal())).bind("vd"), this);
3  }
4
5  void ObjectBlockScopeCheck::check(const MatchFinder::MatchResult &Result) {
6    const auto *MatchedDecl = Result.Nodes.getNodeAs<VarDecl>("vd");
7
8    // Find all `DeclRefExpr`s related to matched variable and its parent functions
9    const auto DREs = match(
10       findAll(functionDecl(hasDescendant(
11                                declRefExpr(to(varDecl(equalsNode(MatchedDecl))))
12                                    .bind("dre")))
13                   .bind("parent-function")),
14       *Result.Context->getTranslationUnitDecl(), *Result.Context);
15   // If there is no variable usage then it is compliant case
16   if (DREs.empty())
17     return;
18
19   // To store first `FunctionDecl` to compare it with another
20   const FunctionDecl *FirstFD = nullptr;
21   // To store at least one `DeclRefExpr` in each function where
22   // variable usage is present
23   llvm::DenseMap<const FunctionDecl *, SourceLocation> DRELocs;
24
25   for (const auto &DRE : DREs) {
26     const auto *FD = DRE.getNodeAs<FunctionDecl>("parent-function");
27     const auto *D = DRE.getNodeAs<DeclRefExpr>("dre");
28
29     if (!DRELocs.count(FD))
30       DRELocs[FD] = D->getExprLoc();
31
32     if (!FirstFD)
33       FirstFD = FD;
34     // If there is more than one function with matched variable usage
35     // then it is compliant case
36     else if (FirstFD != FD)
37       return;
38   }
39
40   diag(MatchedDecl->getLocation(),
41        "Объект должен быть объявлен в теле функции, если его "
42        "использование  встречается только в одной функции");
43   for (const auto &DRLoc : DRELocs) {
44     diag(DRLoc.second, "`%0` используется только в функции `%1`",
45          DiagnosticIDs::Note)
46         << MatchedDecl->getNameAsString() << DRLoc.first->getNameAsString();
47   }
48 }
```

Листинг 7. Первоначальная реализация детектора для правила MISRA C 2012 8.9 в Clang Tidy.
Listing 7. Initial implementation of checker for MISRA C 2012 Rule 8.9 in Clang Tidy.

Если глобальные переменные используются вне функций (например, в определении других глобальных переменных), то их необходимо исключить из списка рассматриваемых детектором, так как перенесение определения глобальной переменной в данном случае в тело функции влечет за собой ошибки компиляции и изменения семантики исходного кода. Этот аспект, как было указано выше, не был учтен в первом варианте реализации, соответственно были ложные срабатывания детектора. В обновленной реализации была добавлена проверка того, что сопоставленное *“родительское объявление”* не является функцией. В таком случае происходит добавление определения данной глобальной переменной в список `BlackListedVDs` (строка 20), чтобы исключить данную глобальную переменную из списка потенциальных кандидатов для выдачи предупреждений.

```
1  void ObjectBlockScopeCheck::registerMatchers(MatchFinder *Finder) {
2    const auto VD = varDecl(unless(common::isLocal()), unless(parmVarDecl()),
3                            unless(isImplicit()))
4                       .bind("vd");
5
6    // Match all `Decl`s that have a global scope and contains references to globals
7    Finder->addMatcher(
8        decl(unless(isImplicit()), unless(hasAncestor(functionDecl())),
9             forEachDescendant(declRefExpr(to(VD)))).bind("decl"),
10       this);
11 }
12
13 void ObjectBlockScopeCheck::check(const MatchFinder::MatchResult &Result) {
14   const auto *VD = Result.Nodes.getNodeAs<VarDecl>("vd");
15   const auto *FD = Result.Nodes.getNodeAs<FunctionDecl>("decl");
16
17   // If `VarDecl` was used not at function add it at blacklist
18   // to not check it at end of translation unit
19   if (!FD) {
20     BlackListedVDs.insert(VD);
21     return;
22   }
23
24   // Store `VarDecl` and it's `FunctionDecl` where it was used
25   if (!BlackListedVDs.count(VD)) {
26     if (VDToFDs.count(VD))
27       VDToFDs[VD].insert(FD);
28     else
29       VDToFDs[VD] = {FD};
30   }
31 }
32
33 void ObjectBlockScopeCheck::onEndOfTranslationUnit() {
34   for (const auto &VDToFD : VDToFDs) {
35     const auto *VD = VDToFD.first;
36     const auto FDs = VDToFD.second;
37
38     // Skip if variable is in blacklist or there is more then one function
39     // where variable is used
40     if (BlackListedVDs.count(VD) || FDs.size() > 1)
41       continue;
42
43     diag(VD->getLocation(),
44          "Объект должен быть объявлен в теле функции, если его "
45          "использование  встречается только в одной функции");
46     const auto *SingleFD = *FDs.begin();
47     diag(SingleFD->getLocation(), "`%0` используется только в функции `%1`",
48          DiagnosticIDs::Note) << VD;
49   }
50 }
```

Листинг 8. Итоговая реализация детектора для правила MISRA C 2012 8.9 в Clang Tidy.
Listing 8. Final implementation of checker for MISRA C 2012 Rule 8.9 in Clang Tidy.

5.2 Пример детектора CSA

В данном разделе будут рассмотрены примеры реализации детекторов с использованием инфраструктуры Clang Static Analyzer для правил MISRA C 2012 для которых требуется символьное исполнение. В разделе 5.2.1 приводится пример реализации детектора с использованием анализа регионов памяти. В разделе 5.2.2 приводится пример детектора, при реализации которого использовался анализ помеченных данных, доступный в CSA.

5.2.1 Правило 18.3

Для иллюстрации реализации простого CSA детектора рассмотрим правило 18.3 из стандарта MISRA C 2012, которое звучит следующим образом:

«Реляционные операторы (>, >=, <, <=) не должны применяться к указателям на объекты, кроме тех случаев, когда они указывают на один и тот же объект.

Возможный вариант кода, нарушающего правило 18.3, представлен на листинге 9.

```
void rule18_3 ( int32_t param ) {
  int32_t a[10], b[10];
  int32_t *p;
  if ( param ) {
    p = a;
    if (p < b) {  // Нарушение - указатели относятся к разным объектам
      …
    }
  } else {
    p = b;
    if (p < b) {  // Нарушения нет - указатель p указывает на ту же память, что и b
      …
    }
  }
}
```

Листинг 9 Иллюстрация различных аспектов правила MISRA C 2012 18.3.
Listing 9. Code markup according to the MISRA C 2012 Rule 18.3.

Из примеров видно, что необходим анализ потока данных, недоступный в Clang-Tidy.
Реализация детектора для правила 18.3 с использованием CSA представлена на листинге 10.

```
1  void PointerRelationsChecker::checkPreStmt(const BinaryOperator *BO,
2                                             CheckerContext &C) const {
3    if (!BO->isRelationalOp())
4      return;
5
6    QualType LHSType = BO->getLHS()->IgnoreCasts()->getType();
7    QualType RHSType = BO->getRHS()->IgnoreCasts()->getType();
8    if ((!LHSType->isPointerType() && !LHSType->isArrayType()) ||
9        (!RHSType->isPointerType() && !RHSType->isArrayType()))
10     return;
11
12   const auto *LeftBase = C.getState()
13                          ->getSVal(BO->getLHS(), C.getLocationContext())
14                          .getAsRegion()
15                          ->getBaseRegion();
16   const auto *RightBase = C.getState()
17                           ->getSVal(BO->getRHS(), C.getLocationContext())
18                           .getAsRegion()
19                           ->getBaseRegion();
20
21   if (LeftBase != RightBase)
22     reportWarning(C, BO);
23   return;
24 }
```

Листинг 10. Реализация детектора для правила MISRA C 2012 18.3 в Clang Static Analyzer.
Listing 10. Implementation of checker for MISRA C 2012 Rule 18.3 in Clang Static Analyzer.

Для реализации детектора был использован единственный обработчик (`checkPreStmt`), перехватывающий бинарные операторы (`BinaryOperator`, строка 1). В обработчике проверяется, что бинарный оператор является оператором сравнения (строка 3), затем проверяется, что типы левой и правой частей оператора являются адресными (указателями или массивами, строки 8-9).
Из примеров на листинге 9 видно, что детектору правила 18.3 необходим анализ алиасов. CSA предоставляет возможность анализа памяти символов исследуемого исходного кода с помощью отображения каждого символа на предполагаемый регион памяти. Регионы могут быть разных типов, например, таких, как куча (`HeapSpaceRegion`), статическая память

(`StackSpaceRegion`) и т.д. Этой функциональности достаточно для корректной работы детектора в большинстве случаев. Точность анализа можно улучшить, если реализовать алгоритм анализа алиасов из модуля `LLVM Alias Analysis` [19] средствами CSA.
На листинге 10, для проверки того, что указатели являются алиасами, извлекаются их базовые регионы памяти (строки 12-19) и затем сравниваются (строка 21). Если регионы памяти не эквивалентны, то выдается предупреждающая диагностика (строка 22).

5.2.2 Правило 22.7

Рассмотрим правило 22.7 из стандарта MISRA C 2012, для которого потребовалась более сложная реализация детектора. Правило звучит следующим образом:

> «Макроопределение `EOF` должно сравниваться только с неизменным возвращаемым значением функции стандартной библиотеки, которая может его вернуть»

Возможный вариант кода, нарушающего правило 22.7, представлен на листинге 11.

```
void rule22_7 ( void ) {
  int c = getchar();
  if (EOF == c) {  // Нет нарушения
    ...
  }

  if (EOF == (char)c) {  // Нарушение - сужающее приведение типов приводит к изменению
    …                    // значения с точки зрения правила
  }
}
```

Листинг 11 Иллюстрация различных аспектов правила MISRA C 2012 22.7.
Listing 11. Code markup according to the MISRA C 2012 Rule 22.7.

Правило предполагает, что не должно быть сужающих преобразований типа для значений, возвращенных функциями, которые могут вернуть `EOF`.
Детектор для данного правила должен отслеживать возвращаемые значения вызовов функций, которые могут вернуть `EOF` с точки зрения стандарта языка `C`, а также их прямое изменение, посредством записи в переменную, содержащую возвращенное значение, и косвенное, посредством приведения типов.
Для реализации детектора для данного правила был применен анализ помеченных данных (*taint analysis*), доступный в CSA. Символьное значение считается помеченным (*tainted*), если известно, что оно было получено из «ненадежного» источника, например, путем чтения стандартного ввода или файлового дескриптора, или из переменных окружения. Анализ помеченных данных – это эффективный метод обнаружения дефектов безопасности, основанный на обнаружении использования помеченных значений в вызовах чувствительных функций.

Рассмотрим подробную реализацию обработчиков детектора (листинг 12).

При реализации детектора были использованы следующие обработчики – `checkPostCall`, `checkLocation`, `checkPostStmt` и `checkPreStmt`.

Функция `mayRetEOF` (строка 2) в обработчике `checkPostCall`, который используется для отслеживания вызовов функций, проверяет принадлежность вызванной функции заранее составленному списку имен библиотечных функций, которые могут вернуть `EOF`, таких, как, например, `getchar` из библиотеки *`stdio.h`*. Если проверка успешна, то возвращенное значение и выражение вызова функции сохраняются в контейнере состояния (вызов `addCallExprToSym`, строка 7). Затем возвращенное значение помечается для его дальнейшего отслеживания (вызов `addTaint`, строка 8).

```
1  void EOFCompUnmodResChecker::checkPostCall(const CallEvent &Call, CheckerContext &Ctx) const {
2    if (!mayRetEOF(Call))
3      return;
4
5    ProgramStateRef State = Ctx.getState();
6    SymbolRef RetVal = Call.getReturnValue();
7    State = addCallExprToSym(RetVal, Call.getOriginExpr(), State);
8    Ctx.addTransition(taint::addTaint(State, RetVal, PointerToEOFRet));
9  }
10
11 void EOFCompUnmodResChecker::checkLocation(SVal Loc, bool IsLoad, const Expr *E,
12                                                                   CheckerContext &C) const {
13   ProgramStateRef State = Ctx.getState();
14   if (!IsLoad && taint::isTainted(State, Loc, PointerToEOFRet))
15     Ctx.addTransition(addExprToSym(Loc, E, State));
16 }
17
18 void EOFCompUnmodResChecker::checkPostStmt(const CastExpr *Exp,   CheckerContext &Ctx) const {
19   const Expr *CastedExpr = Exp->getSubExpr();
20   if (Exp->getType().getUnqualifiedType() ==
21       CastedExpr->getType().getUnqualifiedType())
22     return;
23
24   ProgramStateRef State = Ctx.getState();
25   const SVal CExpVal = Ctx.getSVal(Exp);
26   if (taint::isTainted(State, CExpVal, PointerToEOFRet))
27     Ctx.addTransition(addExprToSym(CExpVal, Exp, State));
28 }
```

Листинг 12. Реализация обработчиков checkPostCall, checkLocation и checkPostStmt детектора правила MISRA C 2012 22.7 в Clang Static Analyzer.
Listing 12. checkPostCall, checkLocation and checkPostStmt handlers implementation for MISRA C 2012 Rule 22.7 checker in Clang Static Analyzer.

На листинге 13 представлена структура использованного контейнера состояния, представляющего собой отображение символов на выражения, в которых данные, относящиеся к ним, изменяются:

```
1  REGISTER_SET_FACTORY_WITH_PROGRAMSTATE(ExprSet, const Stmt *)
2  REGISTER_MAP_WITH_PROGRAMSTATE(ChangeExprs, const SymExpr *, ExprSet)
3  REGISTER_MAP_WITH_PROGRAMSTATE(EOFFuncCallExprs, const SymExpr *, const CallExpr *)
```

Листинг 13. Объявление контейнера для детектора правила MISRA C 2012 22.7 в CSA.
Listing 13. Container declaration for MISRA C 2012 22.7 Rule checker in Clang Static Analyzer.

Библиотека CSA предоставляет несколько видов нетипизированных хранилищ данных (*GDM*), таких как список (`SET`), словарь (`MAP`), линейный динамический массив (`LIST`) и переменная (`TRAIT`), в которых можно хранить пользовательские данные. Каждое изменение контейнера привязывается напрямую к текущему состоянию, соответственно, все состояния, порожденные из текущего, наследуют все изменения контейнера. Библиотека CSA также предоставляет возможность создания фабрик, позволяющих описывать структуры контейнеров с более сложными типами данных. На листинге 13 создана фабрика (строка 1) для хранения списка выражений, ключом для которой является символьное выражение в словаре `ChangeExprs` ниже (строка 2). В строке 3 определен словарь `EOFFuncCallExprs`, содержащий выражения вызова функций, возвращающих `EOF`.
На листинге 12 обработчик `checkLocation` используется для сбора информации об изменениях помеченных символов (строка 13) при прямой записи в них. Далее выражение добавляется в контейнер состояния (строка 15). В обработчике `checkPostStmt`, который используется для сбора информации об изменениях помеченных символов посредством приведения типов, происходит проверка (строки 20-21), что приводимый тип отличается от

исходного, чтобы исключить случаи ложных срабатываний в местах бесполезных приведений типов, затем проверяется, что символ был помечен, а значит был возвращен функцией, которая может вернуть `EOF` (строка 26). Если символ был помечен, то выражение приведения типа добавляется в контейнер состояния (строка 27) для дальнейшего анализа и выдачи диагностики. Необходимо обратить внимание на первый параметр обработчика, который имеет тип `const CastExp*`, означает, что обработчик будет срабатывать только на выражениях приведения типа, что существенно уменьшает количество его вызовов.

Алгоритм реализации обработчика `checkPreStmt`, который используется для идентификации сравнения помеченных переменных со значением `EOF` и выдачу предупреждения, представлен на рис. 4. Обработчик перехватывает только бинарные операторы. После проверки, что бинарный оператор представляет собой оператор сравнения с `EOF`, извлекается символ, который с `EOF` сравнивается. Затем происходит отбор всех выражений, в которых данный символ изменялся и выдача диагностики с подсказками.

Algorithm 1 Алгоритм выдачи диагностического предупреждения и подсказок в месте сравнения с EOF

```
Input
    EOFFuncCallExprs - словарь, содержащий отображения возвращенного значения к выражению вызова, возвращающего EOF
    ChangeExprs - словарь, содержащий отображения символов к выражениям их порождающим
    BOp - выражение бинарного оператора
Output
    Выдача диагностического предупреждения и подсказок в месте сравнения с EOF
if isRelationalOp(BOp) is false then                          ▷ Проверка, что оператор является реляционным
    return
end if
if isComparisonWithEOF(BOp) is false then                     ▷ Проверка, что в операторе есть сравнение с EOF
    return
end if
RelOpSym ← getSymFromRelOp(BOp)                               ▷ Извлечение символа, который сравнивается с EOF
EOFCallExpr ← None
for each Sym in getSymDependenceList(RelOpSym) do             ▷ Перебор зависящих символов
    if Sym in EOFFuncCallExprs then          ▷ Поиск символа среди всех, которые были возвращены EOF функциями
        EOFCallExpr ← getExprBySym(RelOpSym, EOFFuncCallExprs)
        break
    end if
end for
if EOFCallExpr is None then
    return
end if
NoteList                                                      ▷ Инициализация пустого листа подсказок
for each Sym in getSymDependenceList(RelOpSym) do             ▷ Перебор зависящих символов
    if Sym in SymToExprs then                                 ▷ Поиск выражения порождающего символ
        ChangeExpr ← getExprBySym(Sym, ChangeExprs)           ▷ Извлечение выражения, которое изменяет символ
        addNoteToNoteList(ChangeExpr,"Symbol was changed here", NoteList)   ▷ Генерация подсказки
    end if                                                    ▷ в месте изменяющего символ выражения
end for
if isEmpty(NoteList) is true then
    return
end if
addNoteToNoteList(EOFCallExpr,"Symbol was produced here.")    ▷ Генерация подсказки в месте вызова EOF функции
generateWarn(BOp, NoteList)   ▷ Генерация предупреждающей диагностики с подсказками в месте сравнения с EOF
```

Рис. 4. Алгоритм обработчика checkPreStmt для детектора правила MISRA C 2012 22.7 в CSA.
Fig. 4. checkPreStmt algorithm for Clang Static Analyzer MISRA C 2012 Rule 22.7 checker.

6. Тестирование

Для оценки качества реализованных детекторов использовались синтетические примеры из стандарта безопасного кодирования MISRA C 2012 [20], а также следующие проекты с открытым исходным кодом:

- zlib [21] версии 1.2.11 – свободная кроссплатформенная библиотека для сжатия данных;

- openjpeg [22] версии 2.5.0 – свободная библиотека для кодирования и декодирования JPEG 2000 изображений;
- openssl [23] версии 3.0.2 – полноценная криптографическая библиотека с открытым исходным кодом;
- coreJSON [24] версии 3.2.0 – библиотека, соответствующая стандарту кодирования MISRA C 2012, для работы с данными в формате JSON.

Тестирование осуществлялось на компьютере с 4-ядерным процессором Intel E3- 1265LV2 с ограничением тактовой частоты в 2.6 ГГц под управлением операционной системы Ubuntu 20.04.5 (Focal Fossa), объем оперативной памяти 32GB. Для запуска и сбора результатов анализа использовалась инфраструктура статического анализа CodeChecker [25].
На проектах с открытым исходным кодом качество анализа оценивалось путем ручной разметки результатов анализа и подсчета точности, которая вычислялась как отношение истинных срабатываний к сумме истинных (TP) и ложных (FP) рассмотренных срабатываний. Так как срабатываний на проектах с открытым исходным кодом крайне много, то детекторы с большим числом срабатываний оценивались по 100 случайным срабатываниям в различных файлах проекта. Результаты представлены в табл 2.

Табл. 2. Результаты анализа разработанных детекторов для Clang-Tidy / CSA на проектах с открытым исходным кодом
Table 2. Results of the analysis of developed checkers for Clang-Tidy / CSA on open source projects

Проект	**Срабатываний**			**Точность**	**Время анализа**
	Всего	Рассмотренных	FP		
zlib	8388	1034	0	100%	33 сек
openjpeg	20371	1433	9	99.4%	5 мин
openssl	275528	2229	7	99.7%	49 мин, 29 сек
coreJSON	7	7	0	100%	10 сек

Из результатов анализа следует, что разработанные детекторы имеют высокую точность и скорость работы. Большая часть ложных срабатываний связана с детекторами, использующими модуль Clang-Tidy – *MutationAnalyzer*, предназначенный для поиска мутаций переменных в переданном контексте. *MutationAnalyzer* использует достаточно неконсервативный подход к поиску мутаций, например, оператор взятия адреса (&) по умолчанию трактуется как мутация, что не всегда верно. Часть ложноположительных срабатываний связана с CSA детекторами, использующими модуль *RangeConstraintManager*, в котором есть проблема с определениями диапазонов значений символов при расширяющих и сужающих преобразованиях типов.
Также было проведено сравнение реализованных детекторов с анализатором Cppcheck (раздел 2.1) на открытых проектах coreJSON и zlib. Стоит отметить, что coreJSON разрабатывался с учетом стандарта MISRA C 2012, за некоторыми задокументированными исключениями [26]. Соответствие кода стандарту MISRA разработчики coreJSON проверяют коммерческим статическим анализом Coverity (раздел 2.2). При тестировании coreJSON срабатывания детекторов на правилах из исключений не учитывались.
В табл. 3 представлены результаты анализа проекта coreJSON.

Табл. 3. Сравнение разработанных детекторов для Clang-Tidy/CSA с Cppcheck на проекте coreJSON
Table 3. Comparison of developed checkers for Clang-Tidy/CSA witch Cppcheck on coreJSON project

Анализатор	Clang-Tidy/CSA			Cppcheck		
Срабатываний / Правило	TP	FP	FN	TP	FP	FN
10.3	1	0	0	1	1	1
10.4	0	0	0	7	7	0
12.3	0	0	0	11	11	0
13.5	5	0	0	0	0	5
20.9	0	0	0	1	1	0
20.12	1	0	0	0	0	1

На проекте coreJSON с помощью разработанных детекторов были найдены несоответствия кода правилам 10.3, 13.5 и 20.12, которые не были найдены анализатором Coverity.

В качестве примера рассмотрим срабатывание разработанного детектора для правила MISRA C 2012 13.5, которое звучит следующим образом:

> «Правый операнд логического оператора && или || не должен содержать устойчивых побочных эффектов»

Стандарт языка C гласит:

1) вычисление логических операторов && и || происходит слева направо;
2) вычисление логического оператора прекращается, когда результат может быть определен.

Побочный эффект считается *устойчивым* в определенной точке выполнения, если он может повлиять на состояние выполнения программы в этой точке.

Следующие побочные эффекты являются устойчивыми в определенной точке программы:

- изменение файла;
- изменение объекта;
- доступ к *volatile* объекту.

На листинге 14 представлен отрывок исходного кода coreJSON, на котором произошло срабатывание детектора. В данном коде возможен случай, при котором функции `skipAnyString` или `skipAnyLiteral` возвращают значение `true` в операторе `if` (строки 15-17). В таком случае вычисление логических операторов прекращается и правый операнд оператора || – функция `skipNumber` не будет вызвана (строка 17). Вызов функции `skipNumber` может иметь побочный эффект, заключающийся в изменении значения переменной `start` по указателю: `*start = i;` (строка 7). Этот побочный эффект является *устойчивым*: значение указателя `start` далее используется в функции `skipArrayScalars` (строка 34), что может не соответствовать ожиданиям разработчика.

О найденных несоответствиях стандарту MISRA C 2012 было сообщено разработчикам coreJSON в результате чего были внесены соответствующие исправления в код [27].

В табл 4. приведены результаты анализа проекта zlib для правил, результат анализа которых отличался между анализаторами Clang-Tidy/CSA и Cppcheck.

```
1   static bool skipNumber( const char * buf, size_t * start, size_t max ) {
2       bool ret = false;
3       size_t i;
4       ...
5       if( ret == true )
6       {
7           *start = i;     // устойчивый побочный эффект
8       }
9       return ret;
10  }
11
12  static bool skipAnyScalar( const char * buf,  size_t * start, size_t max ) {
13      bool ret = false;
14
15      if( ( skipString( buf, start, max ) == true ) ||
16          ( skipAnyLiteral( buf, start, max ) == true ) ||
17          ( skipNumber( buf, start, max ) == true ) )  // Нарушение правила 13.5
18      {
19          ret = true;
20      }
21      return ret;
22  }
23
24  static void skipArrayScalars( const char * buf, size_t * start, size_t max ) {
25      size_t i;
26      ...
27      while( i < max )
28      {
29          if( skipAnyScalar( buf, &i, max ) != true )
30          {
31              break;
32          }
33      }
34      *start = i;
35  }
```

Листинг 14. Фрагмент кода coreJSON с найденным несоответствием правилу 13.5.
Listing 14. coreJSON code snippet showing non-compliance with Rule 13.5.

Табл. 4. Сравнение разработанных детекторов для Clang-Tidy/CSA с Cppcheck на проекте zlib
Table 4. Comparison of results for developed Clang-Tidy/CSA checkers and Cppcheck on zlib project

Анализатор	Clang-Tidy/CSA			Cppcheck		
Срабатываний / Правило	TP	FP	FN	TP	FP	FN
7.2	1172	0	0	61	0	1111
10.3	98	0	0	20	0	78
15.7	15	0	0	11	0	4
16.3	38	0	0	13	0	25
12.3	26	0	0	30	24	20

На проекте zlib (34 файла) время анализа Cppcheck составляет 42 секунды, а разработанные детекторы на основе Clang Tidy / CSA анализируют тот же проект за 33 секунды. Разница в производительности составляет около 27%. Также стоит отметить существенное количество ложноотрицательных (FN) и ложноположительных (FP) срабатываний в Cppcheck, которые обусловлены некачественным разбором AST. Так, например, анализ правила 15.7 не учитывает конструкции «`if ... else if`» без «`{`», а ложноположительные срабатывания на правиле 12.3 обусловлены некорректной интерпретацией бинарного оператора «запятая»

`(,)` – запятая-разделитель в списке объявлений множества переменных не является бинарным оператором с точки зрения стандарта языка `C`.

Также было проведено сравнительное тестирование разработанных детекторов и анализатора Cppcheck на синтетических примерах MISRA C 2012 [20] для правил, которые классифицируются стандартом как *обязательные*. Тестирование показало, что разработанные детекторы на основе Clang-Tidy / CSA не имеют ложных срабатываний на синтетических примерах MISRA в отличие от Cppcheck, поэтому являются эффективнее с точки зрения точности анализа инструмента Cppcheck даже на *обязательных* правилах.

7. Заключение

В данной работе представлены разработанные статические детекторы для проверки кода на соответствие рекомендациям стандарта безопасного кодирования MISRA C 2012. Статические детекторы реализованы на основе инфраструктуры анализаторов Clang-Tidy и Clang Static Analyzer из компиляторной инфраструктуры LLVM. На синтетических примерах разработанные детекторы не имели ложных срабатываний, а на представленных проектах с открытым исходным кодом точность анализа составила порядка 99%. По сравнению с анализатором Cppcheck разработанные детекторы дают более точные и полные результаты, а также высокую скорость анализа.

В рамках дальнейшей работы планируется разработать опцию компилятора *clang*, которая позволит компилировать, проверяя на полное соответствие исходного кода стандарту MISRA C 2012, а также разработка статических детекторов для проверки правил из стандартов AUTOSAR C++14.

Также планируется провести сравнительное тестирование разработанных детекторов с коммерческими анализаторами, поддерживающими проверку кода на соответствие стандарту MISRA C 2012.

Список литературы / References

[1]. MIRSA official website. https://www.misra.org.uk/, accessed 01.11.2023.
[2]. SEI CERT C Coding Standard. https://wiki.sei.cmu.edu/confluence/display/c, accessed 01.11.2023.
[3]. AUTOSAR official website. https://www.autosar.org/, accessed 01.11.2023.
[4]. The LLVM Compiler Infrastructure. https://llvm.org/, accessed 01.11.2023.
[5]. Clang Tidy. https://clang.llvm.org/extra/clang-tidy/, accessed 01.11.2023.
[6]. Clang Static Analyzer. https://clang.llvm.org/docs/ClangStaticAnalyzer.html, accessed 01.11.2023.
[7]. Introduction to the Clang AST. https://clang.llvm.org/docs/IntroductionToTheClangAST.html, accessed 01.11.2023.
[8]. Cppcheck A tool for static C/C++ code analysis. http://cppcheck.net/, accessed 01.11.2023.
[9]. SonarQube. https://www.sonarsource.com/products/sonarqube/, accessed 01.11.2023.
[10]. Coverity Static Analysis. https://www.synopsys.com/software-integrity/static-analysis-tools-sast/coverity.html, accessed 01.11.2023.
[11]. Klocwork static analyzer. https://www.perforce.com/products/klocwork, accessed 01.11.2023.
[12]. PVS-Studio static analysis system. https://pvs-studio.com/en/, accessed 01.11.2023.
[13]. Motor Industry Software Reliability Association, MISRA-C:1998, Guidelines for the use of the C language in vehicle based software. Nuneaton, Warwickshire CV10 0TU, UK: MIRA Ltd, Jul. 1998.
[14]. The Motor Industry Research Association, Development Guidelines For Vehicle Based Software. Nuneaton, Warwickshire CV10 0TU, UK: The Motor Industry Research Association, Nov. 1994.
[15]. MISRA, MISRA C:2012 Amendment 1 – Additional security guidelines for MISRA C:2012. Nuneaton, Warwickshire CV10 0TU, UK: HORIBA MIRA Ltd, Apr. 2016.
[16]. MISRA, MISRA C:2012 Addendum 2 – Coverage of MISRA C:2012 (including Amendment 1) against ISO/IEC TS 17961:2013 “C Secure”

[17]. AST Matcher Reference, https://clang.llvm.org/docs/LibASTMatchersReference.html, accessed 01.11.2023.
[18]. CRTP pattern. https://en.cppreference.com/w/cpp/language/crtp, accessed 01.11.2023.
[19]. LLVM Alias Analysis Infrastructure, https://llvm.org/docs/AliasAnalysis.html, accessed 01.11.2023.
[20]. MISRA-C-2012 Example Suite, https://gitlab.com/MISRA/MISRA-C/MISRA-C-2012/Example-Suite, accessed 01.11.2023.
[21]. Библиотека zlib. http://zlib.net/, accessed 01.11.2023.
[22]. Библиотека openjpeg. http://www.openjpeg.org/, accessed 01.11.2023.
[23]. Библиотека openssl. https://www.openssl.org/, accessed 01.11.2023.
[24]. Библиотека coreJSON. https://github.com/freertos/corejson, accessed 01.11.2023.
[25]. Инфраструктура статического анализа CodeChecker. https://codechecker.readthedocs.io/en/latest/, accessed 01.11.2023.
[26]. coreJSON: MISRA Compliance, https://github.com/FreeRTOS/coreJSON/blob/main/MISRA.md, accessed 01.11.2023.
[27]. coreJSON: Fix short-circuiting operations with side-effects, https://github.com/FreeRTOS/coreJSON/pull/148, accessed 01.11.2023.

Информация об авторах / Information about authors

Рубен Артурович БУЧАЦКИЙ – кандидат технических наук, научный сотрудник отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Ruben Arturovich BUCHATSKIY – Cand. Sci. (Tech.), researcher in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Ян Андреевич ЧУРКИН – стажер-исследователь отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Yan Andreevich CHURKIN – researcher in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Кирилл Алексеевич ЧИБИСОВ – инженер отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Kirill Alekseevich CHIBISOV – engineer in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Михаил Вячеславович ПАНТИЛИМОНОВ – научный сотрудник отдела компиляторных технологий. Научные интересы: статический анализ, компиляторные технологии, СУБД.

Mikhail Vyacheslavovich PANTILIMONOV – researcher in Compiler Technology department. Research interests: static analysis, compiler technologies, DBMS.

Егор Викторович ДОЛГОДВОРОВ – студент МФТИ, лаборант отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Egor Viktorovich DOLGODVOROV – a student at MIPT, laboratory assistant in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Андрей Викторович ВЯЗОВЦЕВ – студент МФТИ, лаборант отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Andrey Viktorovich VYAZOVTSEV – a student at MIPT, laboratory assistant in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Алексей Георгиевич ВОЛОХОВ – ведущий инженер отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Aleksey Georgievich VOLOKHOV – leading engineer in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Владимир Владимирович ТРУНОВ – студент МФТИ, лаборант отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Vladimir Vladimirovich TRUNOV – a student at MIPT, laboratory assistant in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Гаяне Оганнесовна МИРАКЯН – студентка Российско-Армянский Университета. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Gayane Hovhannes MIRAKYAN – a student at Russian-Armenian University. Research interests: static analysis, compiler technologies, optimizations.

Константин Николаевич КИТАЕВ – студент МФТИ, лаборант отдела компиляторных технологий ИСП РАН. Научные интересы: статический анализ программ, компиляторные технологии, оптимизации.

Konstantin Nikolaevich KITAEV – a student at MIPT, laboratory assistant in Compiler Technology department at ISP RAS. Research interests: static analysis, compiler technologies, optimizations.

Андрей Андреевич БЕЛЕВАНЦЕВ – доктор физико-математических наук, ведущий научный сотрудник ИСП РАН, профессор МГУ. Сфера научных интересов: статический анализ программ, оптимизация программ, параллельное программирование.

Andrey Andreevich BELEVANTSEV – Dr. Sci (Phys.-Math.), Prof., leading researcher at ISP RAS, Professor at MSU. Research interests: static analysis, program optimization, parallel programming.

DOI: 10.15514/ISPRAS-2023-35(5)-13

CC BY

Извлечение именованных сущностей из рецензий к исходному коду

[1,2] *В.В. Качанов,* ORCID: 0000-0002-9371-6483 <*vkachanov@ispras.ru*>
[1,3] *А.С. Хитрова*, ORCID: 0009-0003-1723-5199 <*akhitrova@ispras.ru*>
[1] *С.И. Марков,* ORCID: 0000-0002-6687-4937 <*markov@ispras.ru*>

[1] *Институт системного программирования РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
[2] *Московский физико-технический институт (НИУ),*
141701, Россия, Долгопрудный, Институтский пер., д. 9.
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1.

Аннотация. В данной статье рассматривается задача извлечения именованных сущностей из рецензий исходного кода. В работе приводится сравнительный анализ существующих подходов и предлагаются собственные методы для улучшения качества решения задачи. Предложенные и реализованные улучшения включают в себя: методы борьбы с дисбалансом данных, улучшения токенизации входных данных, использование больших массивов неразмеченных данных и применение дополнительных бинарных классификаторов. Для оценки качества собран и размечен вручную новый набор из 3000 пользовательских рецензий. Показано, что предложенные улучшения позволяют значительно увеличить показатели метрик качества, вычисляемых как на уровне токенов (+22%), так и на уровне сущностей целиком (+13%).

Ключевые слова: машинное обучение; извлечение именованных сущностей; набор данных.

Для цитирования: Качанов В.В., Хитрова А.С., Марков С.И. Извлечение именованных сущностей из рецензий к исходному коду. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 193–214. DOI: 10.15514/ISPRAS–2023–35(5)–13.

Named Entity Recognition for Code Review Comments

[1,2] *V.V. Kachanov* ORCID: 0000-0002-9371-6483 <*vkachanov@ispras.ru*>
[1,3] *A.S. Khitrova* ORCID: 0009-0003-1723-5199 <*akhitrova@ispras.ru*>
[1] *S.I. Markov* ORCID: 0000-0002-6687-4937 <*markov@ispras.ru*>

[1] *Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
[2] *Moscow Institute of Physics and Technology,*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

Abstract. This paper addresses the problem of named entities recognition from source code reviews. The paper provides a comparative analysis of existing approaches and proposes its own methods to improve the quality of problem solving. Proposed and implemented improvements include: methods to deal with data imbalances, improved tokenization of input data, the use of large arrays of unlabeled data, and the use of additional binary

classifiers. To assess quality, a new set of 3,000 user code reviews was collected and manually labeled. It is shown that the proposed improvements can significantly increase the performance measured by quality metrics, calculated both at the token level (+22%) and at the entire entity level (+13%).

Keywords: machine learning, named entity recognition, dataset.

For citation: Kachanov V.V., Khitrova A.S., Markov S.I. Named entity recognition for code review comments. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 193-214 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-13.

1. Введение

Извлечение именованных сущностей (named entity recognition, NER) – важная задача в области обработки естественного языка (natural language processing, NLP). Это комплексная задача, состоящая из двух:

1) определить, является ли слово или словосочетание именованной сущностью, и
2) определить класс, к которому данная сущность относится.

Набор классов не является фиксированным и зависит от постановки конкретной задачи: зачастую ищут имена, названия городов, названия компаний и организаций. Однако NER можно использовать и для более узкоспециализированных областей, например, рецензии и комментарии к исходному коду (source code review). В этом случае набор классов будет специфичным и больший интерес будут представлять слова, связанные с исходным кодом, на который был написан комментарий. Распознавание таких слов поможет лучше понять смысл рецензии, что, в свою очередь, можно использовать в задачах классификации, кластеризации рецензий, семантического сравнения и поиска.

В ходе решения задачи извлечения именованных сущностей необходимо разбить текст на последовательность смысловых единиц – токенов, которые далее и будут классифицированы. Обычно токенами являются отдельные слова и знаки пунктуации. Таким образом, именованная сущность может состоять из нескольких токенов так же, как название организации может содержать несколько слов. Токенизация, то есть процесс разделения на токены, может быть сложнее, чем получение последовательности слов, разделённых пробельными символами, и зависеть от области, которой принадлежит этот текст. Для отрывков исходного кода правила разделения на смысловые единицы могут зависеть от синтаксиса языка программирования.

Существует множество различных подходов к решению задачи NER. Наиболее простыми являются подходы, основанные на использовании словарей, или проверки текста на наличие каких-либо шаблонов. Возможно использование статистических моделей классификации последовательностей, таких как скрытые модели Маркова или условно случайные поля. Так как NER является задачей классификации, возможно использование метода опорных векторов. Но наилучшие результаты дают подходы с использованием нейронных сетей. Это главным образом связано со способностью нейронных сетей (например, рекуррентных или трансформеров) эффективно использовать контекст. Кроме того, использование моделей векторного представления слов позволяет хранить больше информации о токенах в тексте.

Основой реализованных классификаторов является модель семейства BERT [1] (Bidirectional Encoder Representations from Transformers) – предобученная языковая модель, основанная на архитектуре Трансформер (Transformer). Благодаря своей архитектуре BERT способен улавливать контекстную информацию. Эта модель может быть использована для решения множества различных задач обработки естественного языка, в том числе и в рамках задачи распознавания именованных сущностей. В данной работе модели семейства BERT использовались для обучения классификатора токенов на новом наборе данных.

Одной из близких работ в данном направлении является SoftNER [2], в которой авторы исследовали и разрабатывали классификатор именованных сущностей для сообщений из

форума StackOverflow[1]. К числу основных результатов работы относится применение промежуточного векторного представления для токена от различных моделей как частей одного классификатора. Такой подход был взят за основу для дальнейших исследований как наилучший из представленных.

В ходе работы также оценена эффективность всех реализованных методов для нового набора данных и проведен анализ ошибок классификаторов.

Основным вкладом этой работы являются:

- собранный и размеченный на 15 классов набор из 3000 рецензий к исходному коду;
- предложенные методы улучшения работы классификаторов;
- обученные классификаторы, распознающие 31 тип сущностей (в BIO нотации), связанных с разработкой программного обеспечения.

2. Методы извлечения именованных сущностей

Задача классификации токенов в тексте довольно распространенная и изученная. За время исследований было предложено множество подходов к решению задачи. В качестве признаков для описания слов использовались различные характеристики, такие как: форма, лемма, область окружающих слов в скользящем окне, справочная информация, статистические данные, использование заглавных букв, пунктуация и так далее. Применение искусственных нейронных сетей не только расширило список численных характеристик слов и их частей, но и повысило качество определения и классификации слов в тексте [3].

2.1 Скрытые марковские модели

В работе [4] скрытые марковские модели использовались для распознавания и классификации имен, дат, времени и числовых величин в наборах данных MUC-6, MUC-7 и трансляциях новостей. Разработанная модель при обучающей выборке в 100,000 слов научилась классифицировать с точностью в 94%.

Здесь задача NER рассматривается как задача присвоения каждому слову одной из предложенных меток либо специальной метки NOT-A-NAME, чтобы обозначить, что слово не принадлежит ни одной из меток. Далее используется вероятностная модель для расчета возможности появления слов в контексте биграмм.

В более формальной постановке задача состоит в поиске наиболее вероятной последовательности имен классов (NC) при заданной последовательности слов (W): $\max P(NC|W)$.

2.2 Условно случайные поля

Условно случайные поля (conditional random field, CRF) были введены авторами работы [5] как инструмент статистического моделирования для задачи распознавания шаблонов. В работе [6] был предложен вариант использования CRF для извлечения именованных сущностей.

$$P(s|o) = \frac{\exp(\sum_{t=1}^{T} \beta_k f_k(s_{t-1}, s_t, o, t))}{Z},$$

где $o = < o_1, o_2, \dots, o_T >$ – входная последовательность (слов), $s = < s_1, s_2, \dots, s_T >$ – последовательность состояний (меток, соответствующих входной последовательности), Z – некоторый нормализующий коэффициент $f_k(s_{t-1}, s_t, o, t)$ – произвольная функция-признак, β_k – обучаемый вес признаков. Значения переходов между двумя состояниями могут быть вычислены с помощью алгоритмов динамического программирования. В работе была

[1] https://stackoverflow.com/questions

показана эффективность предложенного решения, на соревновании CoNLL 2003 получив 84.04% для английского и 68.11% для немецкого языков.

2.3 Метод опорных векторов

Метод опорных векторов был представлен Кортесом и Вапником [7]. В основе алгоритма лежит идея разделяющей линейной гиперплоскости, которая старается максимизировать расстояние от этой плоскости до ближайших точек обоих классов. В работе [8] описывается применение метода опорных векторов к задаче извлечения именованных сущностей. Суть заключается в создании 8 классификаторов, по одному на класс. В качестве вектора признаков используется более 1200 бинарных меток про каждое слово. При описании каждого слова также используется его контекст размера 7 (3 слова до и 3 после). Имея результаты классификации 8 моделей, метка для слова определяется исходя из коэффициентов уверенности моделей. Если ни одна из них не уверена, что слово принадлежит соответствующему классу, то ставилась метка "O".

2.4 Нейронные сети

С распространением методов машинного обучения и глубоких нейронных сетей для задач обработки естественного языка они начали применяться и для задачи NER. Так как рекуррентные нейронные сети работают с последовательностью данных, они хорошо подходят для обработки естественного языка. Более того, разработанные позже Long Short-term Memory (LSTM) сети умеют "запоминать" длинные зависимости в тексте, улучшая понимание текста целиком. В работе [9] было показано, что решения на основе BiLSTM-CRF показали лучшие результаты на исследуемых выборках (в наборе CoNLL 2003 для английского – 90.94% F1-score, для немецкого – 78.76% F1-score, CoNLL 2002 для испанского – 86.75% F1-score). Применение искусственных нейронных сетей в задаче NER не ограничивается только классической предметной областью с обнаружением сущностей классов ORG, PERSON, DATETIME, но также используется в области медицины [10] для поиска названий болезней, симптомов и фармацевтических препаратов. Моделей на архитектуре Трансформер с момента их появления активно используются во многих задачах, связанных с обработкой естественного языка. Эта архитектура основана на механизме внимания [11], что позволяет отлавливать зависимости между удалёнными словами в предложении. Более того, оказалось, что для извлечения именованных сущностей в большинстве областей модели, основанные на этой архитектуре, превосходят другие [12]. В работе [13] описано применение BERT моделей для решения задачи извлечения именованных сущностей из текстов о кибербезопасности на русском языке. Авторы предлагают метод аугментации данных для получения большего количества примеров именованных сущностей. Одним из способов улучшения результатов работы классификатора является обучение большой языковой модели на специфическом наборе текстов. Наиболее приближенной к нашей задаче можно считать классификацию именованных сущностей, связанных с программным обеспечением в вопрос-ответах со StackOverflow, рассмотренную в статье [2]. Основой предлагаемого решения являются глубокие нейронные сети и обучение с учителем. В работе рассматриваются несколько моделей векторного представления слов (ELMo, GloVe, BERT). Также предлагаются 2 метода повышения качества классификации: путем предобучения модели, кодирующей слова в векторы; использование дополнительных моделей, предоставляющих векторные представления слов. Наилучшей рассмотренной конфигурацией оказалось использование предобученного на большом наборе неразмеченных сообщений со StackOverflow кодировщика BERT (BERTOverflow) и использование промежуточных векторов из двух моделей в качестве дополнительных признаков для

основного классификатора. К достоинствам работы также можно отнести создание открытого вручную размеченного набора данных.

3. Набор размеченных рецензий к исходному коду

В данном разделе описан процесс создания набора данных CodeReviewCommentsNER, который доступен публично на Zenodo[2].

Существует не так много решений для извлечения именованных сущностей для рецензий исходного кода или похожих предметных областей [14, 15]. Единственным доступным набором данных, связанным с разработкой программного обеспечения с достаточно детальной разметкой, является набор SoftNER [2]. К положительным аспектам можно отнести схожую предметную область, а именно комментарии пользователей из форума StackOverflow. Также безусловным плюсом является наличие меток, связанных с исходным кодом: Class, Variable, Function, Value, Data Type.

К недостаткам набора данных SoftNER можно отнести правила токенизации. Например, единым токеном с меткой "Code_Block" является «`size(S.vertices)+1`» и аналогично «`as.numeric(df$EventName)`» с меткой Library_Function, хотя с нашей точки зрения оба этих выражения можно разбить на более мелкие вызовы методов, использование переменных и константных значений.

Также не все предложенные классы являются четко сформулированными, актуальными и достаточно часто используемыми в рецензиях к исходному коду.

Так, класс "IN LINE CODE" всегда будет пересекаться с другими классами, связанными с кодом, и его можно скорее использовать в случае иерархической классификации как обобщающий.

Кроме того, предложенный набор данных имеет распределение сущностей не схожее с тем, что было получено из рецензий к исходному коду. Например, из табл. 1 видно, что в предложенном наборе количество имен переменных в 13 раз меньше, чем в исследованных нами рецензиях.

Табл. 1. Процентное соотношение важных классов в наборе данных
Table 1. Percentage of important classes in the dataset

Набор данных	**CodeReviewCommentsNER**	**SoftNER**
Variable	2.7%	0.2%
Value	1.8%	0.8%
Function	1.5%	0.5%
External_Tool/Application	0.3%	1.1%

Таким образом, из-за отличия предметной области и разногласий в разметке данных, было необходимо собрать собственный набор данных.

Для анализа и разметки было собрано 3000 рецензий к исходному коду из различных проектов с открытым исходным кодом (из Github[3] проектов на Java, Python, C/C++, из Gerrit систем Android Open Source Project[4], Tizen Gerrit[5]). 2577 комментариев были взяты случайным образом из общей базы в 420 тысяч комментариев. 433 комментария подбирались по вхождению ключевых слов для увеличения количества представителей некоторых классов из этой же общей базы.

[2] https://zenodo.org/doi/10.5281/zenodo.10060889
[3] https://github.com/
[4] https://android-review.googlesource.com/
[5] https://review.tizen.org/gerrit/

3.1 Описание классов

В качестве начального набора классов были рассмотрены 20, предложенных авторами работы SoftNER: CLASS, VARIABLE, IN LINE CODE, FUNCTION, LIBRARY, VALUE, DATA TYPE, HTML XML TAG, APPLICATION, UI ELEMENT, LANGUAGE, DATA STRUCTURE, ALGORITHM, FILE TYPE, FILE NAME, VERSION, DEVICE, OS, WEBSITE, USER NAME.

Как было описано выше, такой набор меток требует переработки для рецензий исходного кода.

В классе APPLICATION был расширен смысл и преобразован в External_Tool. Классы UI ELEMENT, LANGUAGE, DATA STRUCTURE, ALGORITHM, VERSION, DEVICE и USER NAME было решено убрать. Добавлен класс Error_Name.

Итоговый набор из 15 классов представлен в табл. 2.

Разметку первых 100 комментариев проводили три специалиста, средний коэффициент согласия каппа Коэна составил 0.82, что указывает на почти полное согласие размечающих. После анализа разногласий по некоторым классам в инструкцию для разметки были внесены дополнения и примеры согласованных именованных сущностей. Дальнейшая разметка проводилась без перекрёстного анализа, и произведена частичная кросс-валидация, показавшая высокий коэффициент согласия. Основным типом несогласованностей был пропуск именованных сущностей, что составило менее 0,1% от общего числа токенов. Для итогового набора размеченных данных был проведен повторный просмотр комментариев и исправлены пропущенные метки.

3.2 Токенизация рецензий

Одной из отличительных черт наших данных является особенность разделения текста на токены. Разбивать просто по пробелам не является верным подходом, так как исходный код часто содержит множество смысловых объектов, которые разделены различными знаками – точками, скобками и т.д., а правила написания программного кода, в свою очередь, зависят от правил синтаксиса конкретного языка. Таким образом возникает необходимость создания правил получения списка токенов из текста, которые будут учитывать все эти отличительные черты текста рецензии, но при этом не создавать излишне большого количества токенов.

С помощью регулярных выражений был создан метод токенизации, удовлетворяющий данным требованиям. Основными принципами являются:

- отделение буквенно-цифровых символов от не буквенно-цифровых;
- отделение кавычек от остальных символов;
- отделение точек и запятых от буквенно-цифровых символов;
- отделение скобок от нескобочных символов.

Благодаря относительно небольшому количеству правил удалось сохранить высокую скорость токенизации, и при этом получить разбиение, которое хорошо соответствует ручной разметке и подходит для разбиения отрывков кода вне зависимости от использованного языка программирования.

В модель передаются данные уже после предварительной токенизации. Далее эти токены подразбиваются на ещё меньшие токены, которые могут быть представлены моделью BERT для получения ее векторного представления.

3.3 Характеристики данных

Всего в 3000 рецензий содержалось 94328 токенов. При разделении на обучающую и тестовую выборки соблюдалось правило, что в тестовую попадает 10% + 1 пример каждого класса, чтобы избежать ситуации, когда примеров малочисленных классов нет в тестовом наборе. Некоторая общая статистика приведена в табл. 3.

Табл. 2. Список меток-классов с описанием
Table 2. List of class labels with description

Имя класса	**Описание**	**Примеры**
Variable	имена переменных и объектов	logicColumn, Position, mount_point_count, MS_BIND
Function	имена функций, методов и процедур, не захватывая символы "()"	_handle_fromlist, EXPECT_THAT, getInput, ForEach
Class	имена классов, структур	DisplayVk, VirtualHost, ImagePipeline, ImporterMesh, UniformLocation
Value	константные строки, включая ", числа, булевы значения, значения «null», «None»	"1px solid grey", "<classpathentry ...>", "0 0,1,2,3 11 19 AUG ? 2018", 0.7, 30, 1, 2, null
File_Name	название файла с расширением, полный путь к файлу/директории	testing/tf.py, wine_data.csv, /data/app/lib/x86/libgame.so
File_Type	упоминание расширения файла, не учитывая полного имени файла	GIFs, webp, .jar, binary
Keyword	ключевые слова используемые в языках программирования	PUBLIC, ifndef, unsigned, class, interface, else, select, yield
Data_Type	название типов данных, включая пользовательские типы и структуры, если используются в качестве типа	unsigned char, Float, ConstBytes, TypeName, Vector, LandmarkPoint, UInt64
Library_Package	упоминания подгружаемых библиотек/модулей	com.google.gerrit.httpd.rpc.change.ListChangesServlet, log4j, mlflow.projects.backend.local._create_virtualenv
Error_Name	имя класса ошибки или ее название	NoManifestException, IOException, InvalidOperation, ImportError, Unimplemented, Exceptions, NPD, ANR, OOM, out of memory
HTML_XML_Tag	html/xml тэг с символами <>	<classpathentry kind="output">, <body>, <c1>, </style>, <svg height="20" width="20">
Operating_System	название операционных систем	android, Chrome OS, Linux, Unix,SerenityOS, Win 8, bsd, arch, selinux
Programming_ Language	имена языков программирования	C++17, xml, cpp,c++, R, pythonic
External_Tool	имена сторонних приложений, которые можно запустить как самодостаточный инструмент	Qt, travis, alibaba-dubbo, pylint, JVM, dockerd, isoltest

Website	ссылки на веб-ресурсы	https://www.example.com/, https://www.example.com/somepage\#heading2, https://example.com/file.png

Табл. 3. Общая характеристика набора данных
Table 3. General characteristics of the data set

	Общий	**Обучение**	**Тест**
Кол-во примеров	3000	2543	457
Кол-во токенов	94328	80838	13490
Кол-во токенов с сущностями	15420	13429	1991
Среднее кол-во токенов сущностей на 1 пример	5.1	5.3	4.4
Кол-во сущностей	8514	7241	1285
Среднее кол-во сущностей на 1 пример	2.8	2.8	2.8

Структура по типам сущностей представлена в табл. 4. Исходя из данных в таблице, можно сделать два замечания: токенов сущностей значительно меньше, чем токенов без какой-либо сущности; некоторых сущностей не так много, но они длинные и состоят из большого количества токенов (Website, File_Name).

Табл. 4. Структура данных по сущностям
Table 4. Entity data structure

Тип сущности	**Количество токенов**	**Количество сущностей**
O	77079	77079
Variable	5089	2525
Website	3669	161
Function	3655	1429
Value	1548	823
Class	1328	666
File_Name	1169	174
HTML_XML_Tag	1027	287
Library_Package	955	278
Keyword	848	815
Data_Type	741	514
External_Tool	347	259
Error_Name	272	131
File_Type	241	154
Operating_System	193	160
Programming_Language	172	138

4. Базовое решение

4.1 Базовые модели

В качестве базового решения были выбраны модели семейства Bidirectional Encoder Representations from Transformer (BERT), а именно RoBERTa [16] и CodeBERT [17]. Первая хорошо зарекомендовала себя в разных задачах обработки естественного языка (natural

language processing, NLP). CodeBERT – модель, основанная на RoBERTa, но обученная на примерах исходного кода с документацией, для построения семантической связи между исходным кодом и описанием этого кода на естественном языке.

Кроме того, как было указано в работе SoftNER, предобученные модели на текстах из нужной области улучшают итоговые результаты. Так как полученная авторами модель BertOverflow находится в открытом доступе, она также использовалась в экспериментах.

Для предобучения на рецензиях к исходному коду были взяты около 420 тыс. уникальных комментариев с различных проектов с открытым исходным кодом. Обучение проводилось на задаче Masked Language Model (MLM), когда часть токенов маскируются специализированным токеном, а модель просят восстановить пропущенный токен.

CodeBERT – это уже предобученная модель на своем наборе данных, поэтому мы обучим её с нуля на своем наборе сообщений (PretrainCodeBert).

Таким образом есть 4 базовые модели, с которыми мы будем проводить дальнейшие эксперименты: roberta-base, codebert-base, BertOverflow, PretrainCodeBert.

Для всех классификаторов, использующих только одну BERT-подобную модель, использовалась следующая архитектура:

- BERT токенизатор: превращает входящую последовательность в токены словаря модели и их индексы, понятные для BERT encoder;
- BERT кодировщик: для каждого входящего элемента возвращает векторное представление, состоящее из 768 чисел;
- Слои классификации: набор слоев нейронных сетей в различной конфигурации, решающих задачу классификации на N классов по входному вектору признаков.

Стандартной функцией потерь для задачи многоклассовой классификации является CrossEntropy. В работе использовали оптимизатор Adam с фиксированным learning_rate=5e-6.

4.2 Вспомогательное разбиение токенов

Токенизация, проходящая для получения векторного представления для BERT, состоит из двух этапов: предварительная токенизация (претокенизация, pre-tokenization) и токенизация с помощью предобученного токенизатора BERT модели. Без использования претокенизации встроенный токенизатор BERT не может разбить длинные токены на составные части, которых нет в словаре модели BERT. Этап претокенизации нужен, чтобы получить более осмысленные разбиения в дальнейшем. Так с помощью заданных правил он разбивает названия функций, в которых был использован snake_case или camelCase, на отдельные слова, что позволяет разделить длинные токены на составные части.

Без претокенизации получим «mAssistants» -> [‘m’, ‘Ass’, ‘ist’, ‘ants’].

С претокенизацией «mAssistants» -> [‘m’, ‘Assistants’] -> [‘m’, ‘Assist’, ‘ants’] – остается информация о слове «Assist».

Эта разница возникает из-за того, что при обучении языковой модели, особенно на текстах областей, не связанных с исходным кодом, она не встречала подобные длинные токены. С другой стороны, каждое слово подобного составного токена по отдельности может существовать в словаре. В случае RoBERTa токены часто кодируются с ведущим пробелом, а значит токен без ведущего пробела имеет больший шанс быть интерпретированным неверно, так как его нет в словаре. Таким образом претокенизация позволяет справиться с данной проблемой, проводя первичное разбиение на подходящие подтокены. Далее предобученыный токенайзер BERT разбивает до тех подтокенов, которые есть в словаре BERT, для дальнейшего использования.

4.3 Дисбаланс классов

Одной из проблем данной задачи является сильный дисбаланс классов. Из табл. 4 видно, что токенов класса "O" более 77%, тогда как "Error_Name" – чуть больше 0.1%. Существует несколько видов борьбы с подобными явлениями в данных, а именно: искусственное увеличение числа примеров малочисленных классов (upsampling), уменьшение примеров многочисленных классов (downsampling), применение взвешенных функций потерь для более чувствительного обучения модели на примерах нужного типа.

Upsampling был применен в процессе сбора примеров комментариев для CodeReviewCommentsNER, когда по итогам разметки его части стало понятно, что некоторых классов мало. Для этого из общего набора рецензий при помощи регулярных выражений были собраны более подходящие примеры.

Downsampling не очень подходит, так как основной перевес имеет класс «O», а этого практически невозможно избежать в силу особенностей текстов и выбранных классов. Для остальных классов абсолютное количество было и так небольшим и применять к ним downsampling не имело смысла.

В качестве функции потерь, поддерживающей возможность передавать веса классам, был выбран FocalLoss [18]. Данная функция потерь была разработана с целью устранения дисбаланса классов в задачах компьютерного зрения:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t),\ p_t = \begin{cases} p, & \text{если } y = 1 \\ 1-p, & \text{иначе} \end{cases}$$

где $\alpha \in [0,1]$, хотя на практике α можно инвертировать и брать из промежутка $[1,\infty)$, $\gamma \in [0,5]$.

Введение такого взвешивающего (weighting) параметра α – общепринятый метод борьбы с дисбалансом классов. Этот параметр выравнивает важность многочисленных и малочисленных примеров, однако не делает различий между простыми и сложными примерами. Для этого вводится параметр γ, который помогает уменьшить вес простых примеров.

Эмпирическим методом были подобраны константы $\gamma = 2, \alpha = \begin{cases} 1, \text{для класса «}O\text{»} \\ 10, \text{иначе} \end{cases}$.

5. Вспомогательные бинарные модели

В работе SoftNER были предложены две дополнительные модели, которые обучались бинарной классификации на «O» и «Entity», после чего предпоследний линейный слой классификатора использовался как часть векторного представления токенов в основной модели. Одна из моделей основана на архитектуре BERT (Segmenter), вторая – контекстно-независимая модель, оценивающая каждое слово отдельно на основе частотных характеристик (Recognizer). Использование таких двух разных моделей по заверениям авторов должно помочь лучше классифицировать слова, которые встречаются как в контексте обычного общения, так и в участках исходного кода.

5.1 Контекстно-независимая модель

Как было сказано выше, одна из моделей контекстно-независимая, то есть обрабатывает каждое слово вне зависимости от контекста, в котором оно находится.

Она составная и содержит в себе три части:

- частотный словарь, построенный по набору данных GigaWord[6], содержащему обычный текст;

[6] https://catalog.ldc.upenn.edu/LDC2011T07

- частотный словарь, построенный по набору вопросов и ответов StackOverflow;
- модель Fasttext[7], обученная по принципу обучения без учителя на значительном наборе рецензий к исходному коду.

Далее частотные характеристики словарей превращаются в векторы методом гауссовской дискретизации [19] и конкатенируются с выходным вектором Fasttext. После чего полученный вектор проходит через пару полносвязных линейных слоев искусственной нейронной сети. Детальная схема представлена на рис. 1.

Так как эта модель также зависит от словаря и разбиения на токены, то для разбиения текста использовался токенизатор из раздела 3.2.

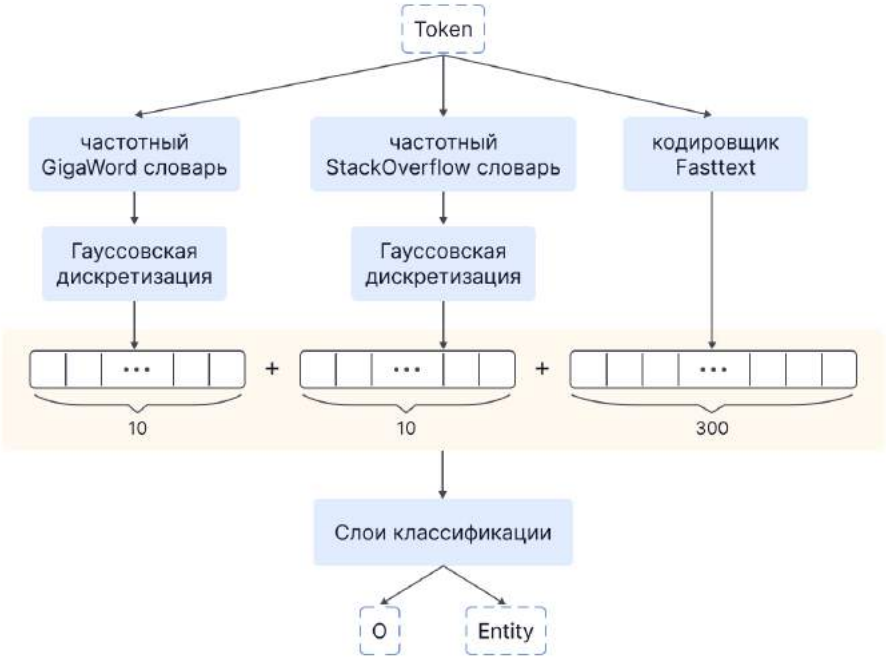


Рис. 1. Архитектура контекстно-независимой модели.
Fig. 1. Context-independent model architecture.

5.2 Контекстно-зависимая модель

Контекстно-зависимая модель построена на архитектуре BERT. Как и для базового решения многоклассовой классификации, здесь имеют место эксперименты с базовой моделью. Нами были использованы те же 4 базовые модели, описанные в разделе 4.1. Структура бинарного классификатора повторяет структуру многоклассового.

5.3 Методы взаимодействия моделей

Как уже было сказано выше, в работе SoftNER дополнительные модели применялись для предоставления дополнительной части векторного представления токена, как показано на рис. 2. В случае такой конкатенации двух векторов по 768 чисел получится слишком резкий

[7] https://fasttext.cc/

перепад размерности векторов с 1536 в 31, поэтому можно добавить промежуточный слой размера 500.

Другой вариант использования дополнительной модели предложен в [20]. В работе рассмотрены два метода: Classify-Verify и Classify-Trust. В первом методе если общая модель предсказала какой-то ответ, то это предсказание сравнивается с предсказаниями специализированной модели и принимается решение об итоговом классе. Если же общая модель предсказала «Отсутствие ответа» (что в нашем случае можно интерпретировать как класс «O»), то «Отсутствие ответа» и будет итоговым ответом. Во втором методе Classify-Trust если общая модель предсказывает какой-то ответ, то автоматически используется ответ специализированной модели, и аналогично первому методу – в случае отсутствия ответа от общей модели – такой ответ является итоговым.

В нашей конфигурации «общей» моделью можно назвать модель бинарной классификации, а «специализированной» – многоклассовую. Метод Classify-Trust (схема реализации представлена на рис. 3) применяется прямо по предложенному методу: если бинарная модель предсказывает «O» – итоговый ответ «O», иначе смотрим на предсказания многоклассовой модели. Метод Classify-Verify сложнее в интерпретации, так как не ясно каким образом сравнивать предсказания бинарной и многоклассовой моделей.

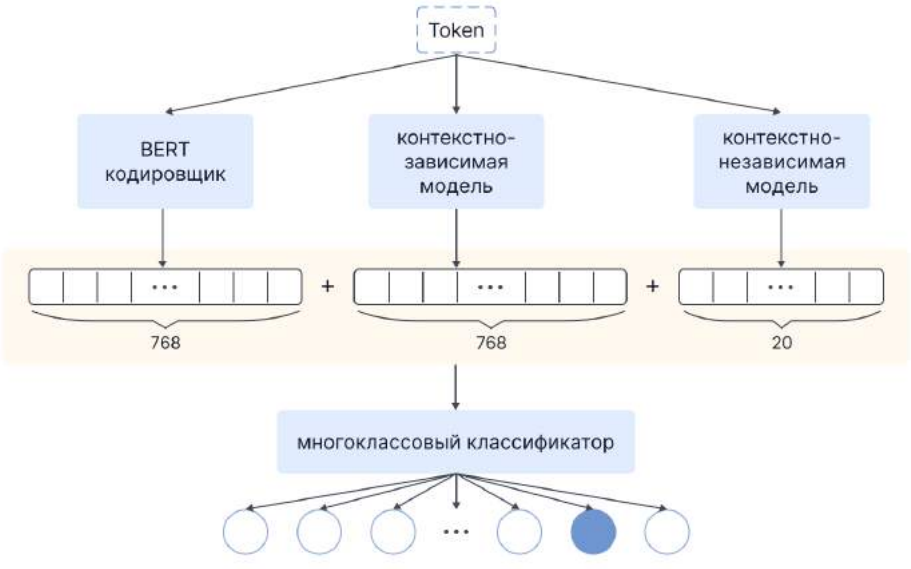


Рис. 2. Схема объединения векторных представлений методом конкатенации.
Fig. 2. Scheme for combining vector representations using the concatenation method.

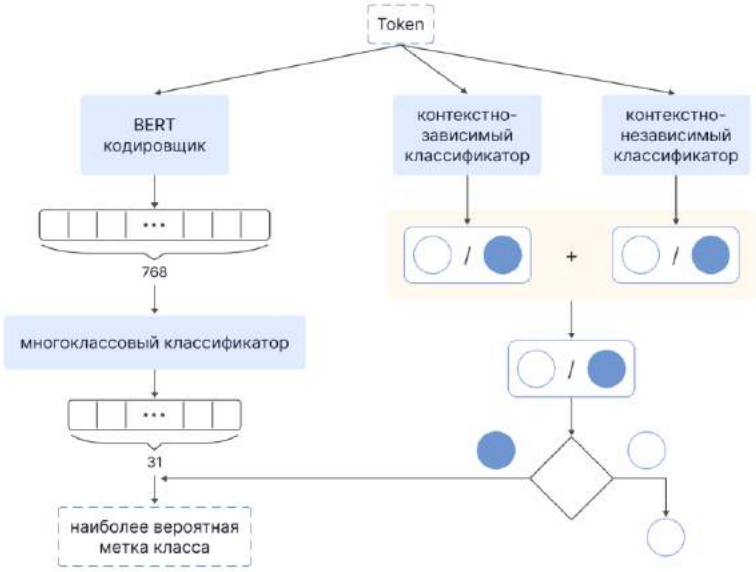


Рис. 3. Применение вспомогательных моделей методом Classify-Trust.
Fig. 3. Using auxiliary models using the Classify-Trust method.

6. Метрики

Базовыми метриками в задаче NER являются, как и в задачах классификации в целом, точность (precision), полнота (recall), f-мера (f1-score). Они вычисляются по каждому классу отдельно и усредняются по одной из стратегий: macro, micro, weighted. Для нас самой подходящей стратегией является macro, так как по ней общее среднее значение считается как среднее по всем классам, независимо от количества элементов в каждом классе отдельно. Однако эти метрики считаются по каждому токену, а как было показано в разделе 3.2, токенизаторы иногда разбивают исходные слова довольно сильно, тем самым порождая большое количество не совсем значимых токенов. Тем более, что с точки зрения конечного пользователя важнее видеть, что классификатор полностью выделил всю именованную сущность, а не только некоторые подтокены из нее.

6.1 BIO нотации

BIO (сокращенно от beginning – «начало», inside – «внутри», outside – «снаружи») нотации – это один из общепринятых форматов маркирования токенов в задаче фрагментирования текстов. Префикс B- перед тегом указывает, что тег является началом фрагмента, I- перед тегом – что тег находится внутри фрагмента. Тег O указывает, что токен не принадлежит ни одному из классов. Существует несколько разновидностей подобных нотаций: BIO/IOB, IOB2, BIOES, BILOU, которые отличаются наличием дополнительных префиксов к тегам или различными правилами разметки. Так по правилам BIO нотации префикс B- ставится токену только тогда, когда за ним токен следует I- тег того же класса. При разметке данных были использованы правила IOB2, где префикс B- ставится всегда в первом токене новой сущности.

6.2 Метрики по сущностям

На International Workshop on Semantic Evaluation (SemEval) 2013 года [21] были введены 4 способа вычислять precision/reacll/f1:

- точное совпадение границ и типа;
- точное совпадение границ, независимо от типа;
- частичное совпадение границ, независимо от типа;
- есть некоторое пересечение между предсказанным и правильным ответом.

Нас заинтересовал Strict, как самый строгий и точный метод, и Type, в котором проверяется, что есть хоть какое-то совпадение с правильным ответом. Type засчитывается в случае любого пересечения по позициям правильного ответа и предсказанного, но только если тип сущности совпадает. Strict же равен 0.0 всегда, кроме случая полного совпадения, предсказанного с правильным ответом.

7. Эксперименты и результаты

В этом разделе будут описаны результаты тестирования моделей в различных конфигурациях на нашем наборе данных CodeReviewCommentsNER.

Для оценки качества моделей многоклассовой классификации мы использовали метрики Type F1 и Strict F1, описанные в разделе 6.2, а также обычный Token F1 – F1-мера по токенам (а так как используется IOB2 нотация, то классов не 16, а 31). Для всех метрик использовалось macro усреднение. Модели бинарной классификации оценивались метриками Точность, Полнота, F-мера по токенам.

Эксперименты проводились на нашем размеченном наборе данных с фиксированным разбиением на обучающую и тестовую выборки.

Измерения проводились на рабочей станции с операционной системой Ubuntu Server 20.04 LTS, процессором Intel® Core™ i7-6700 CPU @ 3.40GHz, 32GB RAM, и графическим ускорителем NVIDIA TITAN Xp с 12GB памяти. Для конфигурации Python использовался виртуальное окружение с Python 3.9.16, torch==2.0.1, transformers==4.27.4.

Размер группы примеров (batch) был выбран 16 – максимальный размер, при котором процесс обучения помещается в память графического ускорителя. Процесс обучения на 20 эпох занимает около 15 минут для конфигураций с одной моделью и около 20 – при использовании дополнительных моделей. Запуск на тестовом наборе из 457 комментариев занимает 4-9 секунд в зависимости от конфигурации.

7.1 Базовые модели

Далее представлены результаты сравнительных запусков обучения 4 базовых моделей (из пункта 4.1) и влияние предложенных нами улучшений, описанных в пункте 4.2, 4.3.

Как видно из табл. 5, улучшения, предложенные в пунктах 4.1, 4.2 (применение новой функции потерь отмечено как "+ focal loss", а использование обоих улучшений – как "+ enhance"), повышают качество классификации. Наибольшее увеличение показателей предложенные улучшения дают самой базовой roberta-base, повышая Token F1 на 16.3%, а метрики по сущностям в среднем на 7.7%. На примере codebert-base видно, что предложенные улучшения могут снизить результаты по некоторым метрикам (Strict F1 −3.2%). Исходя из результатов экспериментов видно, что использование претокенизации не всегда положительно влияет на качество классификации. Такое поведение можно объяснить необходимостью прописывать собственные правила вспомогательного разбиения на токены для каждой модели. Также можно отметить, что применение предобученной модели на текстах из соответствующей предметной области также повышают результаты: PretrainedCodeBert имеет лучше результаты, чем codebert-base, а BertOverflow превосходит roberta-base.

Табл. 5. Результаты тестирования моделей многоклассовой классификации
Table 5. Results of multi-class classification models testing

Модель	Token F1	Type F1	Strict F1
roberta-base	0.5913	0.6741	0.6145
roberta-base + focal loss	0.6692 (+13.2%)	0.7111 (+5.5%)	0.6543 (+6.4%)
roberta-base + enhance	**0.6879 (+16.3%)**	**0.7253 (+7.6%)**	**0.6628 (+7.8%)**
BertOverflow	0.6483	0.6963	0.6428
BertOverflow + focal loss	0.6983 (+7.7%)	0.7108 (+2.1%)	0.6606 (+2.7%)
BertOverflow + enhance	**0.7091 (+9.3%)**	**0.7116 (+2.2%)**	**0.6617 (+2.9%)**
codebert-base	0.6212	0.7414	0.7192
codebert-base + focal loss	0.6836 (+10%)	**0.7467 (+0.7%)**	**0.7193 (+0%)**
codebert-base + enhance	**0.6943 (+11.7%)**	0.7464 (+0.6%)	0.6962 (-3.2%)
PretrainedCodeBert	0.6414	0.7528	0.6954
PretrainedCodeBert + focal loss	0.7219 (+12.5%)	**0.7699 (+2.3%)**	**0.7246 (+4.2%)**
PretrainedCodeBert + enhance	**0.7342 (+14.4%)**	0.7695 (+2.2%)	0.7235 (+4.0%)

7.2 Вспомогательные бинарные модели

В данной секции описаны результаты экспериментов с обучением дополнительных моделей бинарной классификации, описанных в разделе 5. Для обучения бинарной модели использовались только два класса [«O», «Entity»] без BIO нотации. Качество считалось по метрикам Точность, Полнота, F-мера по токенам. Использовались те же данные, что и для многоклассовой классификации с заменой меток всех сущностей на «Entity».

Табл. 6 содержит результаты тестирования бинарных моделей. Все классификаторы, основанные на архитектуре BERT, имеют примерно равные результаты, с небольшим отрывом лучше показал себя классификатор с PretrainedCodeBert. Контекстно-независимая модель Recognizer показала также неплохой результат в 0.8081 F-меры, однако она сильно отстает от контекстно-зависимых классификаторов.

Табл. 6. Результаты тестирования моделей бинарной классификации
Table 6. Results of binary classification models testing

Модель	F-мера	Точность	Полнота
Recognizer	0.8081	0.7883	0.8292
roberta-base	0.9496	0.9462	0.9531
BertOverflow	0.9404	0.9328	0.9486
CodeBert	0.9414	0.9325	0.9514
PretrainedCodeBert	**0.9541**	**0.9498**	**0.9585**

7.3 Ансамбли моделей

Далее показаны результаты экспериментов объединения дополнительных моделей с основной, описанные в разделе 5.3.

В табл. 7 представлены результаты тестирования различных конфигураций объединения моделей. В качестве основной модели использовалась roberta-base с улучшениями, описанными в 4.2, 4.3 (в табл. 7 обозначена как Base), как показавшая наибольшую чувствительность к улучшениям в экспериментах из раздела 7.1. Дополнительные модели: контестно-зависимая Segmenter с базовой моделью roberta-base (в табл. 7 – Seg) и контекстно-независимая Recognizer (в табл. 7 – Reco).

Табл. 7. Результаты тестирования конфигураций ансамблей моделей
Table 7. Results of testing of model ensembles configurations

Модель	Token F1	Type F1	Strict F1
Base	0.6879	0.7253	0.6628
Base + Reco -Emb	0.6872	0.7053	0.6426
Base + Seg -Emb	0.6948	0.7251	0.6783
Base + Seg -Emb 500_31	0.6773	0.7199	0.6782
Base + Seg+Reco -Emb	0.6998	0.7204	0.6825
Base + Seg+Reco -Emb 500_31	0.6791	0.7142	0.6662
Base + Seg -T	0.7127	0.7342	0.6873
Base + Reco -T	0.6073	0.6439	0.6032
Base + Seg||Reco -T	0.6227	0.6424	0.6054
Base + Seg&&Reco -T	**0.7211**	**0.7395**	**0.6954**

Запуски с пометкой «-Emb» обозначают конкатенацию векторных представлений, генерируемых дополнительными моделями, с векторным представлением от Base. Пометка «500_31» означает наличие дополнительного промежуточного линейного слоя. В результате конкатенации векторных представлений их размер на каждый токен составлял (768 + 768 =) 1536 для «Base + Seg –Emb» и (768 + 768 + 20 =) 1556 для «Base + Seg+Reco –Emb».

Как видно из табл. 7, использование двух дополнительных моделей дает больший прирост качества, чем использование этих моделей по отдельности. Хотя по метрике Type F1 больший результат имеет запуск с использованием только Segmenter. Также можно отметить, что применение дополнительного промежуточного линейного слоя размерности 500 только ухудшает результаты классификатора.

Запуски с пометкой «-T» обозначают объединение моделей методом Classify-Trust, в котором вспомогательные модели принимаются оракулами, отвечающими на вопрос: "является ли данный токен не сущностью (принадлежит данный токен классу «O»)?". В случае положительного ответа оракула, токен отмечается как «O», иначе смотрится предсказание многоклассовой классификации. В случае применения двух бинарных моделей-оракулов их результаты можно либо объединять (если хоть один из оракулов дает положительный ответ, в табл. 7 обозначен как ||), либо пересекать (только если оба оракула дают положительные ответы, в табл. 7 обозначен как &&).

Исходя их данных табл. 7 можно сделать вывод, что, применяя метод Classify-Trust, также лучше использовать обе вспомогательные модели, чем каждую их них по отдельности. Стоит также отметить, что запуски Reco и Seg||Reco значительно хуже остальных, так как получается большое количество ложных срабатываний оракулов и значительное количество токенов неверно отмечаются как «O». В отличии от Seg&&Reco, где наоборот метка «O» оракулом выдавалась более точно.

В итоге можно отметить, что в наших экспериментах метод Classify-Trust показывает себя лучше, чем конкатенация векторных представлений. Лучший результат показал запуск "Base + Seg&&Reco -T", использующий обе вспомогательные модели.

Реализованные методы были протестированы на наборе данных из статьи SoftNER, а точнее тех данных, которые находятся в открытом доступе[8]. Среди опубликованных данных содержится обучающая и тестовая выборки с разметкой на 28 классов в IOB2 нотации. Также есть обучающий набор с сокращенным набором меток классов, однако тестового набора с соответствующей разметкой нет. Табл. 8 содержит результаты тестирования. Видно, что по

[8] https://github.com/jeniyat/StackOverflowNER/tree/master/resources/annotated_ner_data/

всем трем метрикам наилучший результат показал классификатор со всем предложенными улучшениями.

Табл. 8. Результаты тестирования реализованных классификаторов на наборе SoftNER
Table 8. Results of testing the implemented classifiers on the SoftNER dataset

Модель	Token F1	Type F1	Strict F1
roberta-base	0.326	0.4806	0.3923
roberta-base + enhance	0.4971	0.5472	0.4666
roberta-base + enhance -Emb	0.4836	0.5607	0.494
roberta-base + enhance -T	**0.5066**	**0.5716**	**0.4984**

Лучшая конфигурация для объединения моделей "Base + Seg && Reco -T" была применена ко все базовым моделям. Результаты тестирования приведены в табл. 9. В каждой секции табл. 9 представлены:

a) результаты базовой модели с улучшениями,
b) классификатор с дополнительными моделями методом Classify-Trust (обозначено как «-T»),
c) случай идеального оракула методом Classify-Trust (обозначено «-T*»).

Для всех классификаторов запуски, обозначенные «-T», показывают лучшие результаты, чем базовые.

В силу архитектуры классификатора с методом Classify-Trust достаточно легко провести эксперимент с идеальным оракулом бинарной классификации (в табл. 9 запуски, отмеченные «-T*»), где вместо текущих классификаторов будет использоваться истинное значение метки токена. Этим экспериментом можно получить максимально возможный прирост показателей от применения метода Classify-Trust. По всем базовым моделям идеальный оракул добавляет от 5 до 8 пунктов F-меры.

В итоге, наилучший результат показал классификатор с базовой предобученной моделью PretrainedCodeBert, с улучшениями, предложенными в пунктах 4.2, 4.3, и со вспомогательными моделями, примененными методом Classify-Trust, получив результаты в Token F1 = 0.7347, Type F1 = 0.7703, Strict F1 = 0.7290.

Табл. 9. Результаты тестирования ансамблей моделей
Table 9. Results of ensembles of models testing

Модель	Token F1	Type F1	Strict F1
roberta-base	0.6879	0.7253	0.6628
roberta-base -T	**0.7211**	**0.7395**	**0.6954**
*roberta-base -T**	*0.7464*	*0.7805*	*0.7327*
BertOverflow	**0.7091**	0.7116	0.6617
BertOverflow -T	0.6965	**0.7467**	**0.6999**
*BertOverflow -T**	*0.7732*	*0.7930*	*0.7586*
codebert-base	0.6943	0.7464	0.6962
codebert-base -T	**0.7172**	**0.7685**	**0.7175**
*codebert-base -T**	*0.7517*	*0.8153*	*0.7629*
PretrainedCodeBert	0.7342	0.7695	0.7235
PretrainedCodeBert -T	**0.7347**	**0.7703**	**0.7290**
*PretrainedCodeBert -T**	*0.7866*	*0.8263*	*0.7917*

7.4 Ошибки извлечения именованных сущностей

На рис. 4 изображена матрица несоответствий классификации лучшей из полученных моделей (PretrainedCodeBert). Для удобства просмотра в результатах тестирования были объединены классы B-Tag и I-Tag, таким образом получили матрицу 16 x 16, вместо 31 x 31. В матрице приведены нормированные значения по истинным меткам (по горизонтали сумма должна равняться единице, с точностью до округления).

При ручном анализе примеров, на которых ошибается классификатор, можно выделить несколько групп ошибок.

Ошибки в подтокенах. Стоит заметить, что классификация происходит на уровне подтокенов, которые получаются после токенизатора RoBERTa, и разработанные улучшения в этом направлении (пункт 4.2) не всегда помогают. Иногда изначальные длинные сущности разбиваются на множество маленьких подтокенов, каждый из которых модели необходимо классифицировать. Для борьбы с этой проблемой есть пара подходов: а) поменять способ обучения так, что в подсчете функции потерь и при итоговой классификации будет учитываться только первый подтокен; б) соединять итоговые предсказания модели по всем подтокенам путем некоторого голосования для единой оценки всей сущности.

Ошибки с классом «O». Они являются самыми распространенными: из 924 некорректно классифицированных токена 523 ошибки связаны с классом «O». Из рис. 4 можно заметить, что 28% «Extenral_tool» классифицировались как «O». Например, в комментарии "*I ran `make lint` and `make pylint` before pushing this commit*." токен "make" не распознался как сущность. Или в примере *"* return just empty ColumnNothing otherwise*" токен "empty" распознался как Keyword, хотя таковым не является.

Ошибки классификации между классами. Самой понятной можно назвать пару Variable и Function (81 ошибочно классифицированный токен). Так, например, в комментарии "*So I think I've got the current fastest implementation using bytesBefore and the underlying SWAR indexOf, thanks to you! ;-)*" токены "bytes" и "Before" модель классифицировала как Function, но правильным является класс Variable. Или в примере "*I pushed a change to use `np.newaxis` whenever possible.*" токены "new" и "axis" определились как Function при верном ответе Variable. Ещё одной часто путающейся парой является Variable и Value (53 взаимных ошибки). Так как к Value мы относили строковые константы, которые зачастую обрамлены кавычками, то возникают следующие ситуации: в комментарии "*'propertyIdWithAreaIds' I think it is better to be more clear with the name.*" propertyIdWithAreaIds классифицировался как Value, но является именем переменной.

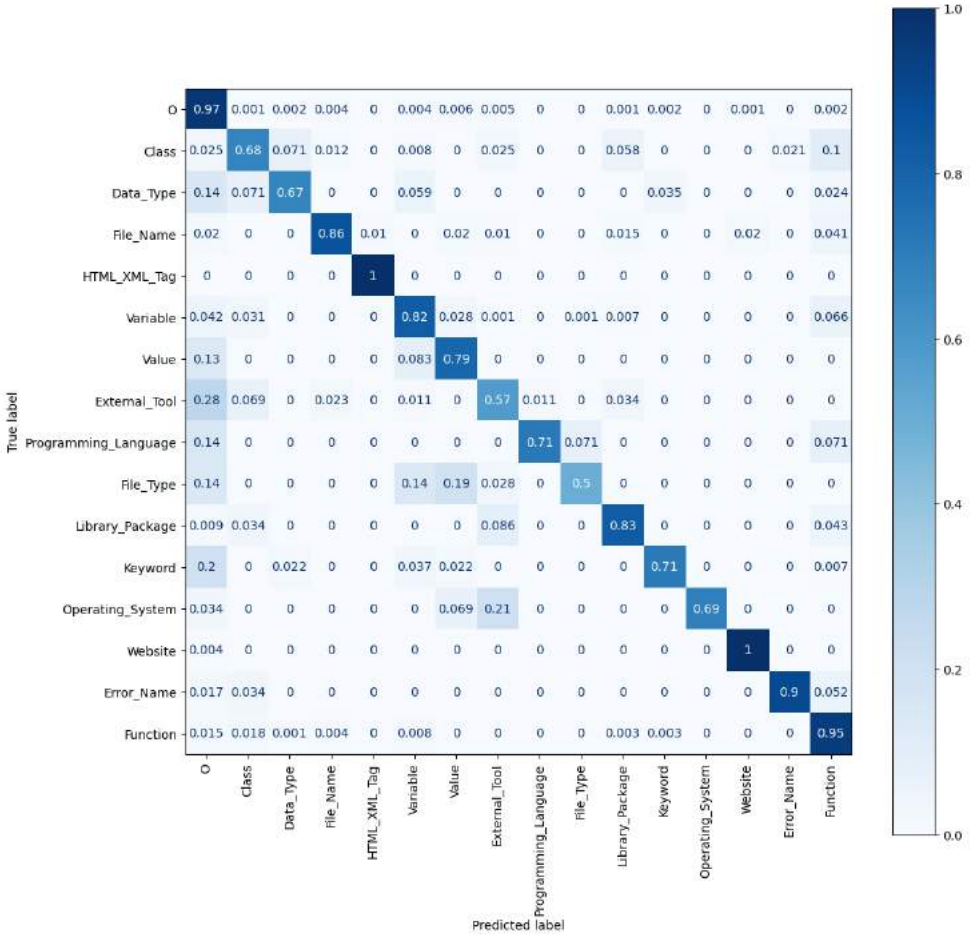


Рис. 4. Матрица несоответствия.
Fig. 4. Confusion matrix.

8. Заключение

В данной работе были исследованы методы повышения качества решения задачи извлечения именованных сущностей в применении к области разработки программного обеспечения. Проведено сравнение реализованных подходов на новом собранном и размеченном вручную наборе из 3000 рецензий, оставленных пользователями на изменения в исходном коде. Количество распознаваемых классов – 15. Предложенные и реализованные методы повышения качества классификации позволили поднять результаты по разным метрикам на 8-13 пунктов F-меры. Ручной запуск показал достаточно хорошее качество классификации для применения инструмента в прикладных задачах. Разработанный инструмент поиска и классификации сущностей позволит лучше решать задачи классификации/кластеризации и семантического поиска комментариев из предметной области.

Список литературы / References

[1]. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: 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). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). DOI:10.18653/v1/N19-1423.

[2]. Tabassum, J., Maddela, M., Xu, W., Ritter, A.: Code and named entity recognition in StackOverflow. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 4913–4926. Association for Computational Linguistics, Online (Jul2020). DOI:10.18653/v1/2020.acl-main.443.

[3]. Rahul Sharnagat, Named Entity Recognition: A Literature Survey, June 30, 2014 https://www.cfilt.iitb.ac.in/resources/surveys/rahul-ner-survey.pdf.

[4]. Bikel, D.M., Schwartz, R. & Weischedel, R.M. An Algorithm that Learns What's in a Name. Machine Learning 34, 211–231 (1999). DOI:10.1023/A:1007558221122.

[5]. John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289.

[6]. Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4 (CONLL '03). Association for Computational Linguistics, USA, 188–191. DOI: 10.3115/1119176.1119206.

[7]. Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995). DOI:10.1007/BF00994018.

[8]. McNamee, P., & Mayfield, J. (2002). Entity Extraction without Language-Specific Resources. Conference on Computational Natural Language Learning.

[9]. Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270, San Diego, California. Association for Computational Linguistics. DOI: 10.18653/v1/N16-1030.

[10]. Batbaatar E, Ryu KH. Ontology-Based Healthcare Named Entity Recognition from Twitter Messages Using a Recurrent Neural Network Approach. International Journal of Environmental Research and Public Health. 2019; 16(19):3628. DOI:10.3390/ijerph16193628.

[11]. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[12]. Cedric Lothritz, Kevin Allix, Lisa Veiber, Tegawendé F. Bissyandé, and Jacques Klein. 2020. Evaluating Pretrained Transformer-based Models on the Task of Fine-Grained Named Entity Recognition. In Proceedings of the 28th International Conference on Computational Linguistics, pages 3750–3760, Barcelona, Spain (Online). International Committee on Computational Linguistics. DOI: 10.18653/v1/2020.coling-main.334.

[13]. Tikhomirov, M., Loukachevitch, N., Sirotina, A., Dobrov, B. (2020). Using BERT and Augmentation in Named Entity Recognition for Cybersecurity Domain. In Natural Language Processing and Information Systems. NLDB 2020. Lecture Notes in Computer Science, vol 12089. Springer, Cham. https://doi.org/10.1007/978-3-030-51310-8_2.

[14]. Malik, G., Cevik, M., Bera, S., Yildirim, S., Parikh, D., & Basar, A. (2022). Software requirement specific entity extraction using transformer models. Proceedings of the Canadian Conference on Artificial Intelligence. DOI:10.21428/594757db.9e433d7c.

[15]. D. Ye, Z. Xing, C. Y. Foo, Z. Q. Ang, J. Li and N. Kapre, "Software-Specific Named Entity Recognition in Software Engineering Social Content," 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Osaka, Japan, 2016, pp. 90-101, DOI:10.1109/SANER.2016.10.

[16]. Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A Robustly Optimized BERT Pre-training Approach with Post-training. In Proceedings of the 20th Chinese National Conference on Computational Linguistics, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

[17]. Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1536–1547, Online. Association for Computational Linguistics. DOI: 10.18653/v1/2020.findings-emnlp.139.

[18]. T. -Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 2, pp. 318-327, 1 Feb. 2020, DOI:10.1109/TPAMI.2018.2858826.

[19]. Anderson, P.E., Reo, N.V., DelRaso, N.J. et al. Gaussian binning: a new kernel-based method for processing NMR spectroscopic data for metabolomics. Metabolomics 4, 261–272 (2008). DOI:10.1007/s11306-008-0117-3.

[20]. Charlie Xu, Solomon Barth, Zoe Solis. Applying Ensembling Methods to BERT to Boost Model Performance. Stanford University. Accessed at: 01.11.2023.

[21]. Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), pages 1–9, Atlanta, Georgia, USA. Association for Computational Linguistics.

Информация об авторах / Information about authors

Владимир Владимирович КАЧАНОВ – аспирант. Сфера научных интересов: машинное обучение, программная инженерия.

Vladimir Vladimirovich KACHANOV – postgraduate student. Research interests: machine learning, software engineering.

Ариана Сергеевна ХИТРОВА – бакалавр. Сфера научных интересов: машинное обучение, обработка естественного языка.

Ariana Sergeevna KHITROVA – bachelor. Research interests: machine learning, natural language processing.

Сергей Игоревич МАРКОВ – специалист, старший научный сотрудник. Сфера научных интересов: статический анализ кода, динамический анализ кода, программная инженерия, машинное обучение.

Sergei Igorevich MARKOV – specialist, senior researcher. Research interests: static program analysis, dynamic program analysis, software engineering, machine learning.

DOI: 10.15514/ISPRAS-2023-35(5)-14

CC BY

Here We Go Again: Modern GEC Models Need Help with Spelling

V.M. Starchenko, ORCID: 0009-0004-6638-9124 <*vstarchenko@hse.ru*>
A.M. Starchenko, ORCID: 0000-0003-1650-7597 <*aleksey-starchenko@mail.ru*>
HSE University,
20, Myasnitskaya st., Moscow, 101000 Russia

Abstract. The study focuses on how modern GEC systems handle character-level errors. We discuss the ways these errors effect the performance of models and test how models of different architectures handle them. We conclude that specialized GEC systems do struggle against correcting non-existent words, and that a simple spellchecker considerably improve overall performance of a model. To evaluate it, we assess the models over several datasets. In addition to CoNLL-2014 validation dataset, we contribute a synthetic dataset with higher density of character-level errors and conclude that, provided that models generally show very high scores, validation datasets with higher density of tricky errors are a useful tool to compare models. Lastly, we notice cases of incorrect treatment of non-existent words on experts' annotation and contribute a cleared version of this dataset. In contrast to specialized GEC systems, LLaMA model used for GEC task handles character-level errors well. We suggest that this better performance is explained by the fact that Alpaca is not extensively trained on annotated texts with errors, but gets as input grammatically and orthographically correct texts.

Keywords: GEC, validation; spellcheck; preprocessing; generated datasets.

For citation: Starchenko V.M., Starchenko A.M. Here We Go Again: Modern GEC Models Need Help with Spelling. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 215-228. DOI: 10.15514/ISPRAS-2023-35(5)-14.

Acknowledgements. This work is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University) in 2023. We are grateful to Anastasia Vyrenkova and Olga Lyashevskaya for extensive discussion and comments on the paper, to Olga Vinogradova for help with the research in general. All possible errors made in the study are exclusively on us.

Проблема валидации современных систем исправления грамматических ошибок: случай ошибок на уровне символов

В.М. Старченко, ORCID: 0009-0004-6638-9124 <*vstarchenko@hse.ru*>
А.М. Старченко, ORCID: 0000-0003-1650-7597 <*aleksey-starchenko@mail.ru*>
Национальный исследовательский университет «Высшая школа экономики»,
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. Исследование сосредотачивается на проблеме того, как современные системы исправления грамматических ошибок обрабатывают ошибки на уровне слова. Работа обсуждает, как подобные ошибки могут взаимодействовать с эффективностью модели, и оценивает, как модели с разными архитектурами справляется с ними. Делается вывод о том, что специализированные системы исправления грамматических ошибок сталкиваются с проблемами при исправлении ошибок,

приводящих к созданию несуществующих слов, и что предобработка с помощью простой системой обработки подобных ошибок значительно улучшает общую эффективность модели. Для оценки этого работа модели тестируется для нескольких валидационных датасетах. Вдобавок к валидационному датасету соревнования CoNLL-2014 в работе предлагается синтетический датасет с повышенной плотностью ошибок на уровне слова. На основании сравнения эффективности модели на двух датасетах, работа делает вывод о том, что валидационные датасеты с высокой плотностью ошибок, представляющих проблему для моделей, — это полезный инструмент для сравнения моделей. Кроме того, работа указывает на случаи некорректной аннотации несуществующих слов в разметке экспертов и предлагает очищенную версию датасета. В отличие от специализированных систем исправления грамматических ошибок, модель LLaMA, используемся для задачи исправления грамматических ошибок хорошо справляется с ошибками на уровне слова. Мы предполагаем гипотезу, в соответствии с которой этот результат объясняется тем фактом, что эта модель не обучается на специальной аннотированной выборке, содержащей ошибки, а получает в качестве входа грамматически и орфографически корректные тексты.

Ключевые слова: автоматическое исправление грамматических ошибок; валидация; спеллчек; предобработка; синтетические датасеты.

Для цитирования: Старченко В.М., Старченко А.М. Проблема валидации современных систем исправления грамматических ошибок: случай ошибок на уровне символов. Труды ИСП РАН, 2023, том 35 вып. 5, с. 215-228 (на английском языке). DOI: 10.15514/ISPRAS–2023–35(5)–14.

Благодарности. Исследование осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ в 2023 году. Мы благодарны А. С. Выренковой и О. Н. Ляшевской за подробные комментарии к работе, О. Н. Виноградовой за помощь в исследованиях. Все возможные ошибки, которые могли остаться в настоящей работе, исключительно на нашей совести.

1. Introduction

Tools for GEC (Grammatical error correction) tasks have greatly improved over recent decades. In terms of metrics, modern big language models outperform a human annotator in the GEC task [1]; overviews [2-4] present the performance growth at different stages. GEC models are however still noticed to fail in correcting several types of errors that would be easily and necessarily corrected by a human [1].

Despite the part “grammatical” in GEC, the task is usually understood wider than the mere correction of illicit grammar use. As the expected result is a text judged natural by a native speaker, spelling, punctuation, word choice, stylistic and other types of errors are treated, as well.

One must note that the best-performing modern models for GEC show pure results with character-level errors, and this problem had been preserved during the last decade [5-7]. If the errors ranked according to their difficulty, this type is considered one of the easiest to correct [8].

Consider a spelling error in Table 1, which the GECToR model [9] fails to correct (*diag**o**nosed* instead of *diagnosed*). In contrast, several other errors are successfully handled, including article use, word form selection and phrasal verbs. Notice that the error in Table 1 is not challenging to detect because it results in a non-existent word, and the closest candidate in terms of Levenshtein distance is a required one. This type of errors is effectively handled with a number of tools performing with a quality acceptable for practical use for a very long time [10-11].

Table 1. Example of the failure and successes of a GEC model

source	When we are **diagonosed** out with certain genetic disease , are we suppose to disclose this result to our relatives ?
corrected = target	When we are **diagonosed** with a certain genetic disease, are we supposed to disclose this result to our relatives ?

Although some researchers apply spellcheckers or character-based models as a part of preprocessing [12-13], [7], [14], it is still not a common practice for modern GEC. For example, the possibility of preprocessing of spelling is not discussed in the recent detailed overview of approaches to GEC [4].

In this study we focus on how character-level, primarily spelling errors affect the output of best-performing GEC models with different architecture. We test 3 SOTA GEC systems with different architecture: GECToR (large) [9], BART (large) [15], and T5 (base) [16]. We also add LLaMA 7B model [17-18] fine-tuned as Alpaca 7B [19]. Large language models like LLaMA or GPT [20] have been recently tested for multiple tasks, including GEC, though at the moment they exhibit lower performance than other models [21][1] show for English[2]. Of other modern SOTA GEC system we do not separately discuss the SynGEC [23], as the cited study shows that the innovations introduced by this complex model to regular transformer-based baseline / BART worsen the performance on spelling errors [23: 2525]. The evaluation of GEC-DI [24], which has been very recently released and suggested to us by one of anonymous reviewers, we leave to the further studies.

Table 2 presents performance of the models, evaluated on the validation dataset for CoNLL-2014 [25] with annotation by 10 experts [26]. F0.5 metric is used, which is argued to represent human judgments well [27-29].

Table 2. Performance of SOTA GEC systems and human experts

model name	$F_{0.5}$
BART	78.04
GECToR	76.82
T5	74.38
LLaMA	68.58
human experts	72.58

Three best-evaluated models: BART, GECToR and T5, show higher scores than human experts do with respect to each other. Yet, we are going to confirm that they perform imperfectly with character-level errors.

Elaborating on the nature of the spelling pitfall of big language models, we notice that both training and especially validation datasets for GEC tasks are noisy when dealing with character-level errors. We contribute a cleared version of CoNLL-2014 validation dataset [25] (its 10-annotators version [26]) and a synthetic dataset with a higher density of spelling errors (but also including all other types of errors), which can be used for testing the impact of this kind of errors. We further suggest that such datasets with high error density are a useful tool to test models that generally show very high performance over tricky types of errors.

Based on both datasets we show that all the tested models designed specifically for GEC show higher performance with spelling errors corrected at the preprocessing stage. In contrast, results of LLaMA model, despite its purer performance in general, is almost not affected with the preprocessing.

The rest of the paper is organized as follows: Section 2 describes the ways in which character-level errors interact with the performance of models. Section 3 discusses the representation and the source of this kind of errors in training and validation data. Section 4 presents three datasets we work with. Section 5 presents an experiment that evaluates the influence of preprocessing of spelling errors on the performance, based on these datasets. Sections 6 interprets the experiments and presents the discussion. Section 7 is the conclusion.

[1]We use a prompt different from the one suggested for GPT 3.5/4 in [21], as our prompt gives a higher score: $F_{0.5}=68.58$ compared to $F_{0.5}=64.34$. The used prompt is: *You will receive a text in English and you must check whether it contain any errors, according to English language rules. Return a corrected version of the text. Don't correct stylistic errors. Do not correct sentences that may be correct in some context. The final text should not contain errors.*

[2]Though s.f. [22] for Swedish, for which GPT 3 outperforms all other models.

2. Relationship between model performance and misspellings

In this section we discuss how character-level error may affect the performance of a model based on the CoNLL-2014 validation dataset.

We only restrict ourselves to the errors resulting in non-existent words. In most cases such errors result from misprints (*wth* instead of *with*, *otherm* instead of *other*), problems with spelling orthographically difficult words (*hypertesion* / *hypertention* instead of *hypertension*, *percieved* instead of *perceived*) and the influence of the native language of an author (e. g. insertion of *o* in consonants clusters by Singapore students: *tech**o**nology*, *diag**o**nosed*). Rarer an error emerges as a result of a morphological process including creating a non-existent word form (plural *medias* instead of *media*) or derivation (*disclosement* instead of *disclosure*). Supposed misprints that lead to the use of an existing word which does not fit the context (*brunch* instead of *bunch*) are considered as word choice errors and therefore are not discussed.

Noticeably, in most cases relevant errors of the considered dataset are not ordinary misprints. More than in half cases, misspelling result from inability of a speaker to deal with phonology—orthography incongruity or forming a wrong morphological pattern, rather than from their inaccuracy. While some of these misspellings still do not fall under the narrow definition of grammatical errors (that is, related to ungrammaticality), they can definitely be viewed as a part of the language system (undertrained mental phonological—orthographic interface) and not an accidental typesetting problem.

Proceeding to the interaction of the character-level errors and model performance, trivially, a model may fail to correct a spelling error (Table 3, BART [15]) and leave a incorrectly spelled word as it is.

Table 3. Example of a simple spelling error

source & corrected	However , it is a good practice not to **intesively** use social media all the time .

In some cases, the model tries to deal with a misspelling, but fails to fix it correctly. In the example in Table 4 by BART [15], instead of inserting the missing letter *r* into the word *concurrently*, the model replaces the whole word, which leads to a semantic distortion of the source sentence. Preprocessing of the text with a spellchecker prevents the model from this alteration.

Table 4. Example of a spelling error leading to semantic distortion

source	… he or she **concurently** has a knowledge about others .
corrected	… he or she **definitely** has knowledge about others .
spellcheck + corrected	… he or she **concurrently** has knowledge about others .

Lastly, a misspelling may affect processing of other types of errors, either close or long-distance. In the example in Table 5 by GECToR [9], in addition to the leaving the misspelling in the word *dilenma*, the output of the model contains collocation *feel into*, which is syntactically related to the word *dilemma* and infelicitous in the context. Furthermore, the model does not handle the word *reflects*, which is semantically incorrect and stands further away from the misspelling in the sentence. If the misspelling is corrected prior to model application, both *reflect* and *feel into* are handled better, although the latter case is still an imperfect correction.

Table 5. Example of spelling error interacting with other error types

source	During that period , if one of the family member reflects genetic disorder symptoms , he will fell in an ethical **dilenma** for sure .
corrected	During that period , if one of the family members reflects genetic disorder symptoms , he will feel into an ethical **dilenma** for sure .
spellcheck + corrected	During that period , if one of the family members has genetic disorder symptoms , he will feel in an ethical **dilemma** for sure .

Summing up, the output of the discussed GEC model is influenced by spelling errors at different levels, starting from the mere inability to handle spelling and up to preventing correction of other errors at the scale of the whole sentence. Similar distortions can be noticed for other considered models, as well.

In the next section we suggest an explanation for this notorious inability to correct character-level errors, which at least partially lies in the annotation for training and validation datasets.

3. Noise in training and validation data

3.1 Validation data

CoNLL-2014 dataset contains 137 character-level errors that produce non-existent words. Despite elaborate and thorough annotation of different types of errors in other domains, the coverage of character-level errors producing non-existent words in the annotation is not high. Of all them, 94 were missed by at least one annotator, 266 cases of unspotted misspellings were found in total. It means that more than a half of character-level errors cannot be accounted properly during the evaluation process.

Table 6. Number of uncorrected non-existent words grouped by number of annotators

annotators	1	2	3	4	5	6	7	8	9	total
errors	32	23	10	9	8	5	5	1	1	94

The detailed statistics on how many errors were missed by the annotators is presented in Table 6. One can notice that some errors are remarkably stealthy, unnoticed by most of annotators. The highest scores are: 9 annotators, *newpaper — newspaper*; 8 annotators *techonology — technology*; 7 annotators *subconsiously — subconsciously*, *covenient — convenient*, *againt — against*, *simliar — similar*, *acccount — account*.

The high quantity of non-annotated errors poses a problem for validation. This problem is especially significant due to the validation procedure, used in the setup with multiple annotations [26].

In order to account for multiple annotations by multiple experts, the output of the model is evaluated over all annotations and the highest $F_{0.5}$-score is assigned to the model.

This approach is suggested to capture the cases in which annotators correct an error in different yet equally grammatical ways. Otherwise, the availability of paraphrases with similar meanings could not be properly accounted for.

However, in case of inaccuracy in the annotation, all other equal, it is the erroneous target sentence that receives the highest score for the model that does not make a correction. Thus, presence of both correct and incorrect options among annotations yields to indistinguishability of correctly and incorrectly working models.

To avert this problem, we contribute a new version of the dataset, in which all listed cases of missed character-level errors are manually added to the annotation[3].

Notice that such corrections may not be considered as interfering with a personal choice of a rarer yet well-formed construction by the expert. All the corrected words are not found in any dictionary of Standard English (and are not among commonly used spellings that are not yet represented in dictionaries) and therefore must be viewed as overlooked unintentionally. The exact inserted annotations, including the error type, are based on the most frequent option among annotators who corrected a particular error.

[3]Additionally, rare occasions of misspellings in the corrections suggested by annotators were fixed.

3.2 Training data

Having found from the validation dataset that experts often disregard spelling errors, one could suppose that training datasets must show even more noise of this type. Training datasets are much larger and are often annotated much less thoroughly (for example, one of frequently used training datasets cLang-8 [30-31] was annotated by language learners).

In contrast, training datasets appear to be much clearer with respect to character-level errors.

To estimate it, we check the training part of the FCE dataset [32], which contains about 18k sentences from texts by ESL learners, annotated by a single expert. For a random subset of the dataset, we automatically located all words that are marked as non-dictionary ones and then manually annotated about whether they contain an error.

Table 7 compares the number of uncorrected character level errors in training and validation datasets and the density of sentences with character-level errors (both corrected and uncorrected).

Table 7. Number of uncorrected character-level errors and the overall density of character-level errors in training and validation datasets

	uncorrected errors, % of all character-level errors	**% of sentences with character-level errors of all sentences**
FCE, training	3%	13.5%
CoNLL-2014[4], validation	19% per annotator 69% for 10 annotators	8.6%

Table 7 shows that in the validation dataset, character-level errors are not included in the annotation much more often than in the training dataset. Consequently, even if this kind of errors is corrected well by the model, it will be difficult to evaluate when compared with models that demonstrate lower performances. This difficulty is enlarged with a low density of character-level errors in the dataset (only 8.6% of sentences in the dataset have them).

On the other hand, some proportions of uncorrected character-level errors are included in the training dataset and thus may impact performance of the model.

3.3 The source of noise and a way to prevent it

Spelling errors are in most cases easy to correct. If there are no errors of other types or poor word choice in a fragment, annotators perfectly agree on how misspellings must be corrected. The main problem is to notice them.

It is known that a person does not read familiar words letter by letter, but processes words or at least parts of words as a whole [33], [34]. For this reason, slight distortions of visual appearance of words are not necessarily perceived while regular reading. In contrast, grammatical errors which lead to infelicity at the sentence-level are expected to be conceived during regular reading easier.

Therefore, correcting spelling errors practically requires from an expert to process the text twice, performing both natural reading and the other task, which due to its untypicality takes more effort.

It is natural that without constant conscious effort even an utmost high-skilled reader is going to miss some of character-level errors. Provided that an expert is also expected to annotate other types of errors, the doing so is inevitable.

As for the testing datasets, one can notice that because of their big sizes and costly annotation process, they are usually partially automatically annotated, which naturally includes spellcheck. As a result, the percentage of non-annotated character-level errors in the training dataset is lower.

[4]Based on the dataset with 10 annotators, missed errors per one annotator are calculated as total number of errors divided by number of annotators: *266/10=26.6*. The value of 69% for 10 annotators is calculated for errors that were missed by at least one annotator.

The problem with the higher can be fixed by adding a simple spellchecker that locates non-existent words in am annotator's interface. It must not be a more complicated tool, as it may interfere with the way an expert corrects errors.

4. Datasets

4.1 Introduction of three datasets

Further on, we will focus on the problem of character-level errors in validation data. We elaborate on the problem of partially correct annotation of the validation dataset. Then we proceed to the problem of low density of a particular type of errors in a dataset, which does not allow to test a model over this type of errors properly.

As discussed in Section 3.1, we build on the validation dataset for the CoNLL-2014 [25] and use its version that was independently annotated by 10 experts [26]. Henceforth, we call it Original dataset.

By correcting inaccuracies in annotation, discussed in Section 2, we create **Corrected dataset**[5].

Lastly, we generate **Synthetic dataset**[6] with a higher density of spelling errors in order to highlight the trends that emerge from comparing Corrected and Original datasets.

4.1 Generation of Synthetic dataset

We build on the CoNLL-2014 validation dataset, rather than create a new one with only synthetically induced errors, to capture the interaction of character- and word-level errors, described in Section 2. We preserve all non-character-level errors, in order to capture their interaction with the character-level ones.

We rely on the algorithm for generating datasets suggested in [35], that is, probabilistically introduce spelling errors in the source sentences at a rate of 1–3 per sentence, randomly selecting deletion, insertion, replacement, or transposition of adjacent characters for each introduced error.

The new density of character-level errors is much higher than the one in Original dataset. Yet it is not unrealistically high and does not make texts impossible to understand.

Corrections to all errors induced into the dataset were added to each annotation.

5. Performance of the models on character-level errors with different validation datasets

5.1 Experiment 1: Original and Corrected datasets

Table 8. Performance of four models on the original and corrected validation datasets, with and without preprocessing

Model name	original dataset		corrected dataset	
	only model, $F_{0.5}$	spellchecker + model, $F_{0.5}$	only model, $F_{0.5}$	spellchecker + model, $F_{0.5}$
BART	78.04	78.47	78.07	78.57
GECToR	76.82	77.03	76.84	77.13
T5	74.38	74.68	74.44	74.81
LLaMA	68.58	68.58	68.77	68.78

[5]drive.google.com/drive/folders/169Xvvgn4eBIhSIzYjPE8YsTL93JwjlB2

[6]drive.google.com/drive/folders/1lruoHhAyTrvMaJniAUz0I6G6H9hF57dP

5.1.1 Setup

Noise in training and validation data affects performance and evaluation of a model.

Noise in the training data is expected to worsen its overall result. In order to evaluate how good a model performs at the task of correcting character-level errors, we compare its metrics on validation to the metrics of the model after preprocessing of the source with a simple spellchecker (its architecture is described in more detail in Section 5.1.2).

Noise in the validation data may not allow us to evaluate the performance of a model properly. This problem is aggravated by the way in which multiple unequal annotations are accounted for (see Section 3.1). This approach neatly captures the possibility of variation between different grammatical options of error correction. However, if the annotation includes an erroneous option, it is not necessarily outweighed by corrections of other annotators. That is, if at least one of the annotators missed an error, everything else equal, it may be this annotation which will be accounted for during validation.

To evaluate the impact of the noise in the validation dataset, we compare performance of models on Original and Corrected datasets. For each model, we separately evaluate the performance of a model on its own and the performance of the system of both a spellchecker as a preprocessing tool and the model.

Before proceeding to the scores of the models, we describe how the spellchecker system is organized.

5.1.2 Spellchecker

A spellchecker used for preprocessing should eliminate non-existent words and not affect the rest of the text. Non-existent words are a type of errors which is handled well by different spelling correction systems [36]. On the other hand, big GEC models under consideration handle non-character-level errors, including word choice and discourse incongruence, well. For this reason, spelling errors that lead to creation of an existent word (bunch — brunch) are not corrected and are left to GEC models. We also do not correct spelling issues related to British / American orthography differences like the contract of -ise and -ize derivational suffixes (organised vs. organized).

In the outlined setup, the most reasonable is a dictionary-based approach, which is not expected to creatively alter the source text.

To enlarge the dictionary, we use multiple available spellcheckers, showing high quality on the task of correcting non-existent words: *hunspell*[7], *autocorrect*[8] and *spellchecker*[9].

5.1.3 Evaluation

Table 8 presents how models perform on the original and corrected validation datasets with and without preprocessing by the spellchecker.

The difference in the evaluation results is not large, which may be expected. 94 changes in 116 of 1342 sentences have been made, so the changes on the third–forth significant digit is reasonable. Despite modest differences of scores, some feasible trends can be noticed.

Firstly, the performance on the original and the corrected datasets either differs just slightly or is higher for the latter (up to $\Delta F_{0.5}=0.19$). Models, for which evaluation grew, are better at correcting spelling errors (see the next section form more detailed discussion) than whose performance didn't change significantly. Provided all that, one can conclude that in the new version of the dataset models are being punished less for correcting non-existent words and exhibit higher scores.

[7]Availabe at: https://pypi.org/project/hunspell/.

[8]Availabe at: https://github.com/filyp/autocorrect/.

[9]Availabe at: https://pypi.org/project/pyspellchecker/.

Noticeably, in combination with spellchecking, Corrected dataset produces consistently higher scores than Original dataset does. This result confirms that the corrected dataset is more sensitive to the character-level errors correction and awards better models that show higher quality with this type of errors.

Secondly, all models except for LLaMA perform better when preprocessed with a spellchecker. The growth of the score is low (0.21–0.5), but consistent and goes along with the hypothesis that these models do not perform well for character-level errors.

This means that applying a simple spellchecker to SOTA GEC models results in higher scores and this improves their overall performance.

Ultimately, we test our models against a synthetic dataset that has a higher density of character-level errors to provide a more robust confirmation of our hypothesis.

5.2 Experiment 2: Synthetic dataset

We present the performance of models with and without spellchecker preprocessing in Table 9.

Table 9. Performance of models with and without preprocessing, dataset with the high density of spelling errors

model	**only model, $F_{0.5}$**	**spellchecker + model, $F_{0.5}$**
BART	76.79	**83.6**
GECToR	75.34	82.3
T5	76.89	81.77
LLaMA	**82.25**	82.68

For Synthetic dataset, scores with and without spellchecker-assisted preprocessing are significantly higher than for Original or Corrected dataset. It is expected, provided that the former contains by far more character-level errors than the latter two.

Evaluation reveals that four models handle character-level errors to a different degree. BART and GECToR, though best performing on the Original and Corrected dataset, lowered their results on the Synthetic dataset. In contrast, adding multiple character-level errors increased the scores in T5 and LLaMA, meaning that the models are better at correcting these types of errors.

Three models: BART, GECToR and T5 show a considerable growth in metrics in combination with spellchecking. It strikingly differentiates them from LLaMA, which is less affected by the spellcheck. While on the original dataset, LLaMA model performs worst, the dataset with the high density of character level errors promotes it on top. The score of the model with and without a spellchecker are almost equal, meaning that LLaMA perfectly handles character-level errors induced into the dataset.

Yet the spellcheck preprocessing allows the BART model to regain its first place. Therefore, for Synthetic dataset, SOTA GEC model combined with a simple spellchecker shows the best performance, while all three specialized GEC models without spelling check perform quite poorly.

6. Discussion

The experiment performed in the previous section confirms the hypotheses made in Section 5.1. Three big models, trained specifically for the GEC task: BART, GECToR and T5 perform worse than a simple spellchecker, when dealing with non-existent words. The experiment also allows to differentiate between these three models: T5 performs on this task better than two other models.

Quite the opposite, LLaMA perfectly deals with character-level errors.

This result contradicts the idea that big word or sentence-level models perform bad with character-level errors, suggested in different studies [5-7].

What distinguishes LLaMA from all other models, making it good in correcting one (or, possibly, some) types of errors, but leaving it with a generally worse score?

One possible explanation is exposure or lack of exposure of a model to special training GEC datasets. Specialized GEC models are trained on grammatically and orthographically incorrect input, partially inaccurately annotated. Errors in annotation may lead to incorrect patterns being learned by a model. In contrast, LLaMA is (mostly) trained on grammatically correct texts produced by native speakers and is not generally expected to produce ungrammatical text if not asked otherwise (and does not do so in our data).

Therefore, LLaMA usually does not preserve or produce wrongly spelled or ungrammatical output. What it is often punished for by the metrics is being over-creative, producing sentences that are too different from the initial ones. If a dataset is annotated by an expert who aims to keep the initial sentence as close to the original as possible, provided its grammaticality, a creative correction is going to receive a lower score. On the other hand, in the task of returning of not just a grammatically correct, but also natively sounding English sentence (for JFLEG dataset [37]), large language models like LLaMA or GPT models may perform better than specialized models [21]. On the other hand, a creative correction produced by such model may become semantically unequal to the source, making this correction erroneous.

Further tuning of the large language models is therefore not aimed at correcting grammatical or orthographic errors in the input, but rather restraining it from changing the source too much.

7. Conclusions

Our study evaluated the performance of modern SOTA GEC systems on character-level errors. We described how this type of errors interferes with the performance of GEC systems and confirmed that they still struggle through handling character-level errors, like they deed over recent decade [5-7]. In contrast to the cited studies, we however notice that not all large language models perform badly with character-level errors: LLaMA, though being worse in the GEC task in general, performs on this particular error type better. We relate this difference to the exposure to annotated ungrammatical texts, which contain noise in training data.

An immediate practical output of the study is the suggestion of performing spellcheck preprocessing as a common practice with GEC models. Some studies do so for English [12-13], [7], [14] and it is a regular practice for Chinese because of the peculiar traits of its graphical system [38-39]. Still, handling character-level errors is not discussed in many recent studies.

In a longer-term perspective, this suggestion cannot be considered most suitable: one could desire for a big language model to correct all kinds of errors, rather than just a spellcheck including system. Suggested steps to achieve this result are to clean training (at least from character-level errors) and validation datasets for specialized models.

The last important result is that the sensitivity of a particular validation dataset may not be sensitive enough to evaluate performance of a model for a particular type of errors. Study [1] lists several types of errors that modern models are unable to handle adequately, including the correction of unnatural phrases, correction of patterns requiring information about sentence structure, and correction of errors that involve inter-sentence relationships. In such cases, to capture the difference in model performance, validation datasets with a higher error density can be done.

In this study, we start with character-level errors, for which a high-density dataset is relatively easy to synthesize and show that it allows to highlight differences in performance of models. This dataset can be used in further studies to report on the results for specifically character-level errors, though without missing the information about their interaction with other types of errors. To obtain such an opportunity for other kinds of errors, more work on collecting natural examples with them or more elaborate synthesizing is to be done.

References

[1]. Qorib M. R., Ng H. T. Grammatical error correction: Are we there yet? In Proceedings of the 29th International Conference on Computational Linguistics. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, 2022, pp. 2794–2800.

[2]. Leacock C., Chodorow M., Gamon M., Tetreault J. Automated Grammatical Error Detection for Language Learners. Morgan & Claypool Publishers, 2014. 154 p.

[3]. Wang Y., Wang Y., Dang K., Liu J., and Liu Z. A comprehensive survey of grammatical error correction. ACM Trans. Intell. Syst. Technol., 12(5), 2021, pp. 1–51. doi: 10.1145/3474840.

[4]. Bryant C., Yuan Z., Qorib M. R., Cao H., Ng H. T., Briscoe T. Grammatical Error Correction: A Survey of the State of the Art. Computational Linguistics, 49 (3), 2023, pp. 643–701. doi: 10.1162/coli_a_00478.

[5]. Susanto R. H., Phandi P., Ng H. T. System combination for grammatical error correction. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014, pp. 951–962. [Online]. doi: 10.3115/v1/D14-1102.

[6]. Rozovskaya A., Roth D. Grammatical error correction: Machine translation and classifiers. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, 2016, pp. 2205–2215. doi: 10.18653/v1/P16-1208.

[7]. Chollampatt S., Wang W., Ng H. T. Cross-sentence grammatical error correction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019, pp. 435–445. doi: 10.18653/v1/P19-1042.

[8]. Gotou T., Nagata R., Mita M., Hanawa K. Taking the correction difficulty into account in grammatical error correction evaluation. In Proceedings of the 28th International Conference on Computational Linguistics. Barcelona, Spain (Online): International Committee on Computational Linguistics, 2020, pp. 2085–2095. doi: 10.18653/v1/2020.coling-main.188.

[9]. Omelianchuk K., Atrasevych V., Chernodub A., Skurzhanskyi O. GECToR – grammatical error correction: Tag, not rewrite. In Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications. Seattle, WA, USA → Online: Association for Computational Linguistics, 2020, pp. 163–170. doi: 10.18653/v1/2020.bea-1.16.

[10]. Cargill T. The design of a spelling checker's user interface. ACM SIGOA Newsletter, 1(3), 1980, pp. 3-4.

[11]. Bentley J. Programming pearls: A spelling checker. Communications of the ACM, 28(5), 1985, pp. 456–462.

[12]. Chollampatt S., Ng H. T. Connecting the dots: Towards human-level grammatical error correction. In Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 327–333. doi: 10.18653/v1/W17-5037.

[13]. Ge T., Wei F., Zhou M. Fluency boost learning and inference for neural grammatical error correction. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 1055–1065. doi: 10.18653/v1/P18-1097.

[14]. Sakaguchi K., Post M., Van Durme B. Grammatical error correction with neural reinforcement learning. In Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Taipei, Taiwan: Asian Federation of Natural Language Processing, 2017, pp. 366–372.

[15]. Katsumata S., Komachi M. Stronger Baselines for Grammatical Error Correction Using a Pretrained Encoder-Decoder Model. In Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, Suzhou, China: Association for Computational Linguistics, 2020, pp. 827–832.

[16]. Rothe S., Mallinson J., Malmi E., Krause S., Severyn A. A Simple Recipe for Multilingual Grammatical Error Correction. In 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, 2021, pp. 702–707. doi: 10.18653/v1/2021.acl-short.89.

[17]. Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M. A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E., Lample, G. (2023) Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (online). Available at: https://arxiv.org/abs/2302.13971v1, accessed 18.12.2023.

[18]. Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., Hajishirzi, H. (2022) Self-instruct: Aligning language model with self-generated instructions. *arXiv preprint arXiv:2212.10560* (online). Available at: https://arxiv.org/abs/2212.10560, accessed 18.12.2023.

[19]. Taori R., Gulrajani I., Zhang T., Dubois Y., Li X., Guestrin C., Liang P., Hashimoto T. B. Alpaca: A Strong, Replicable Instruction-Following Model. The Center for Research on Foundation Models of Stanford Institute for Human-Centered Artificial Intelligence. Available at: https://crfm.stanford.edu/2023/03/13/alpaca.html, accessed 18.12.2023.

[20]. Floridi L., Chiriatti M. Gpt-3: Its nature, scope, limits, and consequences. Minds and Machines, 30, 2020, pp. 1–14.

[21]. Coyne S., Sakaguchi K., Galvan-Sosa D., Zock M., Inui K. Analyzing the Performance of GPT-3.5 and GPT-4 in Grammatical Error Correction. arXiv e-prints, p. arXiv:2303.14342 (online). Available at: https://arxiv.org/abs/2303.14342, accessed 18.12.2023.

[22]. Östling R., Gillholm K., Kurfalı M., Mattson M., and Wirén M. (2023) Evaluation of really good grammatical error correction. arXiv e-prints, p. arXiv:2308.08982 (online). Available at: https://arxiv.org/abs/2308.08982v1, accessed 18.12.2023. doi: 10.18653/v1/2022.emnlp-main.162.

[23]. Zhang Yu., Zhang B., Li Zh., Bao Z., Li Ch., Zhang M. SynGEC: Syntax-Enhanced Grammatical Error Correction with a Tailored GEC-Oriented Parser. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, pp. 2518–2531.

[24]. Zhou, H., Liu, Y., Li, Z., Zhang, M., Zhang, B., Li, C., Zhang J., Huang, F. (2023) Improving Seq2Seq Grammatical Error Correction via Decoding Interventions. *arXiv preprint arXiv:2310.14534* (online). Available at: https://arxiv.org/abs/2310.14534, accessed 18.12.2023.

[25]. Ng H. T., Wu S. M., Briscoe T., Hadiwinoto C., Susanto R. H., Bryant C. The CoNLL-2014 shared task on grammatical error correction. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task. Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 1–14. doi: 10.3115/v1/W14-1701.

[26]. Bryant C., Ng H. T. How far are we from fully automatic high quality grammatical error correction? In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, 2015, pp. 697–707. doi: 10.3115/v1/P15-1068.

[27]. Grundkiewicz R., Junczys-Dowmunt M., Gillian E. Human evaluation of grammatical error correction systems. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 461–470. doi: 10.18653/v1/D15-1052.

[28]. Napoles C., Sakaguchi K., Post M., Tetreault J. Ground truth for grammatical error correction metrics. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 588–593. doi: 10.3115/v1/P15-2097.

[29]. Chollampatt S., Ng H. T. A reassessment of reference-based grammatical error correction metrics. In Proceedings of the 27th International Conference on Computational Linguistics. Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018, pp. 2730–2741.

[30]. Mizumoto T., Hayashibe Y., Komachi M., Nagata M., Matsumoto Y. The effect of learner corpus size in grammatical error correction of ESL writings. In Proceedings of COLING 2012: Posters, Kay M. and Boitet C., Eds. Mumbai, India: The COLING 2012 Organizing Committee, 2012, pp. 863–872.

[31]. Tajiri T., Komachi M., Matsumoto Y. Tense and aspect error correction for ESL learners using global context. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Li H., Lin C.-Y., Osborne M., Lee G. G., and Park J. C., Eds. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 198–202.

[32]. Yannakoudakis H., Briscoe T., Medlock B. A new dataset and method for automatically grading ESOL texts. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Lin D., Matsumoto Y., Mihalcea R., Eds. Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 180–189.

[33]. Coltheart M., Rastle K., Perry C., Langdon R., Ziegler J., Drc: a dual route cascaded model of visual word recognition and reading aloud. Psychological review, 108(1), 2001, pp. 204–256. doi: 10.1037/0033-295X.108.1.204.

[34]. Castles A., Rastle K., Nation K., Ending the reading wars: Reading acquisition from novice to expert. Psychological Science in the Public Interest, 19(1), pp. 5–51, 2018, pMID: 29890888. doi: 10.1177/1529100618772271

[35]. Lichtarge J., Alberti C., Kumar S., Shazeer N., Parmar N., Tong S., "Corpora generation for grammatical error correction," in 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), Burstein J., Doran C., Solorio T., Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3291–3301. [Online]. Available: https://aclanthology.org/N19-1333

[36]. Näther M. An in-depth comparison of 14 spelling correction tools on a common benchmark. In Proceedings of the Twelfth Language Resources and Evaluation Conference, Calzolari N., Béchet F., Blache P., Choukri K., Cieri C., Declerck T., Goggi S., Isahara H., Maegaard B., Mariani J., Mazo H., Moreno A., Odijk J., Piperidis S., Eds. Marseille, France: European Language Resources Association, 2020, pp. 1849–1857.

[37]. Napoles C., Sakaguchi K., Tetreault J., JFLEG: A fluency corpus and benchmark for grammatical error correction. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Lapata M., Blunsom P., Koller A., Eds. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 229–234.

[38]. Qiu Z., Qu Y. A two-stage model for chinese grammatical error correction, IEEE Access, 7, pp. 146 772–146 777, 2019.

[39]. Hinson C., Huang H.-H., Chen H.-H. Heterogeneous recycle generation for Chinese grammatical error correction. In Proceedings of the 28th International Conference on Computational Linguistics, Scott D., Bel N., Zong C., Eds. Barcelona, Spain (Online): International Committee on Computational Linguistics. 2020, pp. 2191–2201. doi: 10.18653/v1/2020.coling-main.199.

Информация об авторах / Information about authors

Владимир Миронович СТАРЧЕНКО — аспирант Школы лингвистики Факультета гуманитарных наук НИУ ВШЭ, стажёр-исследователь Научно-учебной лаборатории учебных корпусов НИУ ВШЭ. Сфера научных интересов: автоматическое исправление грамматических ошибок, корпусная лингвистика, распределённые вычисления.

Vladimir Mironovich STARCHENKO is a graduate student at the School of Linguistics of the Faculty of Humanities of the HSE University, a research intern at the Laboratory of Learner's Corpora of the Higher School of Economics. Research interests: grammatical error correction, corpus linguistics, distributed systems.

Алексей Миронович СТАРЧЕНКО — аспирант и преподаватель Школы лингвистики Факультета гуманитарных наук НИУ ВШЭ, стажёр-исследователь Научно-учебной лаборатории по формальным моделям в лингвистике НИУ ВШЭ. Сфера научных интересов: аргументная структура, номинализации, фокусные частицы, полевая лингвистика, корпусная лингвистика.

Aleksey Mironovich STARCHENKO is a graduate student and lecturer at the School of Linguistics of the Faculty of Humanities of of the HSE University, a research intern at the Laboratory on Formal Models in Linguistics of the Higher School of Economics. Research interests: argument structure, nominalizations, focus particles, field linguistics, corpus linguistics.

DOI: 10.15514/ISPRAS-2023-35(5)-15

Применение мультимодального трансформера для прогнозирования выходных параметров насыщенных углеводородных соединений из состава тяжелой нефти в присутствии катализаторов

[1]*П.А. Пылов,* ORCID: 0000-0003-3214-6592 <*pylovpa@kuzstu.ru*>
[1]*Р.В. Майтак,* ORCID: 0009-0009-4353-6925 <*maytak.roman@mail.ru*>
[2] *Е.Г. Зайцева,* ORCID: 0000-0003-1269-329X <*zaitsevaeg@mail.ru*>

[1] *Кузбасский государственный технический университет имени Т.Ф. Горбачева, 650000, Россия, г. Кемерово, ул. Весенняя, д. 28.*
[2] *Казанский национальный исследовательский технологический университет, 420015, Россия, г. Казань, ул. Карла Маркса, д. 68.*

Аннотация. Предложена интеллектуальная модель на базе мультимодального трансформера для решения задачи прогнозирования времени и площади выхода различных углеводородных компонентов из состава тяжелой нефти при использовании катализаторов на основе шести металлов: никеля, меди, марганца, свинца, цинка и натрия. В качестве входных данных интеллектуальная модель принимает две модальности: хроматограмму образца чистой сырой нефти, представленную в виде графической информации и сопровождающие её табличные данные. На выходе мультимодальный трансформер позволяет получить прогнозные табличные данные, которые формализуют перераспределенный групповой состав нефти и описывают как наименования полученных углеводородов, так и две их качественные характеристики: время выхода спектров компонентов и их относительную площадь. Моделирование прогноза превращений высокомолекулярных соединений в легкомолекулярные на основе разработанной модели позволяет существенно сократить временные, аппаратные и человеческие ресурсы, необходимые для выбора нужного типа катализатора в нефтехимических лабораториях. В процессе исследования было обнаружено, что обучение интеллектуальной модели на данных одного месторождения позволяет в дальнейшем выполнять аналогичный прогноз с приемлемой точностью для данных другого месторождения тяжелой нефти. Величина ошибки прогноза интеллектуальной модели удовлетворяет требованиям, предъявляемыми нефтехимической лабораторией для практического применения мультимодального трансформера.

Ключевые слова: тяжелая нефть; интеллектуальный анализ данных; интеллектуальная обработка хроматограмм тяжелой нефти; мультимодальный трансформер; компьютерное зрение.

Для цитирования: Пылов П.А., Майтак Р.В., Зайцева Е.Г. Применение мультимодального трансформера для прогнозирования выходных параметров насыщенных углеводородных соединений из состава тяжелой нефти в присутствии катализаторов. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 229–244. DOI: 10.15514/ISPRAS–2023–35(5)–15.

Application of a Multimodal Transformer to the Prediction of the Yield of Saturated Hydrocarbon Compounds from Heavy Crude Oil in the Presence Of Catalysts

[1] *P.A. Pylov* ORCID: 0000-0003-3214-6592 <*pylovpa@kuzstu.ru*>
[1] *R.V. Maitak* ORCID: 0009-0009-4353-6925 <*maytak.roman@mail.ru*>
[2] *E.G. Zaitseva* ORCID: 0000-0003-1269-329X <*zaitsevaeg@mail.ru*>

[1] *T.F. Gorbachev Kuzbass State Technical University,*
28, Vesennya st., Kemerovo, 650000, Russia.
[2] *Kazan National Research Technological University,*
68, Karl Marx st., Kazan, 420015, Russia.

Abstract. Heavy oil fields are a promising energy source in the future due to the depletion of natural sources of light oil. However, extraction, transportation and refining of heavy oil is significantly more complicated than light oil - difficulties arise at almost all technological stages. One of such stages is laboratory analytics of heavy oil and selection of the most optimal catalyst for extraction of required fractions from crude oil sample. Different catalysts are actively used in petrochemical laboratories, but special attention is paid to those of them, the basis of which is metal. In this study, catalysts based on six different metals namely zinc, nickel, copper, manganese, lead and sodium were analyzed. In order to analyze the yield ratios of the required components from the crude heavy oil composition, it is necessary to test different types of catalysts sequentially on a base sample. The yield of different hydrocarbons on a small volume of oil can be reliably estimated by chromatographic study, which takes about 68 minutes for both the base oil sample and the different catalysts. Since testing 6 different catalysts would require almost 7 hours of chromatographic analysis, a rational solution would be to apply data mining techniques to this task. A multimodal transformer model was proposed to solve this problem. It takes as input two modalities: a chromatogram of a sample of pure crude oil presented as graphical data and accompanying tabular data, which are also generated by the chromatograph and consist of text and numbers. At the output, the model produces predictive tabular data that formalize the redistributed group composition of the oil and describe both the names of the newly produced hydrocarbons and their two qualitative characteristics: time and yield area. Obtaining the prediction makes it possible to significantly reduce the time, hardware and human resources required to select the right type of catalyst in petrochemical laboratories. In the process of the study, it was found that training of the intellectual model on the data of one field allows to perform further similar forecast with acceptable accuracy for the data of another heavy oil field. The magnitude of the prediction error of the intelligent model satisfies the requirements set by the petrochemical laboratory for practical application of the multimodal transformer.

Keywords: heavy oil; intellectual analysis of data; intelligent processing of heavy oil chromatograms; multimodal transformer; computer vision.

For citation: Pylov P.A., Maitak R.V., Zaitseva E.G. Application of a multimodal transformer to the prediction of the yield of saturated hydrocarbon compounds from heavy crude oil in the presence of catalysts. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 229-244 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-15.

1. Введение

Нефть является одним из наиболее популярных и востребованных ресурсов в мировой топливно-энергетической сфере [1]. Как правило, природная смесь сложных углеводородов не используется промышленностью в своём первозданном виде, так как для получения технически ценного материала – например, моторного топлива, необходима её переработка. Стоит заметить, что процесс переработки различается для легких, средних и тяжелых нефтей – такая классификация принята на основании различающейся плотности нефти, добываемой из разных месторождений. Поскольку в легкой нефти содержание бензиновых и масляных фракций преобладает над нефтью других типов, она считается наиболее ценной для топливной промышленности. Однако, статистика по добычи нефти показывает, что

фокусирование большей части мировой нефтедобычи (70%) на легкой нефти значительно истощает её запасы в фонде природных ресурсов [2], что заставляет нефтяные компании переходить к разработкам более сложных пластов тяжелой высоковязкой нефти, требующих иного технологического подхода к процессу нефтедобычи. Кроме усложнения процесса добычи нефти, усложняется и этап её переработки, поскольку теперь для извлечения легких нефтяных фракций необходимо применять методы каталитического облагораживания и акватермолиза.

Принимая во внимание планомерное истощение разведанных месторождений легкой нефти, а также факт сосредоточения основных мировых запасов углеводородов в составе тяжелой нефти [3], следует отметить, что Российская Федерация занимает третье место в мире по запасам этого труднодобываемого сырья [4]. Совокупность перечисленных факторов подчеркивает актуальность технологической оптимизации процесса переработки тяжелой нефти. Выполнить оптимизацию переработки можно разными способами, например:

1. Совершенствованием оборудования;
2. Видоизменением химической технологии;
3. Автоматизацией одного из этапов трудоемкого процесса переработки тяжелой нефти.

Третий способ включает в себя различные методы химического анализа, в том числе групповой анализ на основе хроматографии. Она позволяет разделить тяжелую нефть на группы химически однородных соединений, поэтому является основным методом для исследования компонентного состава фракций насыщенных и ароматических углеводородов, входящих в состав тяжелой нефти. Несмотря на то, что хроматография выполняется с помощью специальных регистрирующих приборов – хроматографов, которые позволяют получить хроматограммы (здесь и далее в контексте статьи под семантикой термина «хроматограмма» понимается графическое представление непрерывных изменений уровня сигнала хроматографа, зависящих от времени выхода различных компонентов тяжелой нефти) и таблицы селективно определенных органических компонентов, анализ введенного образца тяжелой нефти на этом этапе не заканчивается, а, наоборот, только начинается – хроматограф выступает в роли инструмента, позволяя определить состав вязкой сложной смеси углеводородов, а процесс непосредственной интерпретации результатов остается за исследователем.

Прогнозирование результатов хроматографии тяжелой нефти является творческой задачей, так как она не может быть решена с помощью прямых инструкций методами классического программирования. Наиболее распространенным инструментом для автоматизации подобных задач выступает интеллектуальный анализ данных, на котором основаны алгоритмы машинного обучения. Имплементирование методов прикладного искусственного интеллекта в предметную область переработки тяжелой нефти позволит решить следующие подзадачи:

- Получить на выходе интеллектуальной модели предсказанные табличные данные, на основе которых экспертами формируется прогнозная хроматограмма нефти после катализа;
- Выявить закономерности и особенности в изменениях хроматограмм образца сырой тяжелой нефти при введении в неё катализаторов различного типа и химической природы.

Поскольку автоматизация задачи прогнозирования результатов хроматограмм тяжелой нефти подразумевает генерацию новых табличных данных на основе как графической, так и табличной информации, требуется разработать такую архитектуру интеллектуальной модели, которая будет объединять методы компьютерного зрения и обработки символьных данных (числовых и текстовых) в рамках одной общей структуры.

2. Обзор существующих интеллектуальных методов автоматизации процессов в нефтегазовой области

Выполнение обзора по уже разработанным интеллектуальным моделям, автоматизирующим какой-либо технологический процесс в нефтегазовой области, позволило сделать следующие выводы:

1. В научной среде активно исследуется интеграция методов искусственного интеллекта в задачу оптимизации добычи нефти и газа:
   - В работе [5] оценивается эффективность достоверного прогнозирования критически важных параметров пластовых флюидов (в том числе давление насыщения нефти газом, плотность нефти и газа в общей смеси) на основе искусственных нейронных сетей. В рамках работы авторы сообщают, что практическое применение интеллектуального решения позволяет оптимизировать процесс планирования, разведки и разработки нефтяных месторождений;
   - Научная статья [6] подчеркивает необходимость использования моделей глубокого обучения для сокращения длительности технологических стадий разработки новых месторождений нефти и газа. Относительно классической технологии, автором публикации доказывается не только факт существенного сокращения общих временных затрат, требуемых на разработку месторождения, но и сопутствующий эффект снижения проектных инвестиционных рисков за счет устранения субъективной компоненты, носящей природу человеческого фактора;
   - Автоматизация процесса разведки нефтяных месторождений на основе программной реализации модели случайного леса подробно рассмотрена в работе [7], авторы которой решают проблему ускоренной обработки данных геологоразведки, позволяя тем самым снизить финансовые затраты на исследование новых источников нефти более чем на 10%.
2. Многие исследователи данных смещают фокус своего внимания на экономическую составляющую нефтяной промышленности, используя в качестве инструментария алгоритмы прикладного машинного обучения:
   - В работе [8] был проведен анализ большой выборки финансовых показателей крупной российской нефтегазовой компании на основе модели персептрона. Основной целью исследования авторы обозначили формирование прогноза прибыли компании от продажи нефти на следующий календарный год. На основе разработанной гипотезы авторы получили прогнозное значение чистой прибыли компании, однако, в рамках исследования [8] не производили сравнения предсказанного результата с фактическим значением.
3. Наблюдается низкая активность применения интеллектуальных методов в задаче анализа нефти как таковой. Например, в работе [9] исследователи анализировали процесс сбора проб нефти автоматическими пробоотборниками из трубопроводов. Авторы сообщают, что автоматический сбор является непредставительным, поэтому имплементация технологий искусственного интеллекта может существенно снизить величину ошибки анализа проб.

Другие немногочисленные (чаще всего единичные) публикации посвящены решению конкретных узкоспециализированных задач в рамках объемного (с точки зрения составных этапов) процесса анализа нефти и нефтепродуктов, который до сих пор большей своей частью остаётся неавтоматизированным.

4. Вероятнее всего, широкое применение методов искусственного интеллекта для автоматизации задач разведки и разработки нефтяных месторождений, а также активная программная реализация интеллектуальных прогностических моделей для

предсказания экономической эффективности добычи жидкого углеводородного сырья напрямую связано с повышением общей коммерциализации нефтедобывающего комплекса.

Из обзора следует, что в нефтегазовой отрасли большая часть наукоемких исследований, связанных с применением интеллектуальных методов, сосредоточена в области добычи полезных ископаемых, а не их непосредственной аналитики. Однако, цифровая трансформация предметной области нефтепереработки возможна только при условии автоматизации каждого составного процесса, в том числе и процессов, направленных на исследование нефти: ярким примером является задача прогнозирования табличных данных хроматограмм образцов тяжелого высоковязкого углеводородного сырья при введение в него катализаторов.

В статье приводятся результаты решения данной задачи на основе интеллектуальной модели, позволяющей прогнозировать перераспределения разных компонентов нефти при вводе в базовый образец того или иного катализатора. В качестве основы обучающей выборки выступают хроматограммы сырой тяжелой нефти и соответствующие им табличные данные хроматографа.

3. Постановка задачи предметной области

Подготовка выборок (обучающей и тестовой) для разработки интеллектуальной модели велась на основе данных научно-исследовательской лаборатории «Внутрипластовое горение» Казанского (Приволжского) федерального университета. Поставщиком образцов является эксплуатируемое Ашальчинское месторождение тяжелой высоковязкой нефти, территориально находящееся в Альметьевском районе Республики Татарстан.

Пригодные для анализа моделью данные были получены на основе проведения хроматографического исследования «чистых» образцов сырой тяжелой нефти, а также путем выполнения хроматографии тех же самых образцов после их первоначального взаимодействия с различными металлическими катализаторами в автоклаве. В качестве основы для создания разных катализаторов участвовали шесть металлов: цинк, никель, медь, свинец, марганец и натрий. Таким образом, по отличающимся образцам сырой тяжелой нефти было суммарно получено семь хроматограмм (включая исследование базового образца). Время, требуемое для получения одной хроматограммы, не зависело от типа применяемого катализатора, поэтому было одинаковым для всех исследований и составляло 68 минут.

Отметим, что одно хроматографическое исследование тяжелой нефти позволяет получить сразу два типа информации: графическую, представленную в виде хроматограммы (рис. 1) и символьную, формализованную в виде табличных данных (рис. 2).

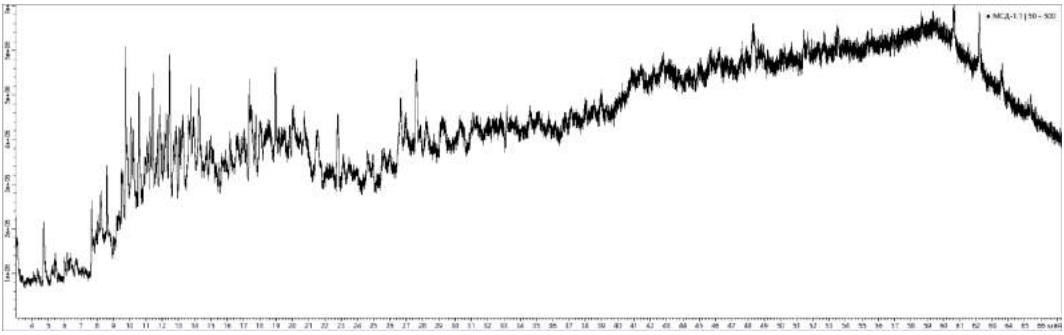


Рис. 1. Хроматограмма образца сырой тяжелой нефти.
Fig. 1. Chromatogram of crude heavy oil sample.

TIC, Ашальчинское CO2 при 200 oC			
Компонент	**Время (мин)**	**Площадь (мВ*с)**	**Площадь (%)**
бензотиофен	9,758	17267704,48	5,018665
4,9-Dimethylnaphtho[2,3-b]thiophe	21,546	16368772,31	4,757401
триметилдибензотиофен	25,602	13334638,39	3,875563
тетраметилдибензотиофен	29,238	12154576,01	3,532592
Benzene, 1,1'-ethylidenebis[4-ethyl-	27,622	11906630,7	3,460529
алкилбензол C16H26	12,448	11783965,5	3,424878
4-метилдибензотиофен	17,472	11682157,2	3,395288
триметилдибензотиофен	26,648	11368077,66	3,304005
алкилбензол C20H34	22,793	11198608,26	3,25475
бензотиофен	9,507	10308088,48	2,995931
бензотиофен	11,854	9803665,463	2,849326
2-и 3-метилдибензотиофен	17,988	8735080,607	2,538753
бензотиофен	11,434	8706273,44	2,530381
алкилбензол C15H24	10,564	8660651,325	2,517121
алкилбензол C19H32	18,97	8436103,279	2,451859
тетраметилнафталин	12,875	8209904,899	2,386117
1-метилдибензотиофен	18,522	7597563,33	2,208147
триметилфенантрен	28,196	7095569,118	2,062248
бензотиофен	12,739	7000756,881	2,034692
алкилбензол C18H30	17,346	6881246,649	1,999957
триметилдибензотиофен	24,609	6343488,301	1,843664
диметилфенантрен	23,147	5965456,548	1,733793
бензотиофен	10,094	5839176,837	1,697091
бензотиофен	11,28	5280047,245	1,534587
алкилбензол	8,246	5168975,952	1,502305
Phenanthrene, 9,10-dihydro-1-methy	16,655	5087925,716	1,478749
алкил-дигидроантрацен	26,97	5013743,813	1,457189
1,1'-Biphenyl, 3,4-diethyl-	17,758	4899112,432	1,423872
Алкилбензол C10	4,734	4842567,827	1,407438
триметилнафталин	11,101	4813859,904	1,399095
4,9-Dimethylnaphtho[2,3-b]thiophe	20,005	4740126,948	1,377665
триметилнафталин	10,245	4587091,694	1,333187
бензотиофен	13,283	4491292,826	1,305344
алкилбензол C13H20	7,662	4431553,469	1,287981
4,9-Dimethylnaphtho[2,3-b]thiophe	20,732	4428490,33	1,287091
бензотиофен	13,072	4368290,006	1,269595
алкилбензол C14H22	8,607	4219503,238	1,226351

Рис. 2. Выдержка из табличных данных органических соединений в составе образца, отсортированных по убыванию площади на хроматограмме.
Fig. 2. Excerpt from tabulated data of organic compounds in the sample, sorted by decreasing area on the chromatogram.

На основе полученной информации (рис. 1, 2), исследователь нефтехимической лаборатории получает первичный анализ компонентов, входящих в состав тяжелой нефти. В зависимости от поставленной технологической задачи, ключевую роль играют отдельные компоненты в составе сложной смеси углеводородов. Для того, чтобы повысить процент выхода приоритетного компонента из образца тяжелой нефти, используются катализаторы на основе металлов.

Однако, не существует достоверной эвристики, по которой применение конкретных катализаторов позволит заранее предопределить получение процентных соотношений требуемой фракции, поэтому специалист нефтехимической лаборатории должен самостоятельно анализировать перераспределение разных компонентов при вводе того или иного катализатора в автоклав. Получить достоверные результаты при малых объемах нефтяного сырья представляется возможным только при использовании

хроматографического исследования, поэтому лаборатории вынуждены проводить данную процедуру анализа нефти многократно (по числу катализаторов).

Таким образом, основной задачей исследования является оценка эффективности применения инструментов интеллектуального анализа данных для решения задачи прогнозирования выходных параметров (времени и площади выхода) углеводородных компонентов из состава тяжелой нефти при введении в неё катализаторов на основе шести металлов: меди, никеля, цинка, свинца, марганца и натрия.

4. Описание метода решения задачи

Стоит заметить, что эффективность интеллектуальной модели существенно зависит от того, каким образом будут организованы обучающие данные. Несмотря на то, что хроматограмма (рис. 1) и её табличное сопровождение (рис. 2) отражают результаты одного хроматографического анализа, они не могут полностью заменить друг друга по двум основным причинам:

1. Хроматограмма непрерывно регистрирует изменение величины аналитического сигнала в течении всего временного интервала исследования. В табличных данных регистрируются лишь пики, которые характеризуют детектирование уникальных углеводородных компонентов в составе тяжелой нефти хроматографом, а задать сколь угодно малый интервал регистраций для табличных данных не представляется возможным.
2. Следствие из первого пункта: в табличных данных фиксируется лишь площадь и время выхода компонента. Однако, без сопровождающей хроматограммы, представить фиксированную площадь на ограниченном временном интервале на плоскости можно неединичным способом, при этом разные вариации представления кривой, имеющей одну и ту же площадь, будут представлять разные образцы тяжелой нефти. Таким образом, восстановить исходную хроматограмму по табличным данным невозможно.

В силу отсутствия возможности установления малого интервала времени для регистрации табличных данных (ограничение большинства хроматографов), возникает проблема точного аналитического задания временного ряда. Решение этой проблемы может быть найдено через аппроксимацию данных с хроматограммы (выполняемой вручную) или через применение интеллектуальных моделей, функционирующих на основе алгоритмов компьютерного зрения. Авторами данной статьи был избран второй вариант, так как он предполагает наиболее полную автоматизацию поставленной задачи предметной области.

В рамках исследования анализировались хроматограммы, получаемые с хроматографа и имеющие разрешение 600 dpi. Для анализа устойчивости интеллектуальной модели разрешение входных тестовых хроматограмм (которые ранее не передавались для исследования моделью) было предварительно занижено до 300 dpi в программном продукте Adobe Photoshop 2022 (версии 23.2.1), что не повлияло на результирующую точность прогноза – загрузка изображений оригинального качества позволило получить аналогичные результаты точности. Использование более низкого разрешения хроматограммы не исследовалось в рамках работы.

В качестве архитектуры интеллектуальной модели была избрана модель мультимодального трансформера, которая позволяет совместно анализировать как графическую (рис. 1), так и символьную информацию, представленную в рассматриваемой задаче в виде таблиц (рис. 2). Подобные архитектуры хорошо зарекомендовали себя в практическом применении при анализе разнородных данных [10]. В рамках автоматизируемой задачи для каждой модальности использовался свой собственный энкодер, который позволял проходить данным через последовательность слоёв самовнимания и определять закономерности для

графических и символьных данных: за аналитику графических данных отвечал трансформер компьютерного зрения (Vision Transformer, ViT) [11], обобщение закономерностей символьных табличных данных выполняла облегченная архитектура трансформера, организованная по примеру TabNet [12]. В обоих составных частях использовалась функция активация ReLU.

Метрикой для оценки точности разработанного решения была определена величина среднеквадратической ошибки между прогнозными значениями площади и времени выхода компонентов на хроматограмме с использованием катализаторов и их фактическими значениями.

Согласно постановке задачи, используется несколько «копий» базового образца сырой нефти, с целью дальнейшего введения в каждую «копию» различающегося катализатора. Для того, чтобы обучить модель машинного обучения, необходимо предоставлять ей связанные пары «хроматограмма – табличные данные», полученные как для базового образца, так и для катализатора. Поскольку в загружаемых на один и тот же вход парах «базовый образец – образец после применения катализатора» хроматограммы сырой нефти будут всегда повторяться, то возникнет процедура ввода избыточной информации («базовый образец» будет повторяться ровно шесть раз – по числу катализаторов) в процесс обучения интеллектуальной модели. Для нивелирования этого негативного эффекта было принято решение создать шесть отдельных ветвей в интеллектуальной модели, каждая из которых сформирует обобщающую способность для конкретного («своего») катализатора. Под термином «ветвь» имеется в виду независимая модель трансформера, гиперпараметры которого в процессе обучения оптимальным образом настраиваются под свой катализатор (рис. 3).

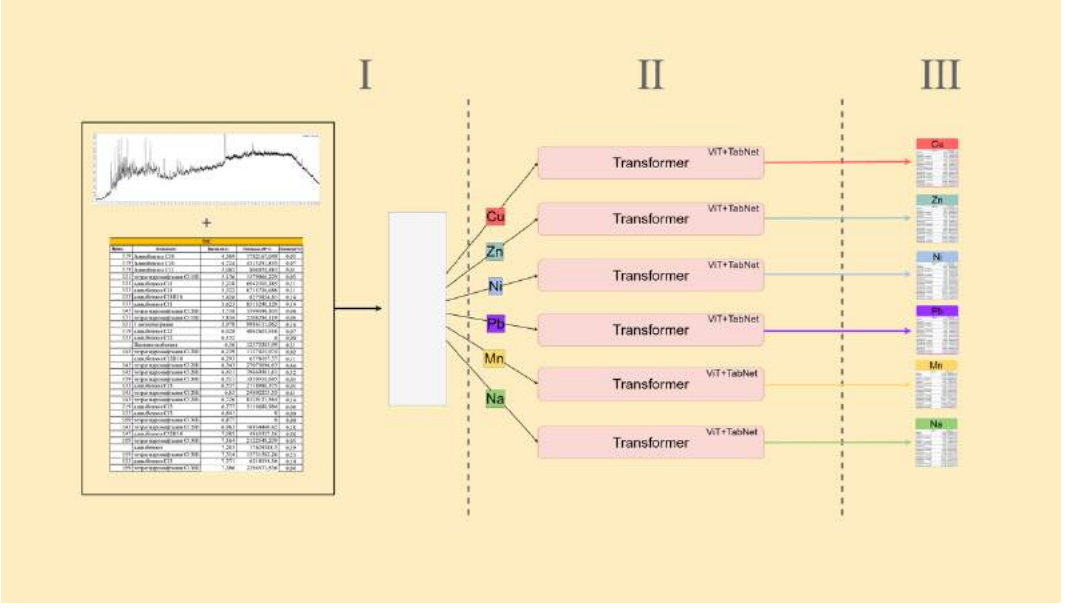


Рис. 3. Общая структура разработанной интеллектуальной модели.
Fig. 3. General structure of the intellectual model developed.

Подробнее рассмотрим итоговую архитектуру разработанного программного решения (рис. 3):

- В I блоке в модель мультимодального трансформера поступает набор данных образца сырой тяжелой нефти (хроматограмма и соответствующие ей табличные

данные). Эти данные передаются без изменений на каждую «ветвь», представляющую собой отдельный трансформер.

- Во II блоке каждая «ветвь» позволяет решать задачу прогнозирования выхода отдельных компонентов из состава базового образца тяжелой нефти при введении в неё катализаторов разного типа.
- В III блоке выходным прогнозным данным присваивается метка соответствующего катализатора. Этот пункт необходим для того, чтобы однозначно идентифицировать прогнозные данные для эксперта предметной области по каждому типу катализатора.

5. Практические результаты применения модели мультимодального трансформера

Практическое применение реализованной модели мультимодального трансформера позволило получить прогнозные значения для состава углеводородов образцов тяжелой нефти при вводе в неё различных катализаторов. В качестве источников данных выступали образцы тяжелой нефти, полученные не только с Ашальчинского, но и с Мордово-Кармальского месторождения. Важной особенностью формирования выборок являлся факт включения образцов нефти Мордово-Кармальского месторождения исключительно в состав тестовых экземпляров, что позволило оценить уровень обобщающий способности разработанной интеллектуальной модели не только на новых данных «известного» месторождения, но и на экспериментально полученных данных другого источника нефти, которые не участвовали в процессе обучения и не могли оказывать влияние на формирование обобщающей способности модели. Такой подход позволяет оценить уровень универсальности разработанного автоматизирующего решения для дальнейшего его переноса на другие месторождения тяжелой нефти.

Для обучения модели были собраны и проанализированы 2100 различных образцов тяжелой нефти. Данные обучающей и тестовой выборки были репрезентативно разделены в соотношении 80:20, после чего было произведено обучение интеллектуальной модели. Результаты экспериментов показали, что в модели трансформера компьютерного зрения оптимальное количество слоев должно составлять 12 (число одинаково для каждой из шести «ветвей»). При установлении этого значения, результаты точности интеллектуальной модели достигают своих наибольших значений. Поскольку каждая «ветвь» мультимодального трансформера (рис. 3) отвечала за свой тип катализатора, то для достижения наиболее точных прогнозных значений, в модели внимания необходимо было настроить оптимальное число «голов» (от английского «number of heads») [11], которое позволяет варьировать число преобразований, через которые проходят входные матрицы данных. Результаты настройки позволили определить оптимальное число «голов» для каждого трансформера (единственный варьируемый разработчиком гиперпараметр):

- 8 для трансформеров, прогнозирующих ввод катализатора на основе меди, никеля и марганца;
- 10 для трансформера, прогнозирующем данные по катализатору на основе цинка;
- 6 для трансформера, прогнозирующем данные по катализатору на основе свинца;
- 12 для модели трансформера, прогнозирующем данные по катализатору на основе натрия.

Инкрементирование числа «голов» сверх выверенных оптимальных значений (в любом из трансформеров) вызывало увеличение времени, требуемого для прогнозирования, которое при этом не сопровождалось увеличением точности. Обоснование такого эффекта авторы находят в повышении вычислительной сложности модели. Декрементирование оптимального числа «голов» позволяло сократить отрезок времени, требуемый для получения прогноза (с 8,2 минут до 5,7 минут), однако, это существенно увеличивало

величину максимальной ошибки – вплоть до 9,12%, что не удовлетворяло требованиям к интеллектуальному решению, предъявляемым предметной областью нефтехимического анализа.

Использование приема параллельной работы трансформеров (рис. 3), каждый из которых отвечает за свой собственный катализатор, позволило получить прогнозные результаты по всем катализаторам за 8,2 минуты (более, чем в 8 раз быстрее, чем при выполнении классического неавтоматизированного анализа).

Время, требуемое для обучения модели мультимодального трансформера (рис. 3), заняло в среднем 205 минут для каждой ветви. В качестве устройства, на котором производилось обучение, использовался ноутбук Apple MacBook Pro 2021 года с процессором Apple M1 Max, оперативной памятью объемом 32 Гб, видеокартой Apple graphics 24-core, твердотельным накопителем объемом 2 Тб и версией macOS Sonoma 14.1.1.

Сразу отметим, что критерий площади выхода прогнозировался интеллектуальной моделью только в рамках относительной площади. Можно заметить, что на рис. 2 представлена как абсолютная величина площади, так и относительная, однако, для получения прогноза в нефтехимическом исследовании достаточно только параметра относительной площади.

В таблице 1 представлены результаты прогнозных значений площади и времени выхода различных углеводородов при вводе в базовый образец тяжелой нефти катализатора из никеля. Наряду с прогнозными значениями, в соответствующих столбцах также зафиксированы фактически полученные величины выхода различных углеводородных фракций на основе тестовых табличных данных соответствующих хроматограмм.

На основании данных таблицы 1 можно рассчитать значения среднеквадратической ошибки для времени и площади выхода компонентов: они составили 0,059 и 0,055 соответственно. При этом величина ошибки при прогнозировании времени выхода элемента в процентном отношении не отклонялась более, чем на 1,704% для любого из углеводородов тяжелой нефти. Ошибка прогноза по параметру относительной площади выхода не превысила 3,007%.

В таблице 2 представлены усредненные результаты времени и площади выхода углеводородов на основании метрики среднеквадратической ошибки и максимального процентного отклонения прогнозных значений от фактических для всех оставшихся катализаторов (медь, цинк, марганец, натрий и свинец).

Результаты расчета среднеквадратической ошибки свидетельствуют о формировании высокой обобщающей способности интеллектуальной модели. Максимальное отклонение прогнозного значения от фактического при определении времени выхода в процентном выражении составило 3,007% на катализаторе из никеля; при определении относительной площади выхода наибольший процент отклонения (2,704%) был зафиксирован на одном из компонентов нефти при использовании катализатора из марганца. Несмотря на то, что некоторые образцы тяжелой нефти были получены из другого месторождения, разработанная модель позволила решить задачу прогнозирования с высокой точностью.

Однако, для понимания полноты решения задачи на данных Мордово-Кармальского месторождения было проведено дополнительное исследование, в рамках которого рассчитывалась среднеквадратическая ошибка и процент отклонения для площади и времени выхода углеводородов исключительно на данных этого месторождения (табл. 3). Ожидалось некоторое снижение точности в силу особенностей природного сырья на месторождении, отличном от того, на котором производилось обучение интеллектуальной модели.

Из таблицы 3 следует, что среднеквадратическая ошибка прогноза модели на данных нового месторождения увеличилась незначительно – менее, чем на 0,002. Модель мультимодального трансформера, реализованная и обученная на данных Ашальчинского месторождения, при тестировании на образцах данных тяжелой нефти Мордово-Кармальского месторождения, позволяет получить практически ту же самую точность.

Табл. 1. Результаты прогнозных и фактических значений времени и площади выхода для компонентов тяжелой нефти (тестовая выборка), получаемых после применения катализатора на основе никеля
Table 1. Results of predicted and actual values of time and yield area for heavy oil components (test sample) obtained after application of nickel-based catalyst

Компонент	Прогнозное время выхода	Фактическое время выхода	Модуль разницы между прогн. и факт. значением	Прогнозная площадь выхода	Фактическая площадь выхода	Модуль разницы между прогн. и факт. значением
Hexadecane, 2,6,10,14-tetramethyl-	16,210	16,178	0,032	17,010	16,801	0,209
Pentadecane, 2,6,10-trimethyl-	12,194	12,176	0,018	10,900	11,046	0,146
Pentadecane, 2,6,10,14-tetramethyl-	13,498	13,545	0,047	9,670	9,830	0,160
Eicosane, 3-methyl-	25,194	25,218	0,024	3,320	3,249	0,071
Octadecane, 3-methyl-	18,404	18,393	0,011	3,070	3,099	0,029
Hexadecane, 2,6,11,15-tetramethyl-	14,420	14,67	0,250	2,980	3,024	0,044
2,6,10-Trimethyltridecane	8,527	8,518	0,009	3,010	2,959	0,051
Tetradecane, 2,6,10-trimethyl-	10,418	10,406	0,012	2,590	2,657	0,067
Tetradecane, 3-methyl-	8,852	8,844	0,008	2,370	2,423	0,053
Hexacosane	42,101	42,069	0,032	2,400	2,337	0,063
Heptacosane	44,323	44,319	0,004	1,945	1,956	0,011
Docosane	30,284	30,281	0,003	1,952	1,930	0,022
Tetracosane	36,815	36,763	0,052	1,927	1,932	0,005
Triacontane	50,749	50,701	0,048	1,828	1,826	0,002
Octacosane	46,598	46,501	0,097	1,820	1,798	0,022
Heneicosane, 3-methyl-	27,574	27,558	0,016	1,770	1,746	0,024
Tricosane	33,685	33,671	0,014	1,724	1,737	0,013
Octadecane	15,998	15,967	0,031	1,723	1,715	0,008
Heptadecane	13,449	13,427	0,022	1,707	1,694	0,013
Nonacosane	48,725	48,626	0,099	1,725	1,679	0,046
Heneicosane	26,523	26,415	0,108	1,648	1,659	0,011
Eicosane	21,970	21,915	0,055	1,635	1,635	0,000
Tetradecane	7,598	7,619	0,021	1,540	1,583	0,043
Hexadecane, 7,9-dimethyl-	11,025	10,922	0,103	1,584	1,580	0,004
Pentacosane	39,714	39,615	0,099	1,537	1,530	0,007
Dodecane, 2,6,10-trimethyl-	7,252	7,246	0,006	1,530	1,517	0,013
Heptadecane, 2,6,10,14-tetramethyl-	18,714	18,672	0,042	1,498	1,491	0,007
Hexadecane, 7-methyl-	10,184	10,173	0,011	1,435	1,438	0,003
Hentriacontane	52,705	52,696	0,009	1,421	1,407	0,014
Hexadecane	11,162	11,151	0,011	1,323	1,354	0,031
Octadecane, 2-methyl-	16,842	16,834	0,008	1,210	1,193	0,017
Nonadecane	18,981	18,97	0,011	1,167	1,173	0,006
Dotriacontane	54,683	54,635	0,048	1,143	1,156	0,013
Nonadecane, 2-methyl-	21,059	21,044	0,015	1,054	1,063	0,009
Pentadecane	9,201	9,195	0,006	1,035	1,011	0,024
Nonadecane, 3-methyl-	21,552	21,549	0,003	1,025	1,004	0,021
Tritriacontane	56,552	56,544	0,008	0,910	0,886	0,024
Tetratriacontane	58,394	58,386	0,008	0,642	0,652	0,010
Pentadecane, 3-methyl-	10,538	10,535	0,003	0,571	0,578	0,007
Dodecane, 2,5-dimethyl-	6,520	6,49	0,030	0,499	0,504	0,005
Tridecane, 7-methyl-	5,893	5,963	0,070	0,152	0,148	0,004

Табл. 2. Данные о величине среднеквадратической ошибки и максимального отклонения (в процентах) прогноза времени и площади выхода углеводородов на тестовой выборке при использовании различных катализаторов
Table 2. Data on the value of the RMS error and the maximum deviation (in per cent) of the prediction of the time and area of hydrocarbon yield on a test sample using different catalysts

Катализатор	Величина ошибки	Площадь выхода	Время выхода
Медь	Максимальный процент отклонения	1,812	1,585
	Среднеквадратическая ошибка	0,042	0,053
Цинк	Максимальный процент отклонения	1,783	1,890
	Среднеквадратическая ошибка	0,039	0,048
Марганец	Максимальный процент отклонения	2,704	2,237
	Среднеквадратическая ошибка	0,057	0,073
Свинец	Максимальный процент отклонения	1,916	2,315
	Среднеквадратическая ошибка	0,061	0,087
Натрий	Максимальный процент отклонения	2,147	2,454
	Среднеквадратическая ошибка	0,054	0,094

Табл. 3. Данные о величине среднеквадратической ошибки и максимального отклонения (в процентах) прогноза времени и площади выхода углеводородов на выборке, образцы которой получены с Мордово-Кармальского месторождения
Table 3. Data on the value of the mean square error and the maximum deviation (in per cent) of the prediction of the time and area of hydrocarbon yield on the sample obtained from the Mordovo-Karmalskoye field

Катализатор	Величина ошибки	Площадь выхода	Время выхода
Медь	Максимальный процент отклонения	1,840	1,625
	Среднеквадратическая ошибка	0,043	0,055
Никель	Максимальный процент отклонения	3,112	1,758
	Среднеквадратическая ошибка	0,057	0,061
Цинк	Максимальный процент отклонения	1,831	1,852
	Среднеквадратическая ошибка	0,040	0,047
Марганец	Максимальный процент отклонения	2,783	2,287
	Среднеквадратическая ошибка	0,059	0,075
Свинец	Максимальный процент отклонения	1,974	2,359
	Среднеквадратическая ошибка	0,063	0,089
Натрий	Максимальный процент отклонения	2,224	2,504
	Среднеквадратическая ошибка	0,056	0,096

При этом разброс результатов точности (по катализаторам всех типов) при множественном запуске разработанной интеллектуальной модели на 275 тестовых данных Мордово-Кармальского (нового) месторождения в среднем составил 1,825% отклонения прогнозных значений от фактических для площади выхода и 1,734% для величины времени выхода.

На основании этого факта можно сделать предположение о том, что извлекаемые мультимодальным трансформером признаки, которыми обладают образцы тяжелой сырой нефти Ашальчинского месторождения, в большей степени являются универсальными, чем индивидуальными. Открытие такой характеристики разработанной интеллектуальной модели позволяет использовать мультимодальный трансформер для автоматизированного анализа тяжелой нефти на других месторождениях без дополнительного обучения новыми данными. Важно отметить, что по мнению сотрудников нефтехимической лаборатории, точность прогноза модели мультимодального трансформера полностью удовлетворяет прогнозным требованиям предметной области нефтепереработки – в подобных исследованиях величина ошибки не должна превышать 5%.

6. Заключение

Исследовано применение модели мультимодального трансформера для решения задачи прогнозирования перераспределений углеводородных фракций в составе тяжелой нефти при введении в неё катализаторов на основе шести металлов: цинка, меди, никеля, натрия, марганца и свинца.

Результаты практических испытаний показали, что автоматизация поставленной задачи была выполнена интеллектуальной моделью с величиной ошибки, не превышающей 3,007% на объединенных тестовых данных Ашальчинского и Мордово-Кармальского месторождения тяжелой нефти, при этом данные второго месторождения не были включены в состав обучающей выборки. Оценка точности интеллектуальной модели исключительно на данных Мордово-Кармальского месторождения продемонстрировала незначительное снижение точности, вызванное повышением максимальной ошибки на отдельном катализаторе до 3,112%. При этом время, требуемое для получения прогноза по всем типам катализаторов (вне зависимости от типа месторождения) составило в среднем 8,2 минуты.

Приведенные результаты позволяют считать, что точность прогноза времени и площади выхода различных компонентов из состава тяжелой нефти при вводе в неё отдельных катализаторов, является достаточной для применения разработанной модели мультимодального трансформера при решении задачи предварительной оценки выхода отдельных углеводородов и выбора необходимого типа катализатора.

Список литературы / References

[1]. Шадрина А. В., Крец В. Г. Основы нефтегазового дела. М., Нац. Открытый Ун-т «ИНТУИТ», 2016. 214 с. / Shadrina, A.V., Krets, V.G. Fundamentals of Oil and Gas Engineering. Moscow, National Open University of INTUIT, 2016, 214 p. (in Russian).

[2]. Басарыгин Ю. М., Будников В. Ф., Булатов А. И. Исследование факторов и реализация мер долговременной эксплуатации нефтяных и газовых скважин. М., Просвещение-Юг, 2004, 242 с. ISBN 5-93491-054-X. / Basarygin Y. M., Budnikov V. F., Bulatov A. I. Research of factors and implementation of measures for long-term operation of oil and gas wells. Moscow, Prosveshchenie-Yug, 2004, 242 p. (in Russian). ISBN 5-93491-054-X.

[3]. Шевченко Д. В., Васильева Л. Х. Математическое моделирование вытеснения тяжелых нефтей горячей водой в тонком пласте. Казань, Издательство "Познание", 2013, 60 с. / Shevchenko D. V., Vasilieva L. H. Mathematical modelling of displacement of heavy oils by hot water in a thin reservoir. Kazan, Poznanie Publishing House, 2013, 60 p. (in Russian).

[4]. Хуснутдинов И. Ш., Копылов А. Ю., Гончарова И. Н. Разработка и совершенствование сольвентных технологий переработки тяжелого органического сырья. Казань, КГТУ, 2009, 265 с. ISBN 978-5-7882-0745-2 / Khusnutdinov I. Sh., Kopylov A. Yu., Goncharova I. N. Development and

improvement of solvent technologies for processing of heavy organic raw materials. Kazan, KSTU, 2009, 265 p. (in Russian). ISBN 978-5-7882-0745-2.

[5]. Хадавимогаддам Ф., Мищенко И. Т., Мостаджеран М. Применение методов искусственного интеллекта в прогнозировании основных свойств нефти. Газовая промышленность, вып. 12(794), 2019, стр. 28-32. / Hadavimoghaddam F., Mishchenko I. T., Mostajeran M. Application of artificial intelligence methods in predicting basic oil properties. Gas Industry, vol. 12(794), 2019, pp. 28-32 (in Russian).

[6]. Азиева Р. Х. Искусственный интеллект в добыче нефти и газа: возможности и сценарный прогноз. Проблемы экономики и управления нефтегазовым комплексом, вып. 3(207), 2022, стр. 38-46. DOI: 10.33285/1999-6942-2022-3(207)-38-46. / Azieva R. H. Artificial intelligence in oil and gas production: opportunities and scenario forecast. Problems of economics and management of oil and gas complex, vol. 3(207), 2022, pp. 38-46 (in Russian). DOI: 10.33285/1999-6942-2022-3(207)-38-46.

[7]. Байбаров Д. А. Оценка продуктивности и экономической эффективности технологий искусственного интеллекта для автоматизации процессов разведки и добычи нефти и газа. XXI век: итоги прошлого и проблемы настоящего плюс, том 10, вып. 3, 2021, стр. 100-105. DOI: 10.46548/21vek-2021-1055-0019 / Baybarov D. A. Assessment of productivity and economic efficiency of artificial intelligence technologies for automation of oil and gas exploration and production processes. XXI century: Resumes of the Past and Challenges of the Present plus, vol. 10, issue 3, 2021, pp. 100-105 (in Russian). DOI: 10.46548/21vek-2021-1055-0019

[8]. Ломакин Н. И., Дженифер О. Ч., Голодова О. А., Сычева А. В., Кабина В. В. AI-система "Персептрон" для прогноза финансового результата деятельности предприятия нефтяной отрасли РФ. Фундаментальные исследования, вып. 12-1, 2019, стр. 98-103. DOI: 10.17513/fr.42629 / Lomakin N. I., Jennifer O. C., Golodova O. A., Sycheva A. V., Kabina V. V. AI-system "Perseptron" for forecasting the financial result of the enterprise of the oil industry of the Russian Federation. Fundamental Research, vol. 12-1, 2019, pp. 98-103 (in Russian). DOI: 10.17513/fr.42629

[9]. Овсеенко Г. А., Козелков О. В., Кашаев Р. С. Использование нейронных сетей в мехатронном устройстве представительного отбора и анализа проб. Приборостроение и автоматизированный электропривод в топливно-энергетическом комплексе и жилищно-коммунальном хозяйстве. Материалы VII Национальной научно-практической конференции, 2022, стр. 92-96. / Ovseenko G. A., Kozelkov O. V., Kashaev R. S. Use of neural networks in mechatronic device of representative sampling and analysis. Instrumentation and automated electric drive in fuel and energy complex and housing and communal services. Proceedings of VII National Scientific and Practical Conference, 2022, pp. 92-96 (in Russian).

[10]. Vaswani A., Shazeer N. et al. Attention is all you need. In Proc. of the 31st Conference on Neural Information Processing Systems (NIPS), 2017, 11 p.

[11]. Dosovitskiy A., Beyer L. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proc. of the International Conference on Learning Representations (ICLR), 2021, 21 p. DOI: 10.48550/arXiv.2010.11929

[12]. Arik S. Ö., Pfister T. TabNet: Attentive Interpretable Tabular Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 35(8), 2020, pp. 6679-6687. DOI: 10.1609/aaai.v35i8.16826

Информация об авторах / Information about authors

Петр Андреевич ПЫЛОВ – аспирант Кузбасского государственного технического университета имени Т.Ф. Горбачева. Совмещает учебу с работой старшим разработчиком высоконагруженных интеллектуальных систем на позиции Senior Computer Vision Engineer. Научные интересы: компьютерное зрение, обработка естественного языка, глубокое обучение, разработка интеллектуальных систем для автоматизации различных прикладных задач.

Petr Andreevich PYLOV – post-grad. student at the T.F. Gorbachev Kuzbass State Technical University. He combines his studies with his work as a Senior Computer Vision Engineer. Research interests: computer vision, natural language processing, deep learning, development of intelligent systems for automation of various applied tasks.

Роман Вячеславович МАЙТАК – магистрант Кузбасского государственного технического университета имени Т.Ф. Горбачева. Учебу совмещает с работой на позиции Middle+ NLP Data Scientist. Научные интересы: обработка естественного языка, глубокое обучение, обработка текстовых и числовых данных моделями машинного обучения, автоматизация технологических задач.

Roman Viacheslavovich MAITAK – Master student at the T.F. Gorbachev Kuzbass State Technical University. He combines his studies with her work as a data scientist at Middle+ NLP. Research interests: natural language processing, deep learning, processing of textual and numerical data by machine learning models, automation of technological tasks.

Елизавета Георгиевна ЗАЙЦЕВА – аспирант кафедры Химической технологии нефти и газа Казанского национального исследовательского университета. Область научной деятельности связанна с разработкой каталитических систем для процессов превращения тяжелого нефтяного сырья. Научные интересы: нефтехимия, каталитическое облагораживание тяжелой нефти, акватермолиз.

Elizaveta Georgievna ZAITSEVA – post-grad. student at the Department of Chemical Technology of Oil and Gas, Kazan National Research University. The field of scientific activity is related to the development of catalytic systems for the conversion of heavy oil feedstock. Research interests: oil chemistry, catalytic refining of heavy oil, aquathermolysis.

DOI: 10.15514/ISPRAS-2023-35(5)-16

CC BY

Применение физически-обоснованной нейронной сети на примере моделирования гидродинамических процессов, допускающих аналитическое решение

[1] *К.Б. Кошелев,* ORCID: 0000-0002-7124-3945 <*k.koshelev@ispras.ru*>
[2] *С.В. Стрижак,* ORCID: 0000-0001-5525-5180 <*s.strijhak@ispras.ru*>

[1] *Институт водных и экологических проблем СО РАН,*
656038, Алтайский край, г. Барнаул, ул. Молодежная, д.1.
[2] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Рассматривается актуальный подход для разработки физически-обоснованной нейронной сети для решения модельных задач для течения Коважного, геофизического течения Бельтрами, течения на участке реки по теории мелкой воды. Физически-обоснованные нейронные сети (PINN) позволяют существенно сокращать время расчета по сравнению с обычными вычислениями. Для каждого модельного течения существует свое аналитическое решение. Обсуждается архитектура программной библиотеки DeepXDE, ее состав по модулям, приводятся фрагменты программного кода на языке программирования Python. Модель PINN протестирована на тестовой выборке. Оценка предсказания выполнена с помощью метрики MSE. Полносвязанная нейронной сеть может содержать в себе 4, 7,10 скрытых слоев с количеством нейронов 50, 50, 100 соответственно. Обсуждается влияние гиперпараметров нейронной сети на величину ошибки предсказания. Расчеты, выполненные на сервере с графической картой Nvidia GeForce RTX 3070, позволяют существенно сократить время обучения для PINN.

Ключевые слова: нейронная сеть; слои; нейроны; течение Коважного; течение Бельтрами; теория мелкой воды; сетка; канал; аналитическое решение; область; точки; обучение; ошибка.

Для цитирования: Кошелев К.Б., Стрижак С.В. Применение физически-обоснованной нейронной сети на примере моделирования гидродинамических процессов, допускающих аналитическое решение. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 245–258. DOI: 10.15514/ISPRAS–2023–35(5)–16

Application of Physics-Informed Neural Network on the Example of Modeling Hydrodynamic Processes That Allow an Analytical Solution

[1] *K.B. Koshelev*, ORCID: 0000-0002-7124-3945 <*k.koshelev@ispras.ru*>
[2] *S.V. Strijhak,* ORCID: 0000-0001-5525-5180 <*s.strijhak@ispras.ru*>

[1] *Institute for water and environmental problems SB RAS,*
1, Molodezhnaya str., Altai region, Barnaul, 656038.
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. We consider an actual approach to develop a physically based neural network for solving model problems for the Kovazhny flow, for the geophysical Beltrami flow, and for the flow in a section of the river by the shallow water theory. Physics-informed neural networks (PINN) allow to significantly reduce the

computation time compared to conventional computations. There is a different analytical solution for each model flow. The architecture of the DeepXDE software library, its composition by modules, and fragments of program code in the Python programming language are discussed. The PINN model is tested on a test sample. The prediction is evaluated using the MSE metric. The fully connected neural network can contain 4, 7, 10 hidden layers with the number of neurons 50, 50, 100 respectively. The influence of hyperparameters of the neural network on the magnitude of the prediction error is discussed. The calculations performed on a server with Nvidia GeForce RTX 3070 card can significantly reduce the training time for PINN.

Keywords: neural network; layers; neurons; Kovasznay flow; Beltrami flow; Swallow Water Equations; grid; canal; analytical solution; domain; points; training; error.

For citation: Koshelev K.B., Strijhak S.V. Application of physics-informed neural network on the example of modeling hydrodynamic processes that allow an analytical solution. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 245-258 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-16.

1. Введение

В настоящее время существует большое количество различных вычислительных методов для решения задач гидродинамики, разработанных на протяжении последних пятидесяти лет, которые работают достаточно эффективно, если известны основные уравнения и все масштабы турбулентности разрешены.

За последние пять лет, было предпринято ряд попыток создать модели на базе нейронных сетей (NN) для решения задач несжимаемой жидкости в форме уравнений Навье-Стокса. Самый распространенный подход для изучения турбулентных течений заключался в построении моделей замыкания турбулентности с управляемыми данными.

Другое перспективное направление связано с развитием физически-обоснованных нейронных сетей, так называемых Physics-Informed Neural Network - PINN, для решения систем уравнений Эйлера и Навье-Стока.

Нейронные сети являются универсальными аппроксиматорами непрерывных функций. Универсальная теорема аппроксимации утверждает, что нейронные сети могут быть использованы для аппроксимации любой непрерывной функции с произвольной точностью, если не накладывать ограничений на ширину и глубину скрытых слоев. Теорема, доказанная математиком Джорджем Цыбенко в 1989 году, утверждает, что полносвязанная нейронная сеть с одним скрытым слоем может точно аппроксимировать любой нелинейный непрерывный оператор. Эта универсальная теорема аппроксимации наводит на мысль о потенциальном применении нейронных сетей для обучения нелинейных операторов на основе доступных данных. Однако теорема гарантирует лишь малую ошибку аппроксимации для достаточно большой сети, и не учитывает важные ошибки оптимизации и обобщения.

В настоящее время имеется необходимость в разработке, обосновании и практической реализации новых архитектур нейронных сетей для аппроксимации непрерывных функций.

Авторы из Brown University, США пошли путем, используя свойство универсальной аппроксимации нейронных сетей, которое вместе с автоматическим дифференцированием функции в рамках фреймворка для машинного обучения (TensorFlow, PyTorch) позволяет разрабатывать «решатели» для систем уравнений Эйлера, Навье-Стокса, которые не требуют генерации расчетной сетки.

В частности, в работах [1-2] авторы впервые ввели концепцию физически-обоснованных нейронных сетей (PINN) для решения прямых и обратных задач, включающих несколько различных типов уравнений в частных производных и обыкновенных дифференциальных уравнений, с использованием бессеточного метода.

2. Математическая модель

Уравнения, описывающие двумерные стационарные несжимаемые ламинарные течения, имеют следующий вид:

$$\nabla \cdot u = 0$$

$$\frac{\partial u}{\partial t} + \nabla(u^T u) - \frac{1}{Re}\Delta u = -\nabla p/\rho$$

Здесь u – векторная функция, представляющая скорость жидкости, p – скалярная функция давления жидкости, ρ- плотность, Re– число Рейнольдса.

Из курса гидродинамики известно, что существует ряд модельных краевых задач, для которых существуют аналитические решения. Это течение Коважного, течение Куэтта, вихрь Тейлора-Грина, геофизическое течение Бельтрами, ячейка Хеле-Шоу, всплытие пузырька в вязкой жидкости, перемешивание жидкости между двумя эксцентрично вращающимися круговыми цилиндрами [3-4].

3. Модель физически-обоснованной нейронной сети

В рамках данной работы рассматривается архитектура полносвязанной нейронной сети для построения физически-обоснованной нейронной сети для моделирования различных простейших течений. Для нейронной сети вводятся понятия: входной слой с нейронами с заданными на входе признаками в форме координат точек и дискретных значений времени, несколько скрытых слоев с нейронами, выходной слой с нейронами. Также рассматривается задание начальных и граничных условий, расчетной области с заданным облаком точек, и функции потерь, которая связана с уравнениями неразрывности, движения, с заданными граничными и начальными условиями. Дополнительно для нейронной сети выбирается функция активации нейронов и метод оптимизации для функции потерь.

Подобная физически-обоснованная нейронная сеть состоит из трех основных блоков (рис.1). Первая часть включает в себя модуль для вычисления остаточных слагаемых для дифференциальных уравнений в частных производных или относительную погрешность решения в норме L2, а также погрешности для начальных и граничных условий. Параметры для полносвязанной нейронной сети определяются путем нахождения минимума для общей функции потерь. Входы для нейронной сети преобразуются в соответствующие выходы. Вторая часть это полносвязанная нейронная сеть с физическими данными, которая берет выходные поля скоростей и вычисляет их производные, используя исходные уравнения для движения и неразрывности при решении задач механики жидкости. Также оцениваются граничные и начальные условия, данные наблюдений из эксперимента. Последним шагом является механизм формирования обратной связи, который минимизирует функцию потерь, используя заданный оптимизатор (Adam, L-BFGS-B), в соответствии с некоторой скоростью обучения, чтобы получить оптимальные параметры для нейронной сети [5].

Для вычисления невязок при решении системы уравнений Навье-Стокса для скорости и давления, частные дифференциальные операторы вычисляются с помощью процедуры автоматического дифференцировании (AutoDiff - AD), которое может быть непосредственно сформулировано в нейронных сетях глубокого обучения, например, используя оператор "tf.gradients()" в библиотеке TensorFlow [6]. Пример архитектуры физически-обоснованной нейронной сети для системы уравнений Навье-Стокса приведен на рис. 2.

4. Библиотека DeepXDE

Для разработки тематической PINN можно использовать открытую библиотеку DeepXDE. В библиотеку DeepXDE входят различные модули (рис. 3).

Для программной реализации нейронной сети и решения задачи течения Коважного, Бельтрами, в прямоугольном канале была использована открытая библиотека DeepXDE, разработанная на языке программирования Python, с использованием библиотек для машинного обучения TensorFlow, PyTorch [7-8]. Библиотека DeepXDE поддерживает следующий функционал:

- 5 типов граничных условий (1-го рода, 2-го рода, 3-го рода, другие);
- построение простейшей геометрии.

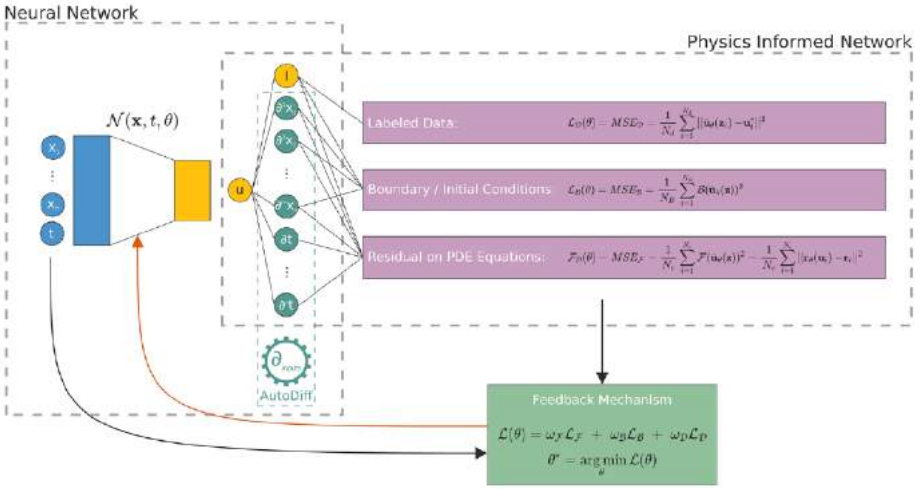


Рис.1 Общая схема для построения физически-обоснованной нейронной сети [5].
Fig. 1 The typical scheme for PINN architecture [5].

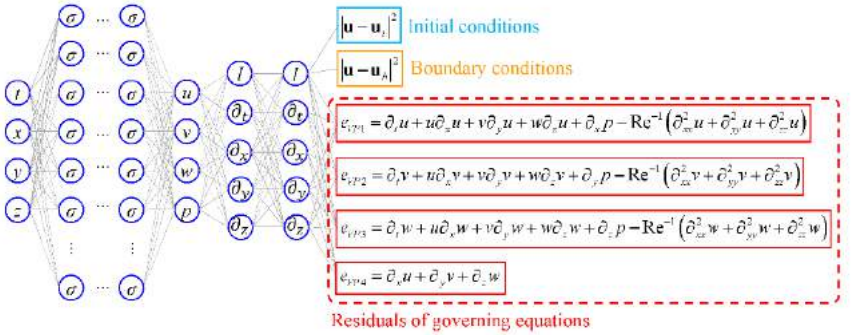


Рис. 2. Архитектура нейронной сети для системы уравнений Навье-Стокса [12].
Fig. 2. The architecture of neural network for Navier-Stokes system equations [12].

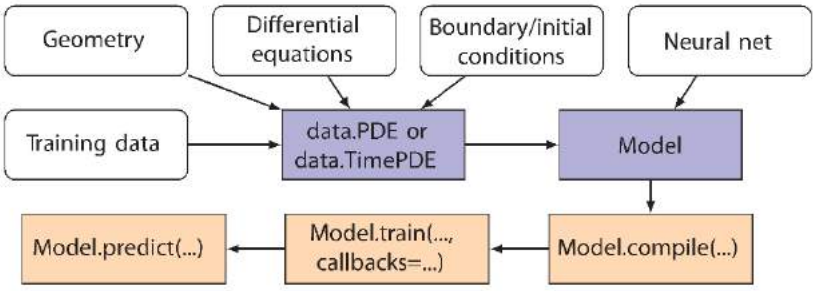


Рис.3 Состав библиотеки DeepXDE [7].
Fig.3 Structure of DeepXDE library [7].

Для этого имеется реализация работы с типовыми объектами: 1D: random_points, random_boundary_points, periodic_point, background_points, background_points_left, background_points_right; 2D: "Disk", "Polygon", "Rectangle", "Triangle"; 3D: Cuboid (Hypercube), Sphere (Hypersphere). Имеется поддержка булевых операций.

В DeepXDE реализованы различные методы для решателей NN: Fractional PDE решатель, IDE решатель, ODE решатель, time-independent PDE решатель. Имеется набор тестовых примеров (pinn_forward; pinn_inverse, другие). Функция потерь определена через задание величины ошибки MAE. Имеется поддержка работы с различными фреймворками для машинного обучения TensorFlow, PyTorch, Paddle, JAX.
Библиотека DeepXDE может быть настроена под Linux ОС для работы с графической картой Nvidia GPU с использованием дополнительных библиотек на языке CUDA.

5. Модельные течения, допускающие аналитические решения

5.1 Двумерное течение Коважного

Двумерное течение Коважного является стационарным течением за решеткой (сеткой) и имеет точное решение вида для функции скорости u(x,y), v(x,y), давления p(x) [9].

$$u(x,y) = 1 - e^{(-\lambda x)}cos(2\pi y)$$
$$v(x,y) = -\frac{\lambda}{2\pi}e^{(-\lambda x)}sin(2\pi y)$$
$$p(x) = -\frac{1}{2}e^{-2\lambda x}$$

Здесь параметр λ определяется соотношением:

$$\lambda = \sqrt{\frac{Re^2}{4} + 4\pi^2} - \frac{Re}{2}$$

Данная задача является хорошим тестом для проверки устойчивости численного решения и точного решения, чтобы продемонстрировать пространственную экспоненциальную скорость сходимости выбранного алгоритма [9-11].

5.1.1 Программная реализация

Приведем фрагмент программного кода на языке программирования Python.
Задание исходных величин:

```
Re = 20
nu = 1 / Re
l = 1 / (2 * nu) - np.sqrt(1 / (4 * nu ** 2) + 4 * np.pi ** 2)
```

Функция на Python для расчета уравнений для количества движения, неразрывности [8]:

```
def pde(x, u):
    u_vel, v_vel, p = u[:, 0:1], u[:, 1:2], u[:, 2:]
    u_vel_x = dde.grad.jacobian(u, x, i=0, j=0)
    u_vel_y = dde.grad.jacobian(u, x, i=0, j=1)
    u_vel_xx = dde.grad.hessian(u, x, component=0, i=0, j=0)
    u_vel_yy = dde.grad.hessian(u, x, component=0, i=1, j=1)
    v_vel_x = dde.grad.jacobian(u, x, i=1, j=0)
    v_vel_y = dde.grad.jacobian(u, x, i=1, j=1)
    v_vel_xx = dde.grad.hessian(u, x, component=1, i=0, j=0)
    v_vel_yy = dde.grad.hessian(u, x, component=1, i=1, j=1)
    p_x = dde.grad.jacobian(u, x, i=2, j=0)
    p_y = dde.grad.jacobian(u, x, i=2, j=1)
    momentum_x = (
        u_vel * u_vel_x + v_vel * u_vel_y + p_x - 1 / Re * (u_vel_xx + u_vel_yy)
    )
    momentum_y = (
        u_vel * v_vel_x + v_vel * v_vel_y + p_y - 1 / Re * (v_vel_xx + v_vel_yy)
    )
    continuity = u_vel_x + v_vel_y
```

```
    return [momentum_x, momentum_y, continuity]
```

Функция для задания аналитического решения для u (x,y):

```
def u_func(x):
    return 1 - np.exp(l * x[:, 0:1]) * np.cos(2 * np.pi * x[:, 1:2])
```

Функция для задания аналитического решения для v(x,y):

```
def v_func(x):
    return l / (2 * np.pi) * np.exp(l * x[:, 0:1]) * np.sin(2 * np.pi * x[:, 1:2])
```

Функция для задания аналитического решения для p(x):

```
def p_func(x):
    return 1 / 2 * (1 - np.exp(2 * l * x[:, 0:1]))
```

Функция для задания граничного условия для выходного условия:

```
def boundary_outflow(x, on_boundary):
    return on_boundary and np.isclose(x[0], 1)
```

Задание расчетной области и граничных условий:

```
spatial_domain = dde.geometry.Rectangle(xmin=[-0.5, -0.5], xmax=[1, 1.5])
boundary_condition_u = dde.icbc.DirichletBC(
    spatial_domain, u_func, lambda _, on_boundary: on_boundary, component=0
)
boundary_condition_v = dde.icbc.DirichletBC(
    spatial_domain, v_func, lambda _, on_boundary: on_boundary, component=1
)
boundary_condition_right_p = dde.icbc.DirichletBC(
    spatial_domain, p_func, boundary_outflow, component=2
)
```

Определение исходных данных для системы уравнений:

```
data = dde.data.TimePDE(
    spatial_domain,
    pde,
    [boundary_condition_u, boundary_condition_v, boundary_condition_right_p],
    num_domain=2601,
    num_boundary=400,
    num_test=100000,
)
```

Формирование модели для нейронной сети:

```
net = dde.nn.FNN([2] + 4 * [50] + [3], "tanh", "Glorot normal")
model = dde.Model(data, net)
```

В состав архитектуры нейронной сети вход один входной слой с двумя входами (координаты точек и время), 4 скрытых слоя, в каждом задано по 50 нейронов, выходной слой с 3 выходами (две компоненты скорости и давление).

5.1.2 Результаты решения задачи течения Коважного

Задача о течении Коважного, связанная с движением потока через решетку с заданными параметрами в аэродинамической трубе, решалась при Re=20 (рис. 4). Для данного течения не существует начального условия.

Рассматривалась расчетная область с размерами [-0.5; 1:0] x [-0.5; 1.5] и временной областью [0; 1]. На верхней и нижней границах задавалось условие симметрии. На левой границе задавалось условие Дирихле для значения скорости, основанное на точном решении.

На правой границе задавалось специальное граничное условие для выхода потока, которое было получено в работе [9]. Начальное значение для скорости полагалось равным 0.

На каждой границе прямоугольной области задавалась 101 точка с фиксированной пространственной координатой, т.е. Nb = 4 x 101. Для вычисления функции потерь в нейронной сети, случайным образом внутри домена выбиралась 2 601 точка.

В этой работе использовалась нейронная сеть с 4-мя скрытыми слоями и 50-ю нейронами на слой, с 7-ю скрытыми слоями и 50-ю нейронами на слой, с 10-ю скрытыми слоями и 100 нейронами на слой. Также применялся оптимизатор Адама для обеспечения наилучшего набора начальных обучаемых переменных для нейронной сети. Затем L-BFGS-B использовался для тонкой настройки нейронных сетей для достижения более высокой точности. Динамические веса обновлялись каждые 100 эпох обучения.

Анализировались градиенты функции потерь в зависимости от гиперпараметров нейронной сети. Относительные ошибки L2 могли достигать порядка 10^{-5}.

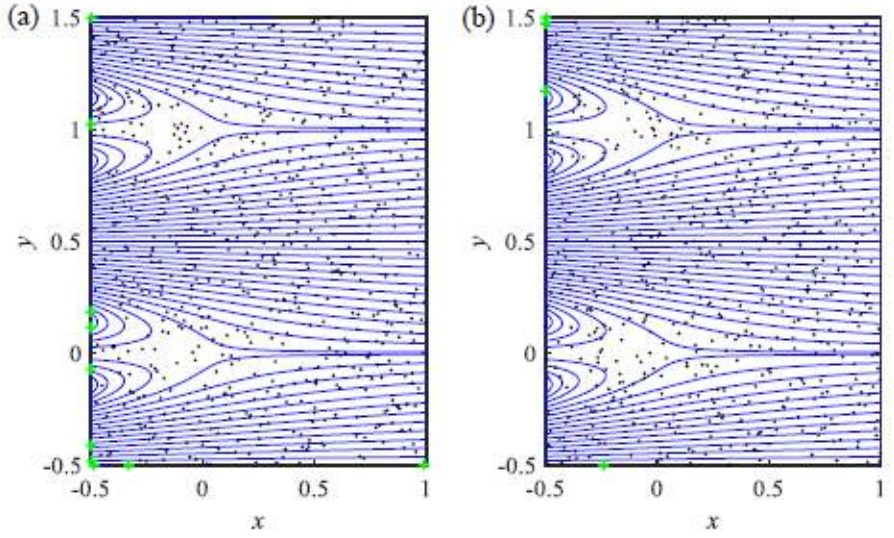


Рис. 4. Расчетная область для течения Коважного [12].
Fig. 4. Numerical domain for Kovasznay flow [12].

Обучение нейронной сети и процесс предсказания параметров течения выполнялся на ноутбуке с 32 GB RAM. Время обучения нейронной сети занимало от 30 минут до 4 часов.

Результатом выполнения программы является расчет среднеквадратичной ошибки и L2 относительной ошибки для двух компонент скорости и давления:

Mean residual: 0.0005657403
L2 relative error in u: 0.00019108094
L2 relative error in v: 0.0009619565
L2 relative error in p: 0.00042990936

5.2 Течение Бельтрами

Рассматривалась задача о нестационарном трехмерном геофизическом течении Бельтрами, для которого существует известное аналитическое решение в литературе [12].

Это специальный класс течения сплошной среды, в котором векторы скорости и завихренности коллинеараны. Для данного течения существует точное аналитическое решение, которое имеет вид:

$$u = -a[e^{ax}\sin(ay + dz) + e^{az}\sin(ax + dy)]e^{-d^2t}$$
$$v = -a[e^{ay}\sin(az + dx) + e^{ax}\sin(ay + dz)]e^{-d^2t}$$
$$w = -a[e^{az}\sin(ax + dy) + e^{ay}\sin(az + dx)]e^{-d^2t}$$

$$p = \frac{a^2}{2}\left[e^{ax} + e^{ay} + e^{az} + 2\sin(ax + dy)\cos(az + dx)e^{a(y+z)} + 2\sin(ay + dz)\cos(ax + dy)e^{a(z+x)} + 2\sin(az + dx)\cos(ay + dz)e^{a(x+y)}\right]$$

где a и d являются свободными параметрами.
Параметры уравнения: $Re = 1$, $a = 1$, $d = 1$
Для проверки работоспособности PINN использовались различные гиперпараметры (табл. 1).

Табл. 1. Гиперпараметры
Table 1. Hyperparameters

Скрытые слои	4 слоя × 100 нейронов
Первый оптимизатор	Adam, 30.000 итераций, 1e-3 скорость обучения
Второй оптимизатор	L-BFGS-B, 15.000 итераций
Точки коллокации	50000
Точки на границах	5000
Точки для начального условия	5000

5.2.1 Программная реализации

Приведем фрагмент программного кода на языке программирования Python.
Задание исходных величин:

```
a = 1
d = 1
Re = 1
```

Функция на Python для расчета уравнений неразрывности, количества движения для случая нестационарного трехмерного течения представлена ниже [8]:

```
def pde(x, u):
    u_vel, v_vel, w_vel, p = u[:, 0:1], u[:, 1:2], u[:, 2:3], u[:, 3:4]

    u_vel_x = dde.grad.jacobian(u, x, i=0, j=0)
    u_vel_y = dde.grad.jacobian(u, x, i=0, j=1)
    u_vel_z = dde.grad.jacobian(u, x, i=0, j=2)
    u_vel_t = dde.grad.jacobian(u, x, i=0, j=3)
    u_vel_xx = dde.grad.hessian(u, x, component=0, i=0, j=0)
    u_vel_yy = dde.grad.hessian(u, x, component=0, i=1, j=1)
    u_vel_zz = dde.grad.hessian(u, x, component=0, i=2, j=2)

    v_vel_x = dde.grad.jacobian(u, x, i=1, j=0)
    v_vel_y = dde.grad.jacobian(u, x, i=1, j=1)
    v_vel_z = dde.grad.jacobian(u, x, i=1, j=2)
    v_vel_t = dde.grad.jacobian(u, x, i=1, j=3)
    v_vel_xx = dde.grad.hessian(u, x, component=1, i=0, j=0)
    v_vel_yy = dde.grad.hessian(u, x, component=1, i=1, j=1)
    v_vel_zz = dde.grad.hessian(u, x, component=1, i=2, j=2)

    w_vel_x = dde.grad.jacobian(u, x, i=2, j=0)
    w_vel_y = dde.grad.jacobian(u, x, i=2, j=1)
    w_vel_z = dde.grad.jacobian(u, x, i=2, j=2)
    w_vel_t = dde.grad.jacobian(u, x, i=2, j=3)
    w_vel_xx = dde.grad.hessian(u, x, component=2, i=0, j=0)
    w_vel_yy = dde.grad.hessian(u, x, component=2, i=1, j=1)
    w_vel_zz = dde.grad.hessian(u, x, component=2, i=2, j=2)
```

```
    p_x = dde.grad.jacobian(u, x, i=3, j=0)
    p_y = dde.grad.jacobian(u, x, i=3, j=1)
    p_z = dde.grad.jacobian(u, x, i=3, j=2)

    momentum_x = (
        u_vel_t  + (u_vel * u_vel_x + v_vel * u_vel_y + w_vel * u_vel_z) + p_x
        - 1 / Re * (u_vel_xx + u_vel_yy + u_vel_zz)
    )
    momentum_y = (
        v_vel_t + (u_vel * v_vel_x + v_vel * v_vel_y + w_vel * v_vel_z) + p_y
        - 1 / Re * (v_vel_xx + v_vel_yy + v_vel_zz)
    )
    momentum_z = (
        w_vel_t  + (u_vel * w_vel_x + v_vel * w_vel_y + w_vel * w_vel_z) + p_z
        - 1 / Re * (w_vel_xx + w_vel_yy + w_vel_zz)
    )
    continuity = u_vel_x + v_vel_y + w_vel_z

    return [momentum_x, momentum_y, momentum_z, continuity]
```

Отметим, что данная программная реализация показывает, как достаточно легко записать исходную систему уравнений Навье-Стокса в коде программы на Python.

5.2.2 Результаты

Расчеты проводились для разных чисел Re=1,5,10. Полученные результаты для трех полей компонент скорости сравнивались с результатами аналитического решения (рис.5). Расчеты показали, что изменение числа Re существенно не меняет картины течения.
Процесс обучения для функции потерь от количества итераций представлен на рис. 6.

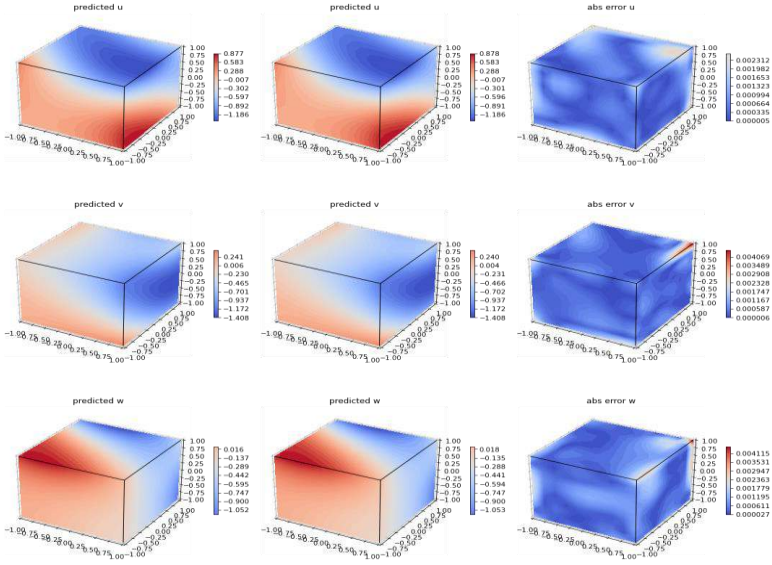


Рис.5 Поля скоростей u,v,w при t = 1.0 и значение ошибки для течения Бельтрами.
Слева на право показаны предсказанная поле скорости, аналитически-рассчитанное поле скорости, абсолютная ошибка.

Fig.5 The velocity fields u,v,w at t = 1.0 and errors for Beltrami flow.
The predicted velocity field, analytically-calculated velocity field, and absolute error are shown from left to right.

Обучение нейронных сетей и процесс предсказания параметров течения жидкости в задаче Бельтрами выполнялись на ПК с Intel Xeon, 32 GB RAM и Nvidia GPU.
Время обучения составило от 30 минут до 2 часов в зависимости от выбора гиперпараметров нейронных сетей. Динамические веса обновлялись каждые 100 эпох обучения. В качестве метрики для оценки результатов предсказания выбиралась величина MSE.
Также проводилось дополнительное исследование для случаев изменения параметра d в диапазоне от 0.75 до 1 с шагом 0.05. Данное исследование было проведено с целью отработки технологии дообучения для предсказания полей скорости и давления для промежуточных значений d и оценки времени предсказания.

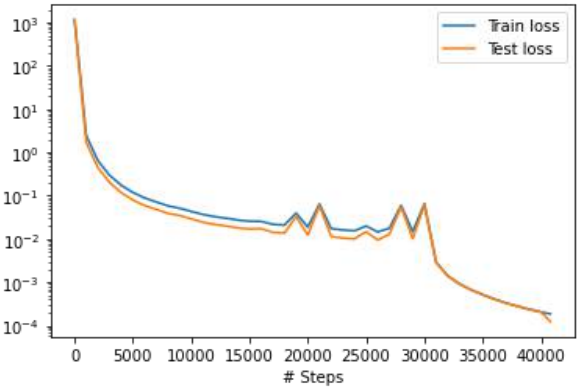


Рис. 6 Динамика изменения функции потерь при обучении нейронной сети для течения Бельтрами.
Fig.6 Dynamics of change in the loss function when training a neural network for Beltrami flow.

5.3 Течение в прямоугольном канале

Для моделирования течения на участке реки часто используются уравнения по теории мелкой воды [13, 14]. Для первоначальной оценки могут быть применены квазиодномерные уравнения Сен-Венана.
Ранее были рассмотрены архитектуры PINN для уравнений по теории мелкой воды в различной модельной постановке и программной реализации [15-17].
В данной работе рассматривалась математическая модель, включающая в себя уравнения неразрывности и движения, для безледоставного периода времени, которая имеет вид:

уравнение неразрывности

$$\frac{\partial W}{\partial t} + \frac{\partial Q}{\partial x} = q, \tag{1}$$

уравнение движения с использованием формулы Маннинга для учета трения о дно

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{W}\right) + gW\frac{\partial z}{\partial x} + \frac{gn^2 Q|Q|}{WR^{4/3}} = 0. \tag{2}$$

Здесь t – время, $x \in [0, X]$ – продольная координата вдоль русла, W – площадь живого сечения, Q – расход, q – боковая приточность на единицу длины, g – ускорение свободного падения, δ – отметка дна, n – коэффициент шероховатости по формуле Маннинга, R – гидравлический радиус, $z = \delta + H$ – уровень поверхности воды, H – глубина воды.
Полагаются известными следующие функции, определяющие морфологические параметры: $w = w(x,h)$ $h = h(x,w)$ $n = n(x)$ $\delta = \delta(x)$ $q = q(x,t)$, где w – площадь сечения, соответствующая расстоянию до отметки дна h. Начальные условия имеют вид:

$$W(x,0) = W_0(x) \quad Q(x,0) = Q_0(x). \tag{3}$$

Граничные условия для спокойного течения:

$$Q(0,t) = \hat{Q}(t) \text{ и } H(X,t) = \hat{H}(t). \quad (4)$$

Для общего случая решение поставленной задачи можно найти только численно.

Сбор необходимой для решения эмпирической информации для реального участка реки является трудоемкой задачей. Как правило, такие данные являются фрагментарными. Расстояние между имеющимися гидропостов в РФ достаточно велико. Как правило, достоверными данными на гидропостах являются только уровни поверхности воды. Данные о расходах зачастую отсутствуют или рассчитываются по суррогатным формулам. Важнейшая информация о морфологических параметрах в открытом доступе отсутствует. Значения боковой приточности могут быть определены для точечных источников, оценка же величины диффузной боковой приточности крайне приблизительна.

В этих условиях применение физически-обоснованной нейронной сети для аппроксимации набора натурных и расчетных данных позволит уточнить результаты как при решении прямой, так и обратной задач. Представляется одним из перспективных направлений получение более точной оценки распределенной (диффузной) боковой приточности.

Для оценки применимости PINN для моделирования течений в руслах и каналах представляется целесообразным начать со случаев, для которых имеются точные решения. Наиболее эффективный подход заключается в том, что для подходящего точного решения одномерных уравнений модели мелкой воды проще подобрать модельный канал, который будет полностью соответствовать выбранному точному решению. Для этой цели рассмотрим стационарное течение в прямоугольном канале шириной D с постоянным коэффициентом шероховатости n и заданными строго положительными значениями расхода $Q(x)$ и площади живого сечения $W(x)$.

Для прямоугольного канала величины глубины и гидравлического радиуса вычисляются по формулам:

$$H(x) = \frac{W(x)}{D} \text{ и } R(x) = \frac{W(x)}{D+2\cdot H(x)}. \quad (5)$$

Для выполнения уравнения неразрывности необходимо задать боковую приточность как

$$q(x) = \frac{dQ}{dx}. \quad (6)$$

Для выполнения уравнения движения надо вычислить отметку дна по формуле:

$$\delta(x) = H(0) - H(x) - \int_0^x \left(\frac{\frac{d(Q(x)^2/W(x))}{dx}}{gW(x)} + \frac{n^2 Q(x)^2}{W(x)^2 R^{4/3}} \right) dx. \quad (7)$$

Для обучения нейронной сети применяется прямая задача (1-4), в которой используются замыкающие соотношения (5), а боковая приточность и отметка дна являются предварительно вычисленными по формулам (6, 7).

5.3.1 Программная реализации

Пример реализации функции на Python для расчета уравнений неразрывности и количества движения представлен ниже.

```
def pde(x, y):
    Q, W = y[:, 0:1], y[:, 1:2]
    dQ_x = dde.grad.jacobian(y, x, i=0, j=0)
    dQ_t = dde.grad.jacobian(y, x, i=0, j=1)
    dW_t = dde.grad.jacobian(y, x, i=1, j=1)
    Q2W = Q * Q / W
    Q2W_x = dde.grad.jacobian(Q2W, x, i=0, j=0)
    H = H_func(W, x[:, 0:1])
```

```
    R = R_func(W, x[:, 0:1])
    R43 = tf.math.pow(R, fdt)
    dz_x=dde.grad.jacobian(H, x, i=0, j=0)+ddelta_x(x)
    qs = qs_func(x[:, 0:1])
    return [dQ_t+Q2W_x+g*W*dz_x+gnn*Q2W/R43, dW_t+dQ_x-qs]
```

В случае применения нейронной сети для обучения на участке реальной реки текст вышеприведенной функции не меняется, но необходимо реализовать вспомогательные функции для расчета глубины H_func, гидравлического радиуса R_func и вычисления величины $\frac{\partial \delta}{\partial x}$ функцией ddelta_x.

5.3.2 Результаты решения

В качестве тестовой задачи рассматривается задача о моделировании течения в прямоугольном канале длиной 1000 м и шириной 10 м с постоянным наклоном дна, обеспечивающим постоянство по всей длине русла расхода 1 м^3/с и площади живого сечения 10 м^2. Стационарное решение достигается путем установления нестационарной задачи за период в 1 сутки.

Количество точек коллокации в расчетной области задавалось: num_domain=25001, num_boundary=500, num_initial=500, num_test=10000. При обучении в архитектуре нейронной сети использовалось 4 скрытых слоя с 50 нейронами. Функции активации для нейронов задавалась в виде tanh. Использовался оптимизатор Адама для обеспечения наилучшего набора начальных обучаемых переменных для нейронной сети. Затем L-BFGS-B использовался для тонкой настройки нейронной сети для достижения более высокой точности.

Расчет предсказания с помощью PINN величины расхода Q представлен на рис. 7.

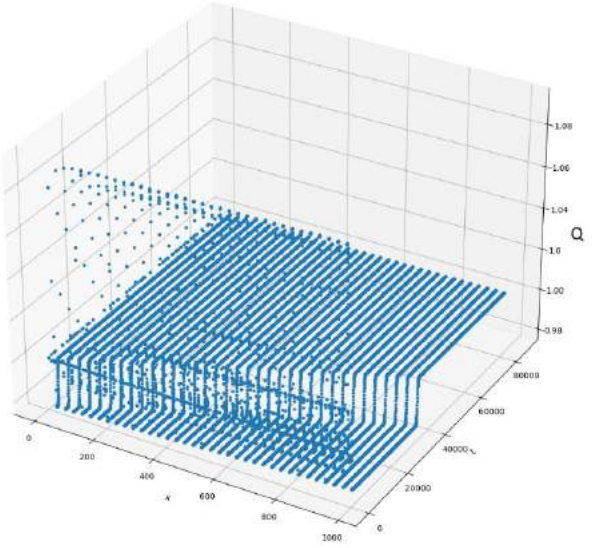


Рис. 7. Величина расхода Q.
Fig. 7. Flow rate Q.

Расчет предсказания величины площади живого сечения W представлен на рис. 8. Следует отметить, что в дальнейшем наибольший интерес представляют нестационарные решения и программная реализация создана именно для таких нестационарных процессов. Однако, пока тестирование выполнялось для стационарного решения, поэтому начальные условия задавались случайным образом. Выбросы значений расхода и площади живого сечения в начальный момент времени при обучении минимизировались и через сутки модельного

времени становились постоянными по всей длине русла канала, что свидетельствует о успехе обучения.

Обучение нейронной сети выполнялось на сервере с GPU Nvidia GeForce RTX 3070. Это позволило сократить время обучения в 15 раз по сравнению с сервером, использующим CPU с 24 потоками.

Заключение

Использование свободного программного обеспечения позволяет разрабатывать архитектуру PINN, расчетные коды для моделирования течений, допускающие аналитические решения. Для задачи с двумерным течением Коважного проведено исследование для случая Re=20.

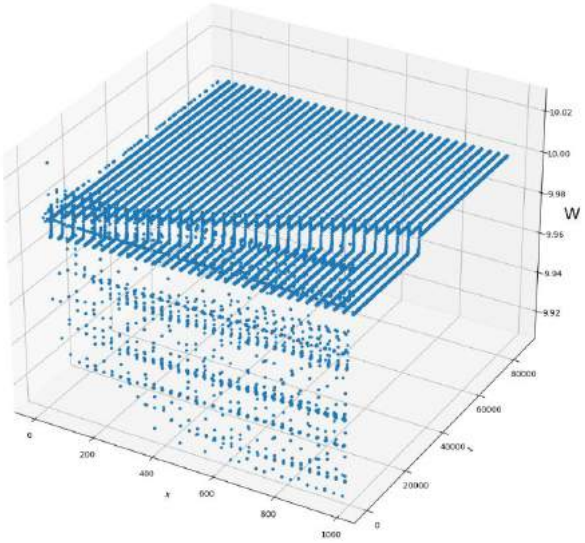


Рис. 8. Величина площади живого сечения W.
Fig. 8. Cross-sectional area value W.

Были поля скорости и давления, построены линии тока. По задаче с нестационарным трехмерным течением Бельтрами получены поля скорости и давления для заданных выбранных параметров. Выполнена оценка ошибки относительно точного аналитического решения. По задаче с течением в прямоугольном канале по теории мелкой воды в процессе обучения нейронной сети получены значения расхода и площади живого сечения, с высокой точностью совпадающие с аналитическим решением. В дальнейшем предстоит работа по применению модели для реальных участков русел рек и нестационарных течений. Дальнейшее развитие PINN для решения задач гидродинамики может быть связано с использованием различным архитектур PINN на базе глубоких операторных нейронных сетей (DeepONet) и нейронных операторов Фурье (FNO).

Список литературы / References

[1]. Raissi M., Perdikaris P., Karniadakis G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686-707.
[2]. Raissi M., Yazdani A., Karniadakis G.E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (2020) 1026-1030.
[3]. Лойцянский Л.Г. Механика жидкости и газа. Учебник для вузов. - 7-е изд., испр. - М.: Дрофа, 2003. - 840 с.
[4]. Петров А.Г. Аналитическая гидродинамика. М.: Физматлит. 2010. - 520 с.

[5]. Cuomo, Salvatore & Schiano Di Cola, Vincenzo & Giampaolo, Fabio & Rozza, Gianluigi & Raissi, Maziar & Piccialli, Francesco. (2022). Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. Journal of Scientific Computing. 92. 10.1007/s10915-022-01939-z.
[6]. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. 2020. – 480 с.
[7]. Lu Lu, et al. DeepXDE: A Deep Learning Library for Solving Differential Equations. SIAM REVIEW. 2021. Vol. 63, No. 1, pp. 208–228.
[8]. https://github.com/lululxvi/deepxde
[9]. Kovasznay L.I.G. Laminar flow behind a two-dimensional grid. Math. Proc. Cambridge. 1948. V. 44. Is. 1. P. 58–62.
[10]. Бубенчиков А.М., Попонин В.С., Мельникова В.Н. Разработка универсальной техники реализации неоднородных граничных условий Дирихле и Неймана в методе спектральных элементов, Вестн. Томск. гос. ун-та. Матем. и мех., 2008, номер 2(3), 87–98.
[11]. Dong S., Karniadakis G.E., Cryssostomidis C. A robust and accurate outflow boundary condition for incompressible flow simulations on severely-truncated unbounded domains. J. Comput. Phys. 2014. V. 261. P. 83–105.
[12]. Jin X., Cai S., Li H., Karniadakis G.E. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. Journal of Computational Physics, Volume 426, 2021, 109951.
[13]. Беликов В.В., Алексюк А.И. Модели мелкой воды в задачах речной гидродинамики. Российская академия наук, Отделение наук о Земле, Институт водных проблем РАН. – Москва: Российская академия наук, 2020. – 346 с. – ISBN 978-5-907036-22-2.
[14]. Зиновьев А.Т., Кошелев К.Б., Дьяченко А.В., Коломейцев А.А. Экстремальный дождевой паводок 2014 года в бассейне Верхней Оби: причины, прогноз и натурные наблюдения // Водное хозяйство России: проблемы, технологии, управление. – 2015. – № 6. – С. 93-104.
[15]. Cedillo S. et al. Physics-Informed Neural Network water surface predictability for 1D steady-state open channel cases with different flow types and complex bed profile shapes. Adv. Model. and Simul. in Eng. Sci. (2022) 9:10.
[16]. Rosofsky S.G. et al. Applications of physics informed neural operators. Mach. Learn.: Sci. Technol. 4 (2023) 025022.
[17]. Feng D., Tan Z., He Q.Z. Physics-informed neural networks of the Saint-Venant equations for downscaling a large-scale river model. *Water Resources Research*, (2023). *59*, e2022WR033168.

Информация об авторах / Information about authors

Константин Борисович КОШЕЛЕВ – кандидат физико-математических наук, доцент, старший научный сотрудник Института водных и экологических проблем СО РАН. Сфера научных интересов: вычислительная гидродинамика, гидрология, геоинформатика.

Konstantin Borisovich KOSHELEV – Cand. Sci. (Phys.-Math.), associate professor, senior researcher at the Institute for water and environmental problems of the Siberian branch of the RAS. Research interests: computational fluid dynamics.

Сергей Владимирович СТРИЖАК – кандидат технических наук, ведущий инженер Института системного программирования им. В.П. Иванникова РАН с 2009 года. Сфера научных интересов: вычислительная гидродинамика, многофазные течения, турбулентность, ветроэнергетика, параллельные вычисления.

Sergei Vladimirovich STRIJHAK – Cand. Sci. (Tech.), leading engineer of the Ivannikov Institute for System Programming of the RAS since 2009. Research interests: computational fluid dynamics.

DOI: 10.15514/ISPRAS-2023-35(5)-17

Моделирование процесса обледенения корпуса рыболовецкого судна на поверхности воды с учетом влияния волнения

[1,2] *К.Б. Кошелев, ORCID:* 0000-0002-7124-3945<*koshelevkb@mail.ru*>
[2] *А.В. Осипов, ORCID:* 0000-0001-9223-4274 <*a.osipov@ispras.ru*>
[2] *С.В. Стрижак, ORCID:* 0000-0001-5525-5180 <*s.strijhak@ispras.ru*>

[1] *Институт водных и экологических проблем СО РАН,*
656038, Алтайский край, г. Барнаул, ул. Молодежная, д.1.
[2] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Абстракт: Изучение обледенения судов является актуальной задачей в связи с судоходством в морях Северного Ледовитого океана. В работе рассматривается задача моделирования обтекания модельного рыболовецкого судна газокапельным потоком и возникновение процесса обледенения. Первоначально моделирование выполнялось с помощью решателя interDyMFoam с учетом задания волны Стокса первого рода для определения положения капель. В дальнейшем моделирование было выполнено с помощью решателя iceFoam в основе которого используется Эйлер-Лагранжев метод для описания газокапельного потока. Рассмотренная модель рыболовецкого судна имела масштаб 1:10. Положение капель задавалось на входе в расчетную прямоугольную область. Расчетная сетка имела от 1.5 до 10 млн. ячеек. С помощью расчетов были получены траектории движения капель вокруг корпуса судна, распределение поля скорости воздуха, положение пленки воды и толщина льда на поверхности палубы. Была выполнена оценка массы наросшего льда. Моделирование выполнялось на вычислительном кластере ИСП РАН. Один типовой пример запускался на 48-96 вычислительных ядрах и продолжался не более трех дней.

Ключевые слова: обледенение, судно; волнение; область; сетка; расчет; капли; размер; скорость; температура; пленка воды; пленка льда; масса льда.

Для цитирования: Кошелев К.Б., Осипов А.В., Стрижак С.В. Моделирование процесса обледенения корпуса рыболовецкого судна на поверхности воды с учетом влияния волнения. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 259–270. DOI: 10.15514/ISPRAS–2023–35(5)–17.

Modeling the Icing Process for the Hull of a Fishing Vessel on the Surface of the Water, Taking into Account the Influence of Waves

[1,2] *K.B. Koshelev, ORCID:* 0000-0002-7124-3945<*koshelevkb@mail.ru*>
[2] *A.V. Osipov, ORCID:* 0000-0001-9223-4274 <*a.osipov@ispras.ru*>
[2] *S.V. Strijhak, ORCID:* 0000-0001-5525-5180 <*s.strijhak@ispras.ru*>

[1] *Institute for water and environmental problems SB RAS,*
1, Molodezhnaya str., Altai region, Barnaul, 656038.
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn str., Moscow, 109004, Russia.

Abstract: Studying the icing of ships is an urgent task. The paper considers the problem of modeling the flow of a model vessel with a gas-droplet flow and the occurrence of the icing process. Initially, the simulation was performed using the interDyMFoam solver, taking into account the assignment of the Stokes wave of the first kind to determine the position of the droplets. Further modeling was carried out using the iceFoam solver, which is based on the Euler-Lagrangian method for describing the gas-droplet flow. The considered model of a fishing vessel had a scale of 1:10. The position of the droplets was set at the entrance to the calculated rectangular domain. The estimated grid had from 1.5 to 10 million cells. With the help of calculations, the trajectories of droplet movement around the hull of the vessel, the distribution of the air velocity field, the position of the water film and the thickness of ice on the deck surface were obtained. The mass of the overgrown ice was estimated. The simulation was performed on the computing cluster of the ISP RAS. One typical calculation was run on 48-96 computing cores and lasted no more than three days.

Keywords: icing, vessel; excitement; area; grid; calculation; drops; size; speed; temperature; water film; ice film; ice mass.

For citation: Koshelev K.B., Osipov A.V., Strijhak S.V. Modeling the icing process for the hull of a fishing vessel on the surface of the water, taking into account the influence of waves. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 259-270 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-17.

1. Введение

Изучение обледенения судов в морской науке является актуальной задачей в связи с эксплуатацией кораблей и рыболовецких судов в морях Северного Ледовитого океана, например, в Баренцевом и Карском морях. Российское рыболовецкое судно “Онега” затонуло в Баренцевом море 28 декабря 2020 года. В результате происшествия девятнадцать человек погибли. Спасательная операция была осложнена четырехметровыми волнами и 20-градусным морозом. Одной из возможных причин стало обледенение и потеря управляемости судна. Когда судно внезапно обледенело, то необходимо правильно оценить изменения массы и моментов инерции судна, чтобы обеспечить остойчивость судна и его безопасное положение на воде, а также принять меры по устранению льда.

Таким образом, при движении судна на открытой воде при резком изменении погоды (порывы ветра, температура, давление, осадки, влажность) возможно образование больших волн и брызг, соударение волн с корпусом судна, движение капель вокруг корпуса судна, с последующим нарастанием льда и потерей устойчивости судна на поверхности воды.

Согласно литературным источникам, существует следующая градация обледенения [1]:

1) Медленное обледенение происходит для температуры – 1 °C до -3 °C и при скорости ветра 0 м/с - 9 м/с и температуре ниже -3 °C;
2) Быстрое обледенение происходит при скорости ветра от 9 до 15 м/с и температуре воздуха от -3°C до - 8 °C;
3) Очень быстрое обледенение происходит при скорости ветра выше 15 м/с и температуре ниже - 3°C, а также при скорости ветра от 9 м/с до 15 м/с и температуре ниже – 8 °C.

Согласно последнему руководству службы прогнозирования Росгидромета (Россия), интенсивность обледенения рыболовецких судов характеризуется следующими показателями:

1) Медленный режим: скорость обледенения составляет менее 0,007 м/ч;
2) Быстрый режим: скорость обледенения от 0,007 м/ч до 0,013 м/ч (характеризуется как опасное явление;
3) Очень быстрый режим: скорость обледенения составляет 0,014 м/ч или более (характеризуется как опасное явление).

Ранее процесс обледенения был изучен для различных типов судов и морских сооружений в сложных климатических условиях [2-4]. Полевые измерения процесса обледенения были проведены на 39-метровом российском рыболовецком судне “MFV Narva” в Японском море в феврале 1973 года [2], а поток брызг был изучен на 115-метровом катере береговой охраны США (USCGC) Midgett в северной части Тихого океана и Беринговом море в течение февраля и марта 1990 года [3]. Эти полевые исследования уникальны, дорогостоящи, опасны для экипажа судна и требуют специального измерительного оборудования [5-8].

Как правило, для изучения обледенения используются натурный и лабораторный эксперименты, а также математическое моделирование на суперкомпьютере.

Программное обеспечение с открытым исходным кодом широко используется для моделирования гидродинамики судов при сильном волнении. Одним из успешных проектов с открытым исходным кодом в области вычислительной механики является код OpenFOAM, реализованный на языке программирования C++ [9].

Библиотека SNUFOAM была разработана в работе [10], авторы которой применили библиотеку с открытым исходным кодом для задач судостроения и морской гидродинамики. Shen и Wan разработали библиотеку naoe-FOAM-SJTU, основанную на коде OpenFOAM, с целью моделирования различных морских гидродинамических задач. Авторы предсказали дополнительное сопротивление для судна в случае задания различной формы морских волн [11]. В библиотеке naoe-FOAM-SJTU был реализован метод динамической сетки, были вычислены характеристики движения судна и морских платформ с 6 степенями свободы.

Vukcevic et al. предложили гидродинамическую модель, объединив коды для невязкого случая течения и вязкого течения [12-13]. В работе [14] были изучены вопросы неопределенности размеров ячейки сетки, временного шага и определена возможность использования OpenFOAM для расчета характеристик сопротивления судна [15].

Модель грузового судна KCS была рассчитана и результаты расчета сравнены с экспериментальным результатом для проверки промежуточной точности решателя. В результате было достигнуто относительно хорошее согласование. Авторы работы также изучили модель движения индонезийского традиционного рыболовецкого судна с северного до южного побережья острова Ява.

2. Математическая модель

В настоящем исследовании CFD-решатели с открытым исходным кодом использовались для расчета сопротивления и движения модели рыболовного судна при регулярных встречных волнах, для моделирования процесса образования капель и процесса нарастания льда на внешней поверхности корпуса судна.

На первом этапе рассматривалось моделирование движения судна на поверхности воды с учетом влияния волны. Моделирование проводилось с использованием решателя interDyMFoam и специального граничного условия, метода VOF, реализованного в рамках программы OpenFOAM, который основан на методе конечных объемов [9]. В результате моделирования было определено положение объемной струи воды и образовавшиеся капли

воды. Для задания волновой поверхности использовалась волна Стокса первого порядка, которая имела форму, показанную на рис. 1.

Задание характеристик для волны Стокса (соотношения, частота, амплитуда) приведено в различных публикациях и монографиях [10-11,19-20].

На втором этапе было смоделировано движение капель в набегающем потоке воздуха, воздействие капель на поверхность модельного судна, а также процесс срастания льда и образования водной пленки. Уравнения непрерывности и Навье-Стокса являлись основными уравнениями, описывающие движение жидкости. Для описания вязкой турбулентной жидкости использовалась модель с уравнениями Рейнольдса в составе решателя interDyMFoam. Граница раздела воздух-вода вычислялась с помощью метода объема в жидкости (Volume of Fluid). Уравнения Рейнольдса замыкались двумя уравнениями по модели турбулентности SST, предложенной Ментером. Модель турбулентности переноса сдвигового напряжения Ментера SST представляет собой модель вихревой вязкости с двумя уравнениями и пристеночными функциями. Данная модель использовалась для многих гидродинамических и аэродинамических приложений.

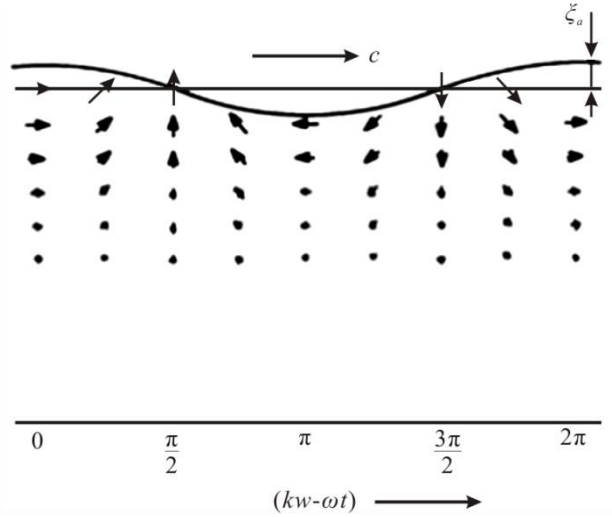


Рис. 1. Волна Стокса первого порядка.
Fig. 1. The Stokes wave of the first order.

Модель сочетает в себе хорошо известные низкорейнольдсовую и высокорейнольдсовую модели турбулентности. Первая модель подходит для моделирования течения в вязком подслое, в то время как вторая модель идеально подходит для прогнозирования поведения потока в областях, удаленных от стенки.

Моделирование нарастания льда проводилось с помощью решателя iceFoam с использованием подхода Эйлера-Лагранжа, по модели жидкой пленки по теории мелкой воды (SWIM) [16, 17], модуля динамической сетки. Модель пленки по теории мелкой воды была протестирована для различных аэродинамических профилей и трехмерных стреловидных крыльев. Результаты определения аэродинамических коэффициентов и толщины льда были сопоставлены с экспериментальными данными [17,18].

Для модели рыболовецкого судна проводилось моделирование гидродинамики судна с учетом волнения, образования водной струи у носовой части судна, движения капель и образованием наледи на поверхности модельного тела.

3. Постановка задачи

Рассматривалось типичное рыболовецкое судно, показанное на рис. 2. Данное судно имело следующие размеры в масштабе 1:1: L=34 м, W=8 м, H=3.6 м, Depth_water=1 м.

Для разрешения геометрических форм надстроек палубы необходимо большое число элементов сетки и это введет к большим вычислительным ресурсам, которые не всегда доступны. Первая оценка показывает, что размер сеток может быть от 40 до 100 млн ячеек.

С целью упрощения проведения расчетов трехмерная цифровая модель судна была упрощена с помощью программного обеспечения Salome для трехмерного моделирования и построения расчетных сеток. Нами были удалены такие сложные элементы, как антенна, кабели, краны, мелкие металлические предметы.

В результате в цифровой модели остались только наиболее важные элементы: капитанская рубка и корабельная палуба (рис. 3). Модель была сохранена в формате STL. Рассматриваемая модель судна для дальнейшего моделирования имела масштаб 1:10.

В результате исследуемая модель имела характеристики: L=3.4 м, W=0.8 м, H=0.36 м, Depth_water=0.1 м; m=150 кг. Центр масс имел координаты в точке (-0.048, 0.104, 0.314). Значения моментов инерции: (1.044, 0.0005, 0.0001, 0.072, 0.075, 1.04).

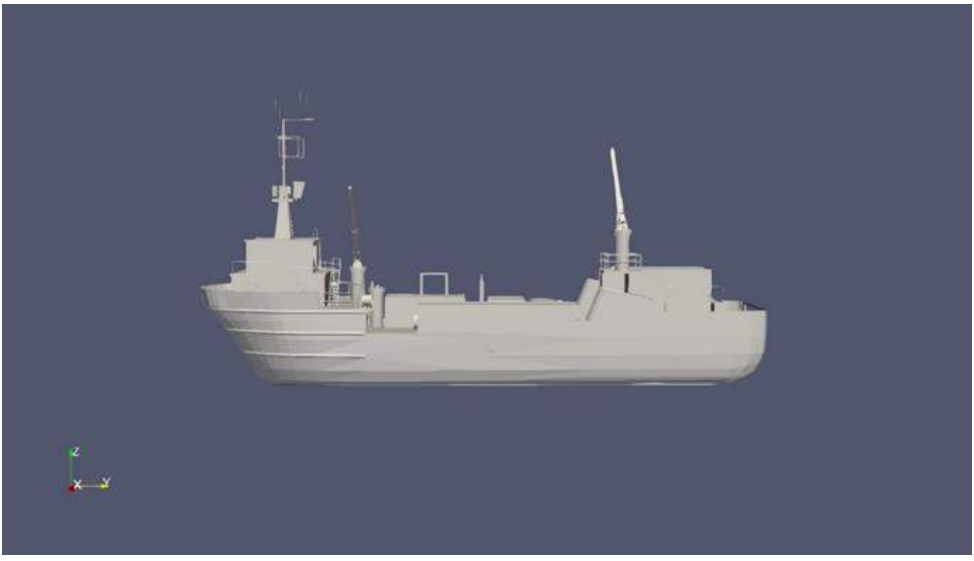


Рис. 2. Рыболовецкое судно в масштабе 1:1.
Fig. 2. The fishing vessel in 1:1 scale.

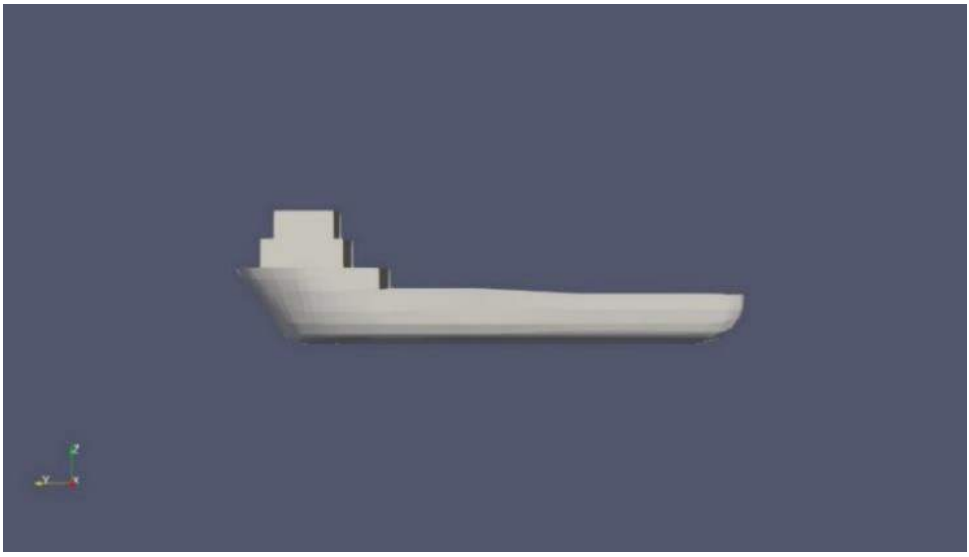


Рис. 3. Упрощенная модель судна в масштабе 1:10.
Fig. 3. The simplified model of the vessel with size 1:10.

Для проведения расчетов была подготовлена прямоугольная расчетная область с размерами: $5L \times 2L \times 3L$, где L это длина судна.

Было построено несколько неструктурированных сеток от 1.5 млн до 10 млн ячеек. Окончательно была выбрана сетка с 3 миллионами ячеек (рис. 4). Расчетная сетка была построена с помощью встроенной утилиты snappyHexMesh. Сгущение сетки происходило около раздела двух фаз воздух-вода и около корпуса судна. На входе расчетной области задавались граничные условия для волнения по модели волн Стокса. Также задавалось начальное распределение необходимых величин для двух фаз воздух-вода.

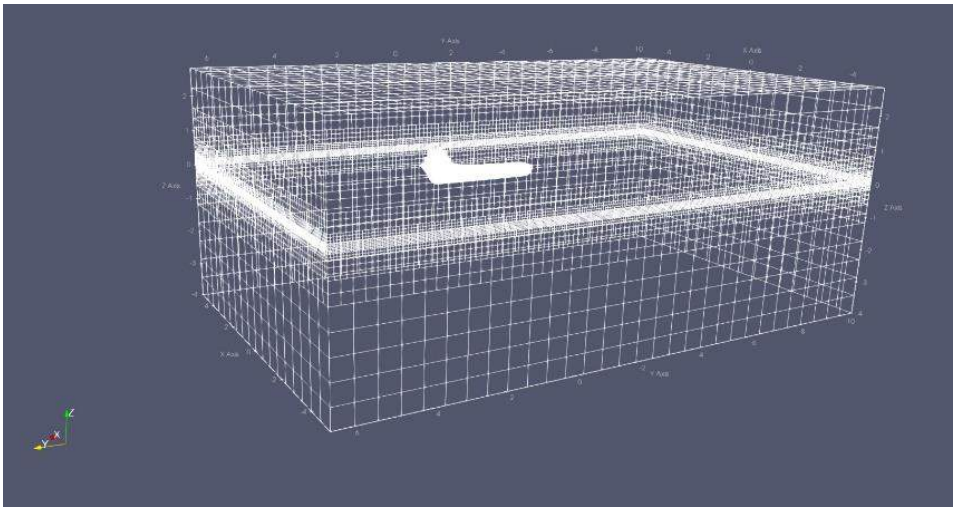


Рис. 4. Расчетная область для судна.
Fig. 4. The numerical domain for the vessel.

На входе вычислительной области были заданы граничные условия для волны в соответствии с волновой моделью Стокса первого рода. Также было задано начальное распределение

требуемых значений для воздушной и водной фаз. В ходе расчета было изучено положение потока массы воды и капель на носу судна для заданного числа Фруда до *Fr=0.4*.

Гидродинамика судна изучалась с учетом влияния регулярных волн, образования струи воды вблизи носа судна, движение капель вблизи поверхности корпуса модели. Параметры для моделирования определены в таблице 1.

Табл. 1. Параметры для расчетов
Table 1. Parameters for calculations

Parameters	Value
Re	10^5 - 3.5×10^6
Fr	0.1-0.4
Ватерлиния	0.15-0.20
U_{air} /(m·s^{-1})	3-8

4. Результаты расчета

В ходе расчета было определено положение водной струи и капель у носовой части судна с помощью гидродинамического решателя по модели “жидкость в объеме” для чисел Фруда Fr=0.1-0.4. Длительность численного эксперимента для движения модельного судна на волнении в виртуальном бассейне составило t=50 секунд. Результаты моделирования для объемной доли *для Fr=0.4* представлены на рисунке 5. Видно образование восходящей струи у судна и положение капель. Для задания начального положения облака брызг (капель), необходимого для расчета обледенения с помощью решателя iceFoam на основе результатов вычислений волнения было определено сечение, в котором оценены такие параметры как вектор скорости капель и их массовый расход.

Далее с помощью решателя iceFoam при заданном положении капель в расчетной области и скорости ветра было проведено моделирование процесса обледенения и получено распределение жидкой пленки воды, толщины льда на поверхности судна. Детальная постановка математической модели движения капель в турбулентном потоке газа и нарастания льда по поверхности обтекаемого тела приведены в [17, 18].

Табл. 2. Задание граничных условий для моделирования волнений
Table 2. Specifying boundary conditions for wave modeling

	Inlet	Outlet	Atmosphere	Ship
U	waveVelocity	zeroGradient	pressureInletOutletVelocity	movingWallVelocity
$\boldsymbol{p_{rgh}}$	fixedFluxPressure	zeroGradient	totalPressure	fixedFluxPressure
$\boldsymbol{\alpha_{water}}$	waveAlpha	zeroGradient	inletOutlet	zeroGradient
k	fixedValue	zeroGradient	inletOutlet	kqRWF
nut	fixedValue	zeroGradient	zeroGradient	nutkWF
omega	fixedValue	zeroGradient	inletOutlet	omegaWF

Численные эксперименты проведены для случаев *MVD=300-1200 мкм*, скорости ветра U_{wind}= *3 м/с* и скорости судна *V = 5 м/с*. Температуры воздуха и капель во входном сечении постоянны во всех расчетных вариантах, *T = 270°K*. Время расчета составило *t = 8* секунд. В

ходе расчета определялись траектории движения капель (рис. 6). Большая часть облака капель оседала на поверхности палубы. При этом образовывалась небольшая застойная воздушная зона, сразу за капитанской рубкой, куда капли не достигали.

Также в ходе расчета было определено распределение поля температуры (рис. 7).

Дополнительно получено распределение для толщины пленки воды (рис.8). Наиболее крупное образование пленки воды происходило на носовой части судна, на капитанской рубке, в задней части поверхности кормы.

На рис. 9 показано распределение толщины льда. Основная часть образовавшегося льда распределялась по поверхности палубы.

Был выполнен расчет для случая *MVD=1000 мкм*, скорости ветра U_{wind}*= 10 м/с* и скорости судна *V = 5 м/с.* Траектория движения капель была уже другой за счет большего значения скорости потока воздуха. Основная часть капель не достигала поверхности палубы и перемещалась в дальний след. Зависимость массы образовавшегося льда от диаметра капель при прочих равных условиях представлена на рис. 10.

Время расчета для одного типового примера составило около 24 часов с interDyMFoam и 30 часов с iceFoam-решателем. Вычисления проводились на вычислительном кластере ИСП РАН в параллельном режиме с использованием 48 или 96 ядер процессорных ядер для одного расчетного примера.

5. Заключение

Проведено первоначальное исследование обтекания корпуса модельного судна и процесса образования наледи на поверхности палубы с использованием открытых решателей interDyMFoam и iceFoam. Температура воздуха и капель соответствовала режиму гладкого льда (glaze ice), который является наиболее трудоемким для вычислений.

Расчетные результаты показали нелинейную зависимость распределения положения льда на поверхности судна от размера капель при прочих равных условиях. Однако, зависимость массы образовавшегося льда от диаметра капель (брызг) оказалась очень близка к линейной.

Дальнейшее развитие может быть направлено на улучшение реалистичности модели судна, расчет обтекания и обледенения полномасштабной модели, уточнение параметров облака брызг, совершенствование модели взаимодействия капель между собой и поверхности судна.

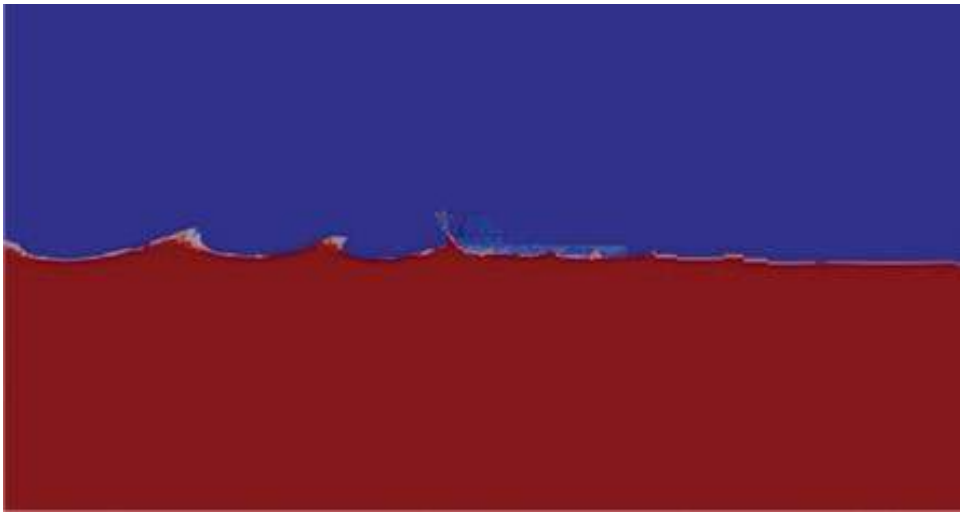


Рис. 5. Значение величины объемной доли α для Fr=0.4.
Fig. 5. The volume fraction for t=50 s at Fr = 0.4.

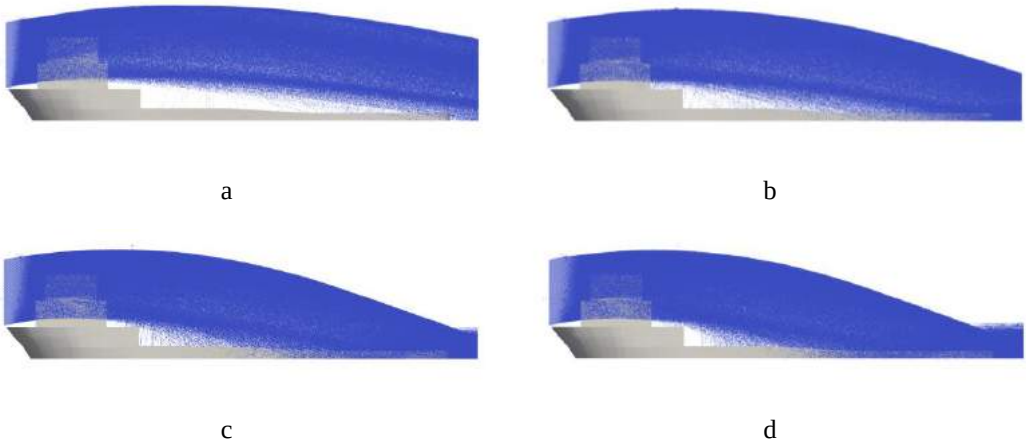


Рис. 6. Траектории движения капель вблизи судна при U_{wind} = 3 м/с,
a) mvd =300 мкм, b) mvd =600 мкм, c) mvd =1000 мкм, d) mvd =1200 мкм.
Fig. 6. The droplet trajectories near the ship at U_{wind} = 3 m/s,
a) mvd =300 μm, b) mvd =600 μm, c) mvd =1000 μm, d) mvd =1200 μm.

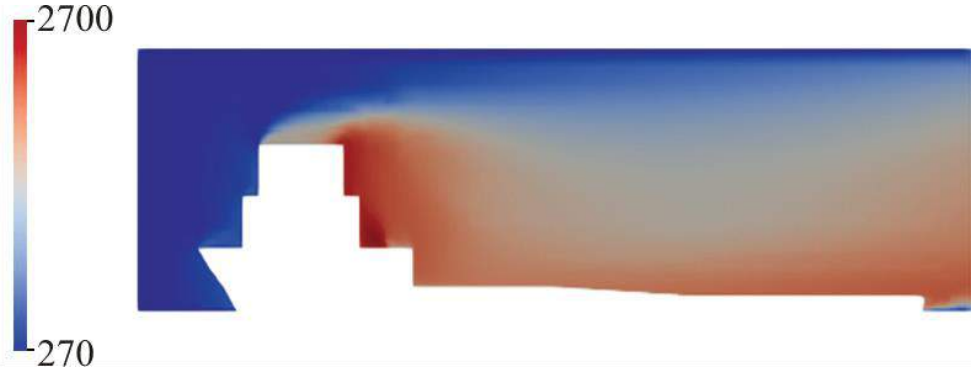
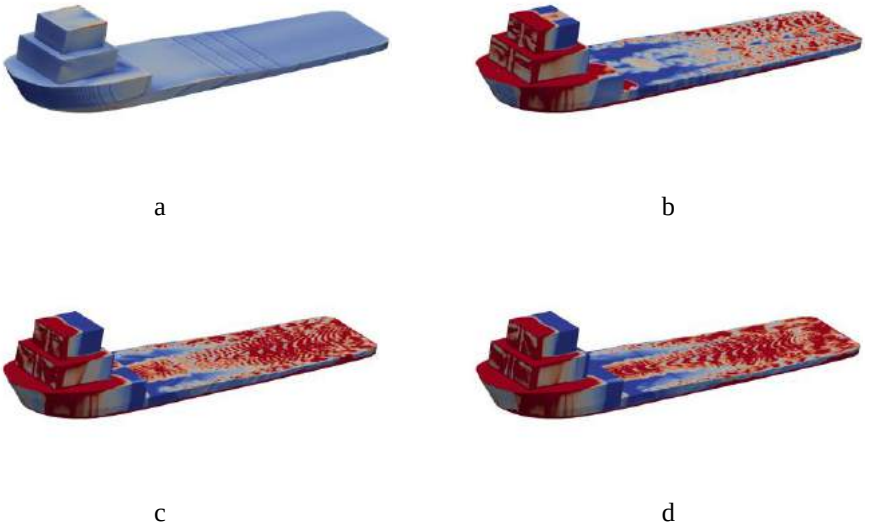


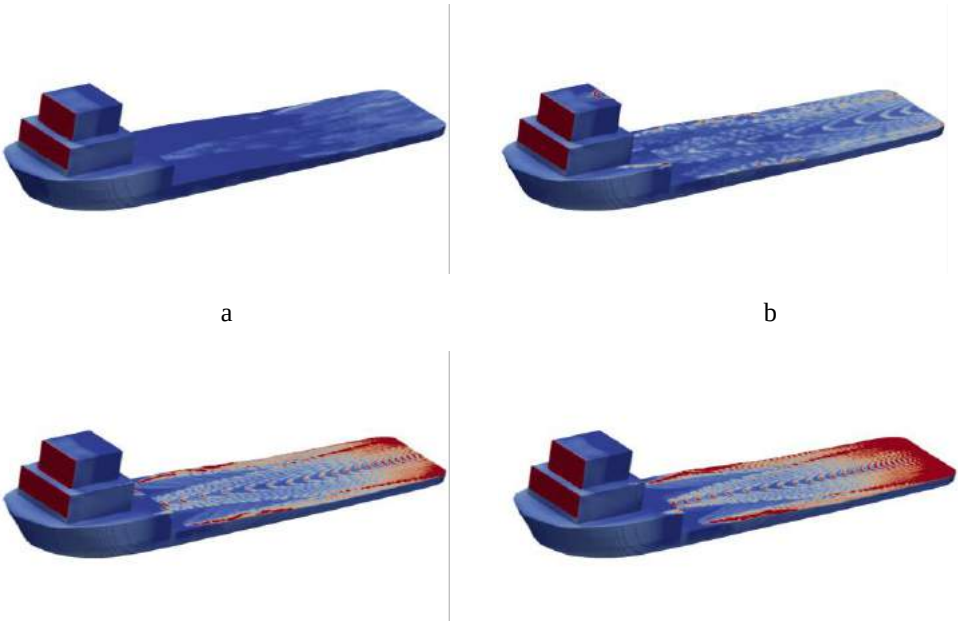
Рис. 7. Поле распределение температуры для U_{wind} = 3 м/с.
Fig. 7. The temperature distribution of air for U_{wind} = 3 m/s.



a b

c d

Рис. 8. Распределение пленки воды по поверхности корпуса судна для U_{wind} = 3 м/с, a) mvd= 300 мкм, b) mvd =600 мкм, c) mvd =1000 мкм, d) mvd =1200 мкм.
Fig. 8. The water film distribution over the surface of the ship hull for U_{wind} = 3 m/s, a) mvd= 300 μm, b) mvd =600 μm, c) mvd =1000 μm, d) mvd =1200 μm.



a b

c d

Рис. 9. Распределение наросшего льда по поверхности корпуса судна для U_{wind} = 3 m/s, a) mvd= 300 мкм, b) mvd =600 мкм, c) mvd =1000 мкм, d) mvd =1200 мкм.
Fig. 9. The distribution of ice accretion on the ship hull surface for U_{wind} = 3 m/s a) mvd = 300 μm, b) mvd =600 μm, c) mvd =1000 μm, d) mvd =1200 μm.

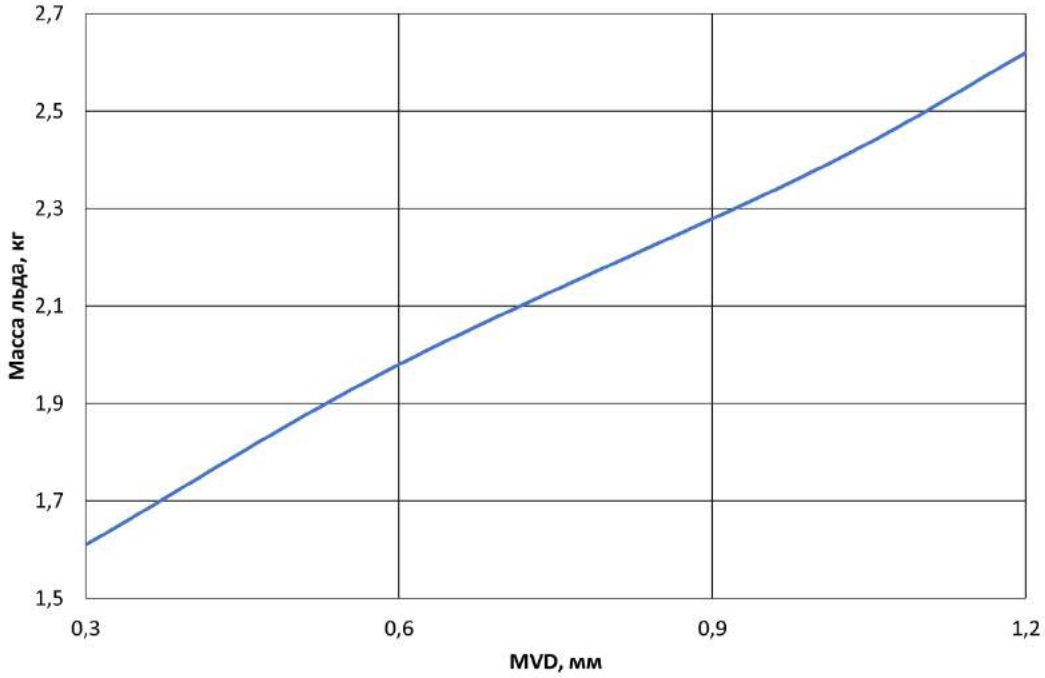


Рис. 10. Зависимость массы образовавшегося льда от диаметра капель при U_{wind} = 10 м/с.
Fig. 10. Dependence of ice mass on droplets' diameter at U_{wind} = 10 m/s.

Список литературы / References

[1]. Качурин Л.Г., Смирнов И.А., Гашин Л.И. Обледенение судов. Учебное пособие. Ленинградский гидрометеорологический институт (ЛГМИ), 1980, 56 с.
[2]. Panov V. V. Vessel Icing. Proceedings AARI, Saint Petersburg, Russia, 1976, 262.
[3]. Zakrzewski W. P. Splashing a ship with collision-generated spray. *Journal of Cold Regions Science and Technology*, 1987, 14(1): 65-83.
[4]. Ryerson C. C. Superstructure spray and ice accretion on a large U.S. Coast Guard cutter. *Journal of Atmospheric Research*, 1995, 36(3-4): 321-337.
[5]. Samuelsen E. M., Edvardsen K., Graversen R. G. Modelled and observed sea-spray icing in Arctic-Norwegian waters. *Journal of Cold Regions Science and Technology*, 2017, 134: 54-81.
[6]. Mintu S., Molyneux D., Oldford D. A state-of-the-art review of research on ice accretion measurements and modelling. Arctic Technology Conference, St. John's, Canada, 2016, 1- 19.
[7]. Mintu S., Molyneux D., Colboume B. A Theoretical model for ship–Wave impact generated sea spray. *Journal of Offshore Mechanics and Arctic Engineering*, 2020, 143(4): 041201.
[8]. Mintu S., Molyneux D. Ice accretion for ships and offshore structures. Part 2–Compilation of data. *Ocean Engineering*, 2022, 248: 110638.
[9]. Weller H. G., Tabor G., Jasak H. et al. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Journal of Computers in Physics*, 1998, 12: 620-631.
[10]. Seo S., Park S., Koo B. Effect of wave periods on added resistance and motions of a ship in head sea simulations. *Ocean Engineering*, 2017, 137: 309-327.
[11]. Wang J. H., Zhao W. W., Wan D. C. Development of naoe- FOAM-SJTU solver based on OpenFOAM for marine hydrodynamics. *Journal of Hydrodynamics*, 2019, 31(1): 1-20.
[12]. Vukcevic V., Jasak H., Malenica S. Decomposition model for naval hydrodynamic applications, Part I: Computational method. *Ocean Engineering*, 2016, 121: 37-46.
[13]. Vukcevic V., Jasak H., Malenica S. Decomposition model for naval hydrodynamic applications, Part II: verification and validation. *Ocean Engineering*, 2016, 121: 76-88.
[14]. Seo S., Park S., Koo B. Effect of wave periods on added resistance and motions of a ship in head sea simulations. *Ocean Engineering*, 2017, 137: 309-327.

[15]. Bahatmaka A., Kim D. J. Numerical modelling for traditional fishing vessel prediction of resistance by CFD approach. *International Journal of Applied Engineering Research*, 2018, 13(8): 6211-6215.
[16]. Bourgault Y., Beaugendre H., Habashi W. G. Development of a shallow-water icing model in FENSAP-ICE. *Journal of Aircraft*, 2000, 37(4): 640-646.
[17]. Koshelev K. B., Melnikova V. G. Strijhak S. V. Development of iceFom solver for modeling ice accretion. *Proceedings of the Institute for System Programming of the RAS*, 2020, 32(4): 217-234 (in Russian).
[18]. Strijhak S., Ryazanov D., Koshelev K., Ivanov A. A neural network prediction for ice shapes on airfoils using icefoam simulations. *Aerospace*, 2022, 9(2): 96.
[19]. Бэтчелор Дж.К. Введение в динамику жидкости. Москва. Ижевск: НИЦ ”Регулярная и хаотическая динамика“, 2004., 768 с.
[20]. Ландау Л., Лифшиц Е. Теоретическая физика. В десяти томах. Том VI. Гидродинамика. Издательство Физматлит, 2017. 728 стр.

Информация об авторах / Information about authors

Константин Борисович КОШЕЛЕВ – кандидат физико-математических наук, доцент, старший научный сотрудник Института водных и экологических проблем СО РАН. Сфера научных интересов: вычислительная гидродинамика, гидрология, геоинформатика.

Konstantin Borisovich KOSHELEV – Cand. Sci. (Phys.-Math.), associate professor, senior researcher at the Institute for water and environmental problems of the Siberian branch of the RAS. Research interests: computational fluid dynamics.

Андрей Владимирович ОСИПОВ – инженер Института системного программирования им. В.П. Иванникова РАН с 2018 года. Сфера научных интересов: вычислительная гидродинамика, метод контрольного объема, подвижные сетки, лагранжев подход.

Andrei Vladimirovich OSIPOV – engineer of the Ivannikov Institute for System Programming of the RAS since 2018. Research interests: computational fluid dynamics, finite volume method, dynamic meshes, particles.

Сергей Владимирович СТРИЖАК – кандидат технических наук, ведущий инженер Института системного программирования им. В.П. Иванникова РАН с 2009 года. Сфера научных интересов: вычислительная гидродинамика, многофазные течения, турбулентность, ветроэнергетика, параллельные вычисления.

Sergei Vladimirovich STRIJHAK – candidate of technical sciences, leading engineer of the Ivannikov Institute for System Programming of the RAS since 2009. Research interests: computational fluid dynamics.

DOI: 10.15514/ISPRAS-2023-35(5)-18

Моделирование динамики электризованного потока частиц при ветровом выносе средствами OpenFoam

Е.А. Малиновская, ORCID: 0000-0003-0385-0396 <*elen_am@inbox.ru*>,
Г.И. Горчаков, *ORCID:* 0009-0002-6454-8326 <*gengor@ifaran.ru*>,
А.В. Карпов, *ORCID:* 0009-0004-8906-8200 <*karpov@ifaran.ru*>,
Л.О. Максименков, *ORCID:* 0000-0002-1909-0777 <*maksimenkov@ifaran.ru*>,
О.И. Даценко, *ORCID:* 0009-0009-6596-3805 <*datsenko@ifaran.ru*>

*Институт физики атмосферы им. А.М. Обухова РАН,
119017, Россия, Москва, Пыжевский пер. 3.*

Аннотация. Исследуется генерация пылевого аэрозоля при скачкообразном каскадном движении заряженных частиц над неровной поверхностью под влиянием ветра. Частицы движутся над двумя элементами типа ряби на эоловой поверхности под влиянием воздушного потока. За препятствиями поток сальтирующих частиц становится неравномерным, характер движения отмечается квазипериодичностью. Решалась задача включения электростатических эффектов в гидродинамическую модель, в которой учтено взаимовлияние частиц и воздушной среды. Предложена параметрическая модель, позволяющая учитывать в моделировании ветрового выноса заряженность самих пылевых частиц и подстилающей поверхности. Вычислительные эксперименты проведены с использованием открытого пакета OpenFOAM – Эйлерово-Лагранжевая турбулентная k-ω-модель. Соответственно, динамика заряженных частиц рассматривается с учётом электризации самой поверхности. Из результатов вычислительных экспериментов для различных плотностных характеристик частиц, заряженных одноименно с поверхностью, оценено влияние электрического поля на частоту изменения числа частиц в потоке, на разброс значений скоростей движения и высоту подскоков частиц, а также на ослабление эффекта воздействия частиц на среду за препятствиями. При учете влияния электростатических эффектов выявлено усиление возмущающего воздействия частиц, вылетающих после препятствий, на воздушную среду (увеличивается расстояние от препятствия, появляется больше локальных областей возмущения). Для скоростей движения сальтирующих частиц отмечается уменьшение величины дисперсии. Высота подскоков частиц увеличивается, что подтверждается известными экспериментами. Уменьшается нижнее значение характерных частот изменения числа частиц в потоке. Неравномерность потока частиц определяет изменения в интенсивности генерации пылевого аэрозоля.

Ключевые слова: сальтации частиц, численное моделирование движения частиц в гидродинамическом потоке, электрическое поле.

Для цитирования: Малиновская Е.А., Горчаков Г.И., Карпов А.В., Максименков Л.О., Даценко О.И. Моделирование динамики электризованного потока частиц при ветровом выносе средствами OpenFoam. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 271–286. DOI: 10.15514/ISPRAS–2023–35(5)–18.

Благодарности: Исследование выполнено при поддержке Российского научного фонда – проект №23-27-00480 "Исследование генерации и выноса пылевого аэрозоля над аридными территориями в условиях неоднородности рельефа и температуры". Автор выражают благодарность О.Г. Чхетиани за полезные консультации и обсуждения.

Modelling the Dynamics of Electrified Particle Flow during Wind Drift Using Openfoam

E.A.Malinovskaya, ORCID: 0000-0003-0385-0396 <*elen_am@inbox.ru*>.
G.I.Gorchakov, ORCID*:* 0009-0002-6454-8326 <*gengor@ifaran.ru*>,
A.V. Karpov, ORCID*:* 0009-0004-8906-8200 <*karpov@ifaran.ru*>,
L.O. Maksimenkov, ORCID*:* 0000-0002-1909-0777 <*maksimenkov@ifaran.ru*>,
O.I. Datsenko, ORCID*:* 0009-0009-6596-3805 <*datsenko@ifaran.ru*>

A.M. Obukhov Institute of Atmospheric Physics RAS,
119017, Russia, Moscow, Pyzhevsky per., 3.

Abstract. We study the generation of dust aerosol in the wind-driven cascading motion of charged particles over an irregular surface. The particles move under the influence of air flow over two elements of ripple type on an aeolian surface. Behind the obstacles the flow of saltation particles becomes non-uniform, the character of motion is noted by quasi-periodicity. The problem of including electrostatic effects into the hydrodynamic model, in which the mutual influence of particles and air medium is taken into account, was solved. A parametric model is proposed, which allows taking into account the chargeability of dust particles and the underlying surface in modeling wind transport. Computational experiments are carried out using the open source OpenFOAM package, the Eulerian-Lagrangian turbulent k-ω-model. Accordingly, the dynamics of charged particles is considered taking into account the electrification of the surface itself. From the results of computational experiments for different density characteristics of particles charged homonymously with the surface, the influence of the electric field on the frequency of change of the number of particles in the flow, on the scattering of values of velocities and the height of particle hops, as well as on the weakening of the effect of particles on the medium behind obstacles is estimated. When the influence of electrostatic effects is taken into account, an increase in the disturbing effect of particles flying after obstacles on the air medium is revealed (the distance from the obstacle increases, more local areas of disturbance appear). A decrease in the dispersion value is noted for the velocities of hopping particles. The height of particle jumps increases, which is confirmed by known experiments. The lower value of characteristic frequencies of change in the number of particles in the flow decreases. The non-uniformity of the particle flow determines changes in the intensity of dust aerosol generation.

Keywords: particle saltations, numerical modelling of particle motion in hydrodynamic flow, electric field.

For citation: Malinovskaya E.A., Gorchakov G.I., Karpov A.V., Maksimenkov L.O., Datsenko O.I. Modelling the dynamics of electrified particle flow during wind drift using OpenFoam. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 5, 2023. pp. 271-286 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-18.

Acknowledgements. The research was supported by the Russian Science Foundation, project №23-27-00480 "Study of generation and export of dust aerosol over arid territories under conditions of terrain and temperature inhomogeneities". Authors would also like to thank O.G. Chkhetiani for useful consultations and discussions.

1. Введение

Аридные и субаридные территории – основной источник пылевого аэрозоля [1], так как пыль оказывает влияние на конденсацию влаги и облакообразование, массопереноса веществ твердой и жидкой фазы, изменение радиационного баланса Земли [2]. При скоростях ветра, превышающих критические значения [3] (около 3.5–5 м/с на высоте 2 м), начинается сальтация – скачкообразное каскадное движение частиц с размерами порядка 80–150 мкм у подстилающей поверхности, которое способствует генерации пылевого аэрозоля в результате фрагментирования и откалывания в момент соударения с поверхностью частиц [1]. Выделяются следующие основные силы, регулирующие ветровой вынос с поверхности и дальнейшее перемещение частиц в потоке:

- сила, возникающая за счет разности давлений при обтекании частицы над частицей и под ней подъемная сила [4-7];

- сила, возникающая при вращении частицы в потоке силы Магнуса [4-7].

Недостаточно изучено влияние микроциркуляций у поверхности на подъем частиц [3,8-9], турбулентных течений [5-6] и микровихрей, возникающих вблизи поверхности [10].

Также электростатический эффект увеличивает (при положительном заряде) и уменьшает (для отрицательного заряда) высоту подскоков сальтирующих заряженных частиц [1]. Увеличивается в результате величина массового потока сальтирующих частиц [11]. Эти факторы влияют на величину скорости падения частицы на поверхность и их количество, что отражается на процессе генерации микрочастиц.

При полевых измерениях, выполненных на опустыненной территории в Калмыкии, выявлено, что концентрации субмикронной фракции аридного аэрозоля размерами 0,2-0,4 мкм меняются в зависимости от величины напряженности электрического поля. Усиление и ослабление поля зависит от скорости и направления ветра по отношению к основному направлению дюнных гряд [1]. Эмиссия пыли усиливается при увеличении величины напряженности электрического поля [12-13].

Выявлено возникновение характерных частот (от 10 до 200 Гц) для числа сальтирующих частиц в потоке над неровной подстилающей поверхностью, в том числе и в численном эксперименте средствами открытого пакета OpenFoam (Эйлерово-Лагранжевая турбулентная k-ω-модель) [14]. Это обстоятельство говорит о наличии существенных колебаний значения массового потока сальтирующих частиц, что влияет на изменения интенсивности генерации пылевого аэрозоля. Возникающий эффект связывается с наличием на подстилающей поверхности естественного микрорельефа – ряби (составленные из отдельных песчинок продолговатые естественные неровности высотой около 1 см на расстоянии от 10 до 25 см друг от друга).

Оценить высоту подскоков и изменение траектории отдельной частицы, сальтирующей над поверхностью, можно с использованием известных сведений о силах, действующих на частицу. Такие оценки применимы для ламинарного движения при наличии ровной поверхности. В статье предлагается применить методы численного моделирования для учета влияния на гидродинамические свойства среды постоянно присутствующего в естественных условиях эффекта заряженных частиц. Также необходимо учитывать фактор наличия микрорельефа подстилающей поверхности.

Процессы движения частиц песка в воздушном потоке моделируются с использованием турбулентных моделей LES методом дискретных элементов [15] и с Лагранжевыми частицами [16-17]. Влияние потока на связанные частицы рассмотрено в [18-19]. В численных экспериментах выявляется влияние эоловых форм рельефа, отдельных элементов [20-21] на характеристики воздушного потока и переносимых в нем песчаных частиц [15, 22], на появление турбулентных структур [23].

В проводимых в статье исследованиях применялся открытый пакет OpenFOAM [24]. Известны его приложения к решению задач моделирования структуры обтекающего потока вблизи эоловых форм рельефа [25-30]. В [25, 29-30] использовался решатель PimpleFOAM в задаче обтекания эоловых форм рельефа, а в [26] – PisoFOAM.

В следующем разделе приведено описание Эйлерово-Лагранжевой модели, реализованной в OpenFOAM.

2. Перенос частиц воздушным потоком в Эйлерово-Лагранжевой модели

Движение сальтирующих частиц в воздушном потоке со скоростью u реализуется при наличии градиента давления: $F_p = -\frac{\pi d_p^3}{6}\nabla p$. Происходит подъем под действием силы разности давлений:

$$F_D = C_D \frac{\pi d_p^2}{8} \rho \left(u - v_p\right) \left|u - v_p\right|, \quad (1)$$

где $C_D = \frac{24}{\mathrm{Re}}\left(1 + 0.15\,\mathrm{Re}^{0.687}\right)$, d_p – диаметр частиц, v_p – скорость движения частицы.

При этом действует сила тяжести:

$$F_g = m_p g \left(1 - \frac{\rho}{\rho_p}\right). \quad (2)$$

Взаимодействие между частицами в OpenFOAM реализовано в рамках модели мягких сфер на основе теории точечных контактов Герца. Силы взаимодействия i-й и j-й частиц зависят от контактных напряжений и упругости материалов с учетом адгезии [31] определяются как [32]:

$$F_{ij} = K\delta^{3/2} + \alpha\sqrt{K}\delta^{1/4}v,$$

δ – радиус контакта (местное смятие), $K = \frac{\sqrt{\frac{r_1 r_2}{r_1 + r_2}}}{3\left(\frac{1-\mu_1^2}{E_1} + \frac{1-\mu_2^2}{E_2}\right)}$, μ_1 и μ_2 – коэффициенты Пуассона, E_1 и E_2 – модули Юнга, α зависит от коэффициента восстановления скорости в результате падения $\psi = \frac{v}{v_0} = 0.69$, где v – скорость до столкновения, v_0 – скорость после восстановления.

Движение воздушного потока реализовано с использованием модели турбулентности RANS $k-\omega$ SST описывается уравнениями для турбулентной кинетической энергии и удельного значения диссипации ω [32]:

$$\frac{\partial k}{\partial t} + U_i \frac{\partial k}{\partial x_j} = P_k - \beta^* k\omega + \frac{\partial}{\partial x_j}\left[\left(\eta + \sigma_k \eta_T\right)\frac{\partial k}{\partial x_j}\right], \quad (3)$$

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta\omega^2 + \frac{\partial}{\partial x_j}\left[\left(\eta + \sigma_\omega \eta_T\right)\frac{\partial k}{\partial x_j}\right] + 2\left(1 - F_1\right)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, \quad (4)$$

$\eta_T = \frac{\alpha_1 k}{\max\left(\alpha_1\omega, SF_2\right)}$ – кинематическая турбулентная вязкость,

$$F_2 = \tanh\left[\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\eta}{y^2\omega}\right)\right]^2\right],\ P_k = \min\left(\tau_{ij}\frac{\partial U_i}{\partial x_j}, 10\beta^* k\omega\right),$$

$$F_1 = \tanh\left\{\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\eta}{y^2\omega}\right), \frac{4\sigma_{\omega 2} k}{CD_{k\omega} y^2}\right]\right\}^4\right\},$$

$$CD_{k\omega} = \max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}\right),\ \varphi = \varphi_1 F_1 + \varphi_2\left(1 - F_1\right),\ \alpha_1 = \frac{5}{9}, \alpha_2 = 0.44,$$

$$\beta_1 = \frac{3}{40}, \beta_2 = 0.0828,\ \beta^* = \frac{9}{100},\ \sigma_{k1} = 0.85,\ \sigma_{k2} = 1,\ \sigma_{\omega 1} = 0.5, \sigma_{\omega 2} = 0.856.$$

При моделировании движения потока частиц у подстилающей поверхности в такой постановке задачи учитывается влияние гидродинамических сил. При этом сами частицы могут менять турбулентные свойства воздушной среды, что, вероятно, влияет на вертикальный подъем пылевого аэрозоля. Важным неучтенным в данной постановке фактором является электризация частиц и подстилающей поверхности. Поэтому далее будет предложен способ учета влияния электрического поля на движение заряженных частиц.

3. Электростатические эффекты

Поток из N_s заряженных сальтирующих частиц получает отрицательный заряд при взаимодействии с подстилающей поверхностью, которая заряжается положительно. Откалывание от сальтирующей частицы пылинок приводит к появлению над подстилающей поверхностью положительно заряженных сальтирующих частиц и отрицательно заряженных пылинок (рис. 1). Сальтирующие частицы могут быть как положительно, так и отрицательно заряженными. Напряжённость электрического поля может достигать 167 кВ/м на высоте 1.7 см [1], а наибольший удельный заряд для сальтирующих частиц соответствует + 60 мкКл/кг [1].

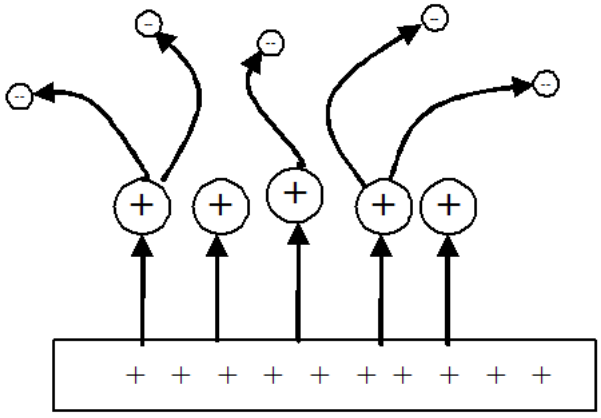


Рис. 1. Положительный заряд на подстилающей поверхности и на крупных частицах.
Fig. 1. Positive charge on the underlying surface and on large particles.

Значения удельных зарядов частиц для всех размеров γ и для крупных частиц γ_+ (табл. 1), которые определены на территории с песчаной подстилающей поверхностью (р. Волга Астраханской области) в [33], имеют меньшие значения (табл. 1.)

Табл. 1. Удельные заряды сальтирующих частиц, измеренных в [33]
Table 1. Specific charges of saltating particles measured in [33]

Временной интервал	Удельные заряды, мкКл/кг	
	γ	γ_+
09:38-13:27	27.0	49.1
14:06-16:00	42.3	52.9
16:26-18:00	28.2	30.7

Предположим, что заряд частиц не столь существенен, чтобы обеспечить взаимовлияние в момент кратких подскоков. Тогда эффект от положительно заряженной поверхности получаем двойной. С одной стороны, частицы над подстилающей поверхностью выталкиваются полем, а, с другой, меняются упругие свойства удара в момент падения.

Электрическое поле создает частичный или полный эффект левитации частицы в воздушной среде. Далее приведены выкладки для учета наличия электрического поля за счет поправки на плотность материала частицы.

Траектория сальтирующей частицы имеет горизонтальную и вертикальную составляющие. Баланс вертикальных составляющих сил определяет высоту подъема частицы. Предположим, что на частицу действует сила Стокса $F_S = 6\pi\mu\rho r_p v_p$, μ - динамическая вязкость среды, v_p - средняя скорость подъема и осаждения частиц, и сила тяжести $F_g = \frac{4}{3}\pi\rho_p r_p^3 g$, где r_p - радиус частицы. Определим ускорение как

$$a = \frac{9\mu v_p}{2\rho_p r_p^2} - g = a_C - g \tag{5}$$

Аналогично при наличии электрического поля в случае, если на поверхности распределен заряд q, возникает электростатическое поле $E = \frac{q}{4\pi\varepsilon_0 h^2}$. Тогда ускорение заряженной сальтирующей частицы, имеющей заряд, приходящийся на единицу массы частицы, q_m:

$$a_e = \pm\frac{qq_m}{4\varepsilon_0 h^2} + a_C - g \tag{6}$$

Влияние на подскоки частиц в гидродинамической модели можно учесть, как относительное изменение плотности частиц, полагая, что под влиянием электрического поля плотность материала частицы оказалась равной ρ_p', получаем

$$a = \pm\frac{9\mu v_p}{2\rho_p' r_p^2} - g = \frac{qq_m}{4\varepsilon_0 h^2} \pm \frac{9\rho v_p}{2\rho_p r_p^2} - g,$$

и может быть внесена такая поправка,

$$\rho_p' = \frac{1}{\frac{qq_m 2r_p^2}{18\mu v_p \varepsilon_0 h^2} + \frac{1}{\rho_p}} \tag{7}$$

Второй возможный эффект при условии заряженной поверхности состоит во влиянии дополнительного ускорения под действием электрического поля на вылет частицы после падения и взаимодействия с подстилающей поверхностью (или с другой частицей).

Электрическое поле будет влиять на коэффициент восстановления скорости частицы после удара. Из результатов туннельных экспериментов для сальтирующих частиц среднее значение коэффициента $\psi = 0{,}69$ [32]. Оценим эффект такого влияния с использованием модели взаимодействия заряженных частиц. Пусть частица массой m движется и в результате удара скорость снижается, тогда изменение импульса

$$\delta p = mv - m\vec{v}_0 = mv_0\left(\frac{v}{v_0} - 1\right) = mv_0\left(\psi - 1\right).$$

Выражая силу через изменение импульса как $F = \frac{\Delta p}{\Delta t}$ и используя (3), получаем, что

$$-v_0\left(\psi'-1\right)=\frac{qq_m}{4\varepsilon_0 h^2}\Delta t-v_0\left(\psi-1\right),$$

откуда

$$-v_0\psi\left(\frac{\psi'}{\psi}-1\right)=\frac{qq_m}{4\varepsilon_0 h^2}\Delta t\,, \tag{8}$$

Из (7), подставляя (8), получаем

$$\frac{qq_m 2r_p^2}{18\mu v_p \varepsilon_0 h^2}=\frac{1}{\rho_p'}-\frac{1}{\rho_p}\,,$$

откуда

$$v_0\psi\left(\frac{\psi'}{\psi}-1\right)\frac{4v_0 r_p^2}{9\mu v_p\Delta t}=\frac{1}{\rho_p}-\frac{1}{\rho_p'}\,.$$

Получаем поправку для коэффициента восстановления скорости:

$$\psi'=\frac{9\mu v_p\Delta t}{4v_0\psi r_p^2}\frac{1}{v_0\rho_p'}\left(\frac{\rho_p'}{\rho_p}-1\right)+\psi \tag{9}$$

Расчеты по формулам (7) и (9) показывают (рис. 2), что увеличение величины напряженности электрического поля и заряда на частице наибольшее влияние оказывает на плотность частицы, коэффициент восстановления скорости почти не меняется.

В связи с этим в следующем разделе влияние электрического поля учитывается при реализации вычислительного эксперимента постпредством поправок для плотности частиц.

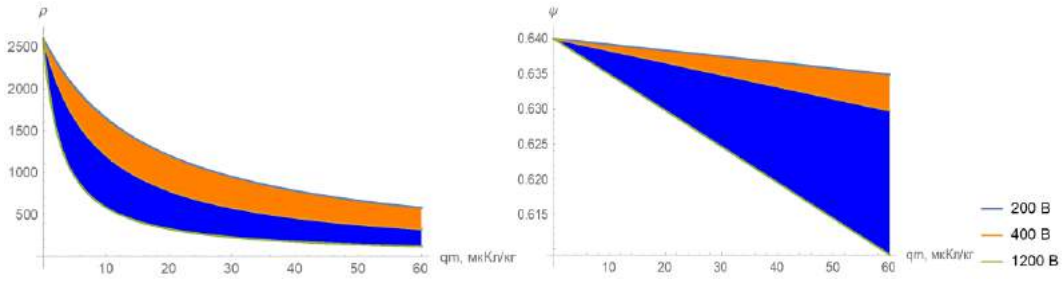


Рис. 2. Изменение плотности частиц в соответствии с (7) (слева) и коэффициента восстановления скорости в соответствии с (9) (справа) при изменении напряженности электрического поля при разных значениях заряда частиц.

Fig. 2. Variation of particle density according to (7) (left) and velocity recovery coefficient according to (9) (right) at change of electric field strength at different values of particle charge.

4. Описание реализации вычислений

Движение воздушного потока с взвешенными в нем и взаимодействующими между собой частицами рассматривалось с использованием Эйлерово-Лагранжевой модели и реализовано средствами открытого пакета OpenFoam [14] (решатель DPMFoam, турбулентная *k*-ω-модель, см. табл. 2)**.** Движение частиц исследовалось над подстилающей поверхностью с двумя препятствиями по типу ряби. Важной характеристикой, которая проявляется в естественных условиях, – неравномерность (квазипериодичность) потока сальтирующих частиц. Это свойство было выявлено и в вычислительном эксперименте с минимальным интервалом

записи данных 0.005 с. Дополнительно оценивались координаты частиц в трех промежуточных моментах времени с учетом средних скоростей. В результате получено число частиц в выделенной области за препятствием в каждый 0.00125 с момент времени. Определены частоты изменения числа частиц в потоке при наличии двух препятствий, которые меняются в зависимости от геометрии поверхности и параметров турбулентной модели.

Как было предложено выше, для учета действия электрического поля на сальтирующие частицы в первом приближении можно учитывать за счет относительного уменьшения плотности движущихся частиц. Предполагаем наличие небольших значений напряженности электрического поля и соответственно зарядов частиц (рис. 2а), приходящихся на единицу массы. Тогда плотность материала частиц будет меняться в экспериментах от известной величины для песчаных частиц 2600 кг/м^3 до 1500 кг/м^3 (см. табл. 1). Это допущение не повлияет на горизонтальный перенос и аэродинамические характеристики, так как размеры частиц остаются неизменными.

Табл. 2. Вычислительные параметры
Table 2. Computational parameters

Решатель	DPMFoam
Вычислительная схема	Gause Linear
Число процессоров	128, 256, 384
Размер ячейки сетки	1.5 мм
Число блоков	12
Число ячеек	31 760

Исследуемая область в вычислительном эксперименте, как и в [14], разделена на два подслоя по высоте с соответствующими граничными условиями (рис. 3):

- Inlet_down и outlet_down (0-0.02 м) со средней скоростью, характерной для этой высоты при логарифмическом профиле скорости ветра;
- Inlet_up и outlet_up (0.02-1 м) с логарифмическим профилем скорости ветра, повторяющем контур поверхности.

На поверхности установлено условие прилипания, верхняя граница свободная.

Область задается шестью вертикальными секциями. На расстоянии 0.6 м от левой границы области (первая секция) располагаются два треугольных элемента, имитирующих структуру эолового рельефа высотой 0.02 м. Последняя секция находится в области за элементом. Определяется влияние движущихся частиц на изменение характеристик воздушного потока в области над элементом и за ним. Расстояние от нижнего края структуры до правой границы равно 2.6 м. Размер ячейки расчетной сетки выбран равным 1.5 мм, так как является оптимальным по времени расчета при наличии устойчивого решения.

Облако частиц генерируется у левой границы области. Число частиц задается, исходя из формулы для общего расхода песка для каждой рассматриваемой динамической скорости [10]:

$$Q = c\sqrt{\frac{d}{D}}\frac{\rho_в}{g}u_*^3 \tag{7}$$

$c = 1.5 - 2.8$, $D = 250 мкм$, $\rho_в = 1.25$ кг/м^3 - плотность воздуха, d – размер частиц, $\rho_p = 2000\frac{кг}{м^3}$ – плотность частиц с учетом неоднородности формы и материала. В соответствии с (4) для пороговой скорости u_*=0.25 м/с рассчитано число частиц,

генерируемых в единицу времени на левой границе до высоты 20 см, которое соответствует 82 частицам.

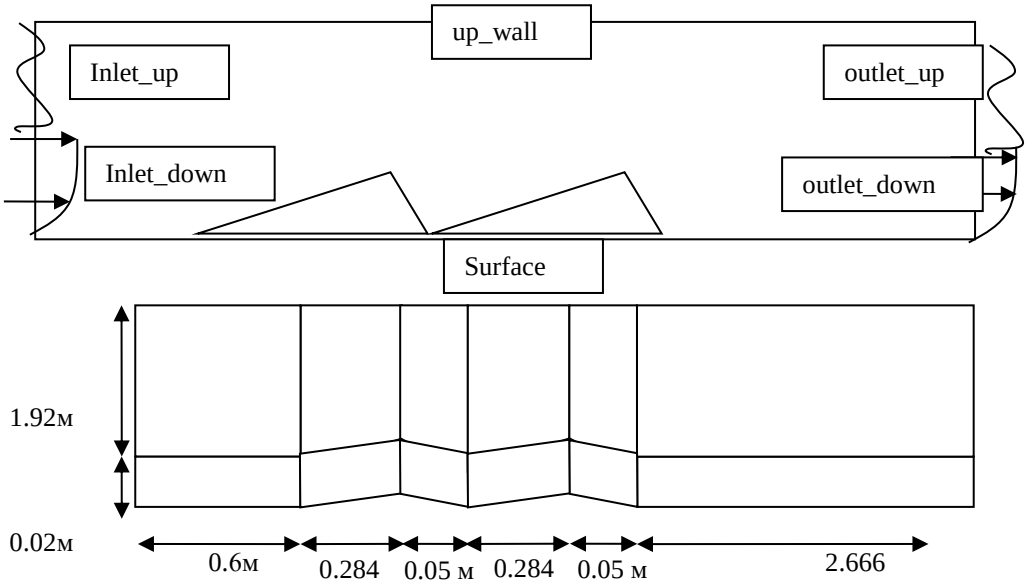


Рис. 3. Схема представления исследуемой области.
Fig. 3. Scheme of representation of the investigated area.

Так как при движении частиц учитывается их взаимодействие друг с другом и с подстилающей поверхностью, в соответствии со справочными данными установлены следующие параметры: $E = 9 \cdot 10^{10}$ Па (9 гПа) – модуль Юнга, $\mu = 0.35$ – коэффициент Пуассона (отношение величин продольной и поперечной деформаций, $\alpha = 0{,}12$ – параметр, связанные с коэффициентом восстановления скорости.

Расчеты выполнялись на вычислительном кластере «РСК Торнадо», созданном и установленным специалистами российской группы компаний РСК. Вычислительный кластер содержит управляющий сервер и 8 вычислительных серверов с двумя процессорами AMD EPYC 7742 64-Core - всего 128 вычислительных ядер на каждом сервере, оперативная память каждого сервера 256 Гб. Вычислительные узлы соединены через коммутатор IB Mellanox 100 гбит/сек с поддержкой протокола MPI, имеется дисковое хранилище данных 411 Тбайт. Для расчетов в рамках данной статьи использовались 1-3 вычислительных сервера.

Для анализа использовались условия при значении динамической скорости u_* =0.25 м/с, близкой к граничному значению, при которой возможен отрыв частиц от поверхности. Для таких значений выявлен эффект квазипериодического изменения числа частиц в потоке [34]. На высоте 5-10 см эта скорость по расчетам для логарифмического профиля близка к 2-2.8 м/с. В момент отрыва воздушного потока от точки на вершине эоловой структуры, происходит усиление ветра. При этом скорость у поверхности *U* может достигать больших значений (5 м/с в таблице 3). Скорости у основания или на склоне дюны имеют меньшие значения (2-2.8 м/с в таблице 3). Для этих двух случаев вычисляются значения турбулентной энергии $k = \frac{3}{2}\left(UI\right)^2$, где *I* – интенсивность турбулентности (*I*=0.1), величина удельной

диссипации турбулентной энергии рассчитывалась как $\omega = C_{\mu}^{\frac{3}{4}} \frac{k^{\frac{1}{2}}}{l}$, $C_{\mu} = 0.013$, l – масштаб.

Табл.3. Схема изменения параметров в вычислительном эксперименте
Table.3. Scheme of parameter changes in the computational experiment

Тип параметров среды	q_m, мкКл/кг	ρ'_p, кг/м^3	l, м	U, м/с	k, м^2·с$^{-2}$	ω, с$^{-1}$	u_*, м/с
1	0	2600	2	5	0.38	55	0.25
	+2	2000					
	+4	1800					
	+6	1500					
2	0	2600	2	2.8	0.14	44	0.25
	+2	2000					
	+4	1800					
	+6	1500					

В следующем разделе приведены результаты тестирования переложенных поправок для вычислительной модели.

5. Результаты моделирования

Различные запуски вычислительного эксперимента указывают на соответствие поведения потока частиц реальным данным [35]. Высота подскоков частиц достигает 10 см, скорости движения частиц достигают 5 м/с (рис. 4). Профили скорости воздушного потока (рис. 5) отклоняются под влиянием движущихся частиц от логарифмического, как это показано в [36]. Частоты вариации числа частиц в потоке дают для спектральных функций всплески при значениях 7-12 и 160-190 Гц (рис. 6), близкие к наблюдаемым в экспериментах характерным частотам [34, 37].

Для четырех экспериментов с различными плотностями материала частиц (2600 – исходная плотность, 2000, 1800 и 1500 кг/м^3) при различных значениях скорости воздушного потока у поверхности выявлено, что при уменьшении плотности частиц (изменении электрического поля и заряда частицы) происходят изменения следующих параметров:

- дольше сохраняется возмущающее воздействие на среду (на более дальнем расстоянии за препятствием частицы влияют на локальное изменение турбулентной энергии) (рис. 7);
- уменьшается (для случая k=0.38 м^{2}с$^{-2}$) и увеличивается (для случая k=0.14 м^{2}с$^{-2}$) значение дисперсии скоростей движения сальтирующих частиц (рис. 4 a, b);
- увеличивается разброс высоты подскоков частиц (рис .4 c, d)
- уменьшается нижнее (для случая k=0.38 м^{2}с$^{-2}$) и увеличивается верхнее (для случая k=0.14 м^{2}с$^{-2}$) значение характерных частот (рис. 6) изменения числа частиц в потоке (увеличивается период между двумя зонами максимального воздействия движущегося потока частиц на воздушную среду).

6. Заключение

Движение под влиянием ветра песчаных частиц в результате сальтаций (подскоки) до высоты нескольких сантиметров над подстилающей поверхностью в условиях незакрепленных песков исследовано в вычислительном эксперименте средствами открытого пакета

OpenFOAM. Предложено учитывать наличие электрического заряда на частицах и поверхности с использованием параметрической модели.

Оценено влияние на динамику движения частиц в гидродинамическом потоке. В статье предлагается параметрическая модель, позволяющая численном эксперименте учитывать влияние заряда частиц и наличие электрического поля.

Так как важной характеристикой для потока сальтирующих частиц в естественных условиях является его квазипериодичность, движение рассмотрено при наличии двух препятствий, подобных ряби.

Движение частицы с удельным зарядом 10мкКл/кг в поле 400 В можно рассматривать с заменой плотности ее материала на 1800 кг/м^3.

Движение воздушного потока с взвешенными в нем и взаимодействующими между собой частицами исследовалось с использованием Эйлерово-Лагранжевой модели с учетом соображений о влиянии зарядов на частицах и поверхности.

При учете влияния электростатических эффектов выявлено усиление возмущающего воздействия на среду частиц, вылетающих после препятствий. В зависимости от значения турбулентной энергии уменьшается или увеличивается дисперсия скоростей движения сальтирующих частиц, увеличивается разброс высоты подскоков, что подтверждается известными экспериментами. В зависимости от значения турбулентной энергии уменьшается нижнее или увеличивается верхнее значение характерных частот изменения числа частиц в потоке.

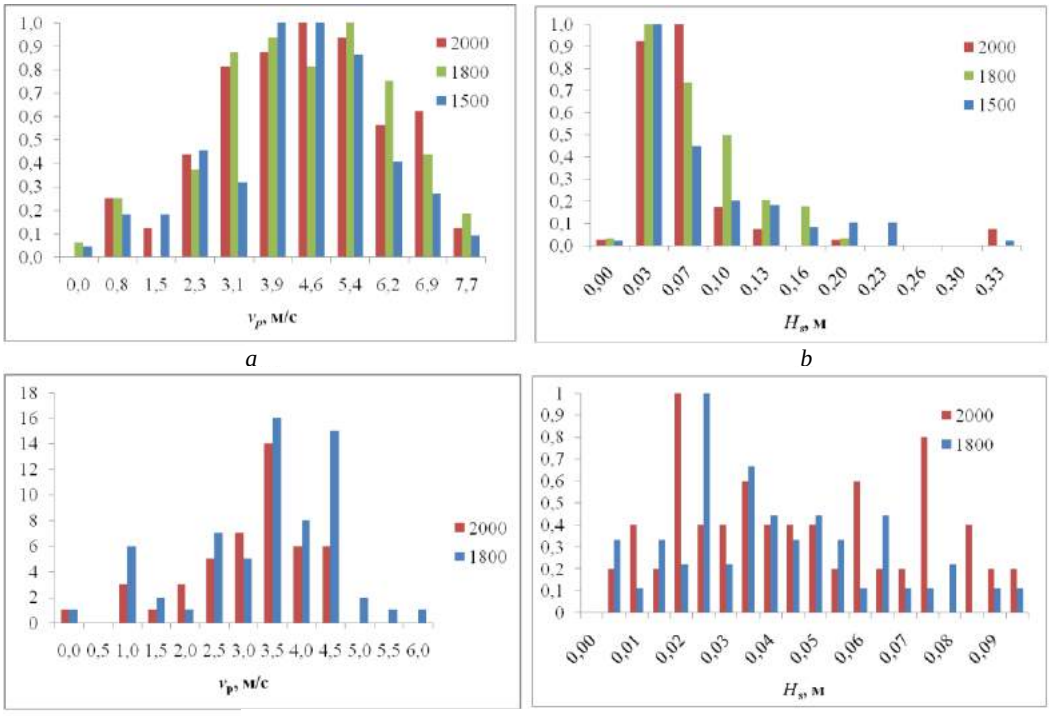


Рис. 4. Плотности вероятности для скоростей движения частиц (a – для типа параметров среды 1, b – для типа параметров среды 2) и высоты расположения (c – для типа параметров среды 1, d – для типа параметров среды 2).
Fig. 4. Probability densities for particle velocities (a – for type of medium parameters 1, b – for type of medium parameters 2) and height of location (c – for type of medium parameters 1, d – for type of medium parameters 2).

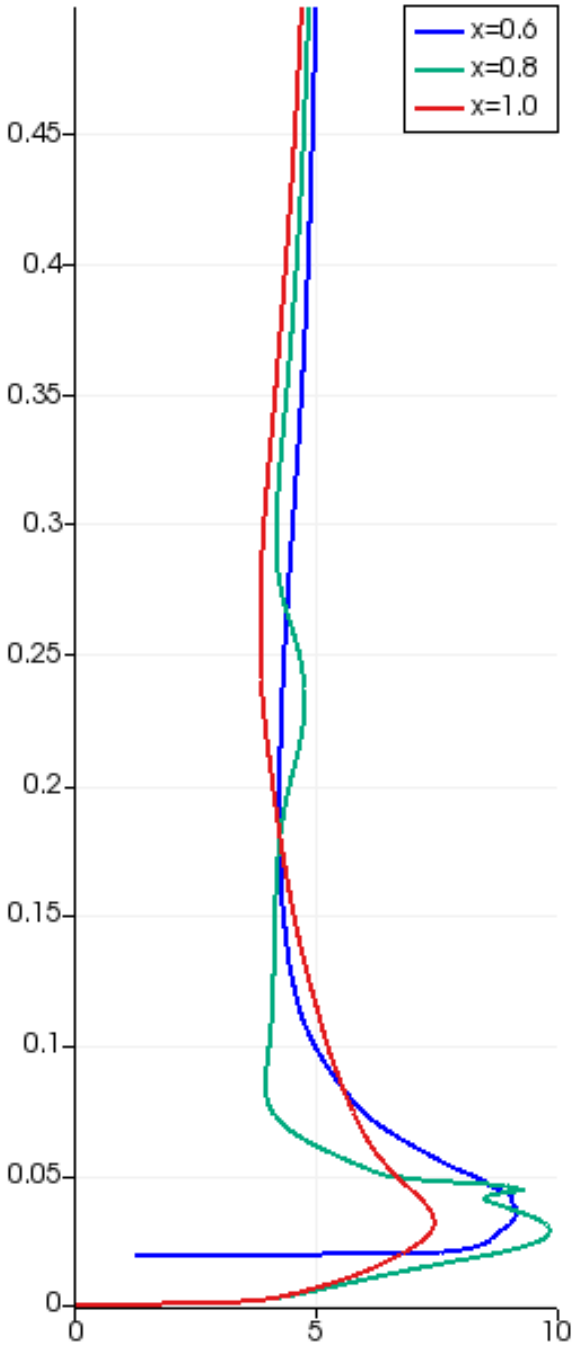


Рис. 5. Характерный профиль скорости воздушного потока под влиянием частиц у поверхности до высоты 0.5 м.
Fig. 5. Characteristic profile of air velocity under the influence of particles near the surface up to a height of 0.5 m.

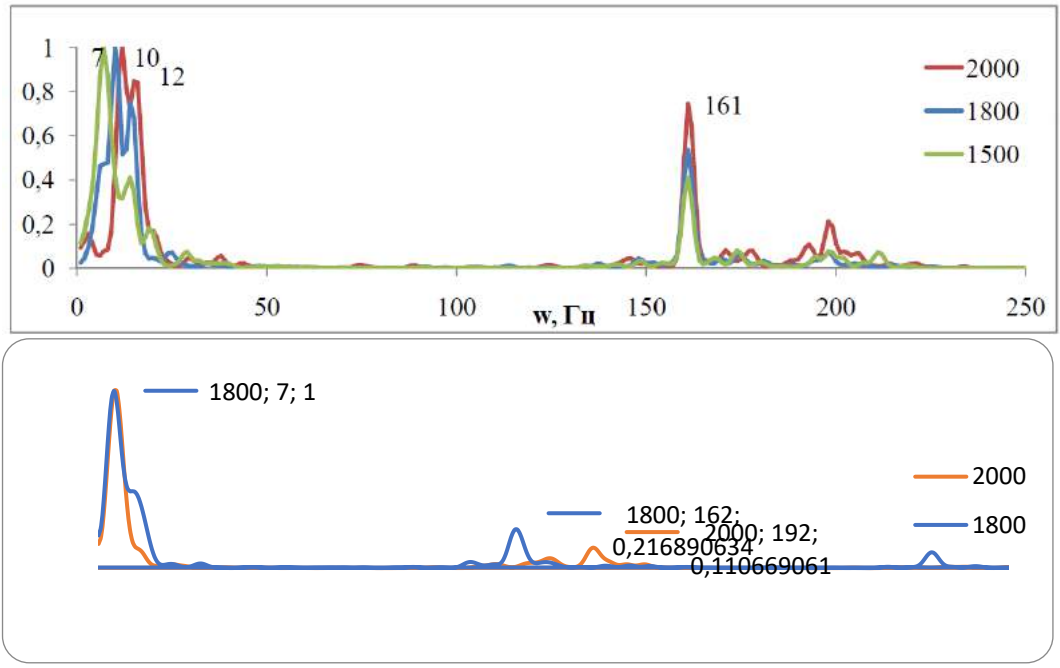


Рис. 6. Спектральные плотности для трех экспериментов с различной плотностью частиц.
Fig. 6. Spectral density functions for three experiments with different particle densities.

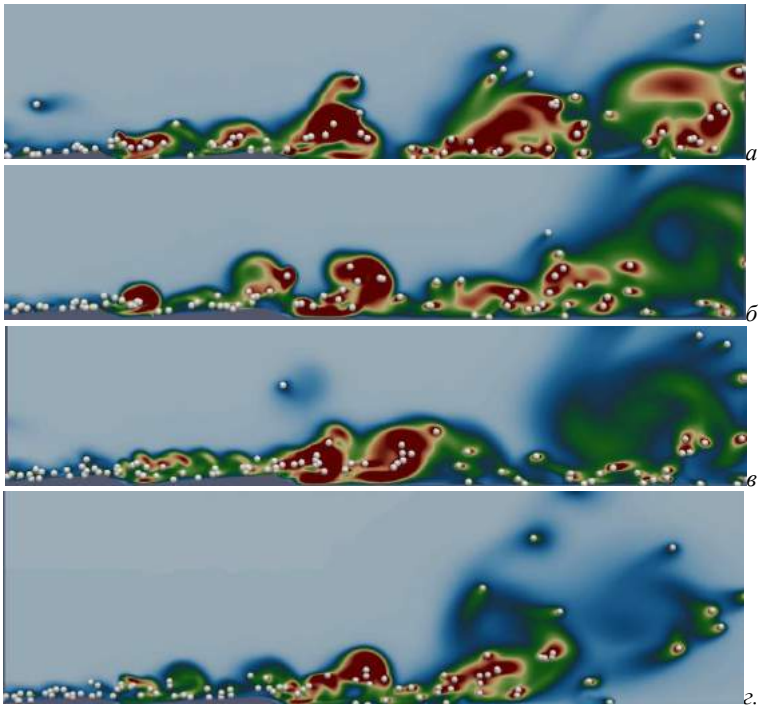


Рис. 7. Движение частиц в потоке над подстилающей поверхностью для плотностей частиц а – 1500, б – 1800, в – 2000, г -2600 кг/м^3.
Fig.7. Movement of particles in the flow above the underlying surface for particle densities a - 1500, b - 1800, c - 2000, d -2600 kg/m3

Список литературы

[1] Schmidt D.S., Schmidt R.A., Dent J.D. Electrostatic force on saltating sand. Journal of Geophysical Research: Atmospheres, 103(D8), 1998. p. 8997-9001.

[2] Kok J.F., Lacks D.J. Electrification of granular systems of identical insulators. Physical Review E., V. 79(5), 1998, pp. 051304.

[3] Бютнер Э.К. Динамика приповерхностного слоя воздуха. Л.: Гидрометиздат, 1978. с. 156.

[4] Anderson R. S., Hallet B. Sediment transport by wind: toward a general model. Geological Society of America Bulletin, 97(5), 1986, pp. 523-535.

[5] Dey S., Ali S. Z. Advances in modeling of bed particle entrainment sheared by turbulent flow. Physics of Fluids, 30(6), 2018, pp. 061301.

[6] Huang G. et al. Large-Eddy Simulation of Erosion and Deposition over Multiple Two-Dimensional Gaussian Hills in a Turbulent Boundary Layer. Boundary-Layer Meteorology, 2019, pp. 1-30.

[7] Малиновская Е.А. Модель отрыва песчаной частицы ветром. Известия РАН. Физика атмосферы и океана, 2017, 53(5), с.588-596.

[8] Семенов О.Е. Введение в экспериментальную метеорологию и климатологию песчаных бурь. Алматы, 2011, 580 с.

[9] Чхетиани О. Г. , Калашник М. В., Ингель Л. Х. Генерация “теплового ветра” над неоднородно нагретой волнистой поверхностью. Известия РАН. Физика атмосферы и океана, 49(2), 2013, с. 137–143

[10] Shao Y. Physics and modeling of wind erosion. Springer Science & Business Media, 2008, 452 p.

[11] Rasmussen K. R., Kok J. F., Merrison J. P. Enhancement in wind-driven sand transport by electric fields. Planetary and Space Science, 57(7), 2009, pp. 804-808.

[12] Esposito F. et al. The role of the atmospheric electric field in the dust-lifting process. Geophysical Research Letters, 43(10), 2016, pp. 5501-5508

[13] Малиновская Е. А. и др. О связи приземного электрического поля и аридного аэрозоля при различных ветровых условиях. Доклады Российской академии наук. Науки о Земле, 502(2), 2022, с. 115-124.

[14] Malinovskaya E.A., Gorchakov G.I., Karpov A.V., Maksimenkov L.O., Datsenko O.I.On the conditions of the emergence of a periodic mode of saltating flow. Известия РАН. Физика атмосферы и океана, 2023 (in press).

[15] Tong D., Huang N. Numerical simulation of saltating particles in atmospheric boundary layer over flat bed and sand ripples. Journal of Geophysical Research: Atmospheres, 117(16), 2012

[16] Huang G. et al. Large-eddy simulation of erosion and deposition over multiple two-dimensional gaussian hills in a turbulent boundary layer. Boundary-Layer Meteorology, 173, 2019, pp. 193-222.

[17] Gu Z. et al. Numerical simulation of dust lifting within dust devils—Simulation of an intense vortex. Journal of the atmospheric sciences, 63(10), 2006, pp. 2630-2641.

[18] *Дерябина М.С., Мартынов С.И.* Моделирование течения вязкой жидкости с частицами через ячейки пористой среды. Вычисл. мех. сплош. сред, 9(4), 2016, pp. 420-429. https://doi.org/10.7242/1999-6691/2016.9.4.35

[19] *Мартынов С.И., Ткач Л.Ю.* Динамика цепочечных агрегатов частиц в потоке вязкой жидкости. Ж. вычисл. матем. и матем. физ., 56(5), 2016, с. 840-855. https://doi.org/10.7868/S004446691605015X

[20] *Dupont S., Bergametti G., Simoëns S.* Modeling aeolian erosion in presence of vegetation. J. Geophys. Res. Earth Surface, 119, 2014, pp. 168-187.

[21] *Araújo A.D., Parteli E.J.R., Pöschel T., Andrade J.S., Herrmann H.J.* Numerical modeling of the wind flow over a transverse dune. Scientific reports. 3, 2013, pp. 2858.

[22] *Dey S., Ali S.Z.* Advances in modeling of bed particle entrainment sheared by turbulent flow. Phys. Fluid, 30, 2018, pp. 061301.

[23] *Siminovich A., Elperin T., Katra I., Kok J.F., Sullivan R., Silvestro S., Yizhaq H.* Numerical study of shear stress distribution over sand ripples under terrestrial and Martian conditions. J. Geophys. Res. Planets, 124, 2019, pp. 175-185.

[24] 10The OpenFOAM® Foundation. http://www.openfoam.org/index.php

[25] *Michelsen B., Strobl S., Parteli E.J.R., Pöschel T.* Two-dimensional airflow modeling underpredicts the wind velocity over dunes. Scientific reports, 5. 2015, pp 16572. https://doi.org/10.1038/srep16572

[26] *Ali M.S.M., Salim S.A.Z.S., Ismail M.H., Muhamad S., Mahzan M.I.* Aeolian tones radiated from flow over bluff bodies // Open Mech. Eng. J., 7, 2013, pp. 48-57. https://doi.org/10.2174/1874155X01307010048

[27] *Gu.J.* Numerical modeling of the wind flow over a transverse dune. Scientific reports, 3. 2013, pp.2858.
[28] Malinovskaya E. A., Chkhetiani O. G. On Conditions for the Wind Removal of Soil Particles. Journal of Applied Mechanics and Technical Physics, 62(7), 2021, pp. 1117-1131.
[29] Malinovskaya E.A. Simulation of the flow around 3D surfaces in the study of changes in aeolian relief forms. International Young Scientists School and Conference on Computational Information Technologies for Environmental Sciences CITES '2019, 2019, p.192-195.
[30] Malinovskaya E. A., Chkhetiani O. G. Modeling of near-surface flows over an aeolian relief. IOP Conference Series: Earth and Environmental Science. IOP Publishing, 386(1), 2019, pp. 012030.
[31] Попов В.Л. Механика контактного взаимодействия и физика трения. От нанотрибологии до динамики землетрясений. Москва: Издательство Физматлит, 2013, с.213
[32] Tsuji Y., Tanaka T., Ishida T. Lagrangian numerical simulation of plug flow of cohesionless particles in a horizontal pipe. Powder technology, 71(3), 1992, 239-250.
[33] Горчаков Г. И. и др. Удельный заряд сальтирующих песчинок на опустыненных территориях. Доклады Академии наук, 456(4), 2014, с. 476-476.
[34] Gorchakov G. I. et al. Quasiperiodic saltation in the windsand flux over desertified areas. Atmospheric and oceanic optics, 29, 2016, 501-506.
[35] Anderson R. S., Hallet B. Sediment transport by wind: toward a general model. Geological Society of America Bulletin. 97(5), 1986, С. 523-535.
[36] Almeida M. P., Andrade Jr J. S., Herrmann H. J. Aeolian transport layer. Physical review letters. 2006, 96(1), 018001.
[37] Malinovskaya E.A., Gorchakova G.I., Karpov A. V., Maksimenkov L.O., Datsenko O.I. On the Conditions of the Emergence of a Periodic Mode of Saltating Flow. Izvestiya, Atmospheric and Oceanic Physics, 2023, 59(6),749–759.

Информация об авторах / Information about authors

Елена Александровна МАЛИНОВСКАЯ – кандидат физико-математических наук, старший научный сотрудник лаборатории геофизической гидродинамики Института физики атмосферы им. А.М. Обухова РАН с 2018 г. Сфера научных интересов: исследование динамических и обменных процессов на границе раздела атмосфера – подстилающая поверхность, математическое и численное моделирование ветрового и вихревого выноса минеральных аэрозолей.

Elena Aleksandrovna MALINOVSKAYA – Cand. Sci. (Phys.-Math.), researcher at the Laboratory of Geophysical Hydrodynamics at the A.M. Obukhov Institute of Atmospheric Physics, Russian Academy of Sciences since 2018. Research interests: study of dynamic and exchange processes at the interface between the atmosphere and the underlying surface, mathematical and numerical modeling of the wind removal of mineral aerosols.

Геннадий Ильич ГОРЧАКОВ – доктор физико-математических наук, профессор, заведующий лабораторией Оптики и микрофизики аэрозоля с 1985 года. Сфера научных интересов: физика атмосферы, оптика атмосферы, атмосферный аэрозоль, атмосферное электричество, атмосферная экология.

Gennady Ilyich GORCHAKOV – Dr. Sci. (Phys.-Math.), Professor, Head of the Laboratory of Optics and Aerosol Microphysics since 1985. Area of scientific interests: atmospheric physics, atmospheric optics, atmospheric aerosol, atmospheric electricity, atmospheric ecology.

Алексей Владимирович КАРПОВ – старший научный сотрудник с 2017 года. Сфера научных интересов: атмосферный аэрозоль, малые газовые примеси, радиационные эффекты аэрозоля, генерация аэрозоля на опустыненных территориях.

Alexey Vladimirovich KARPOV – senior researcher since 2017. Area of scientific interests: atmospheric aerosol, trace gases, radiation effects of aerosol, aerosol generation in deserted areas.

Леонид Олегович МАКСИМЕНКОВ – научный сотрудник лаборатории моделирования атмосферного переноса ИФА РАН с 2021 года. Сфера научных интересов: Математическое моделирование, численные методы, программная обработка данных.

Leonid Olegovich MAKSIMENKOV – researcher in the laboratory of atmospheric transfer modelling IAP RAS from 2021. Research interests: Mathematical modelling, numerical methods, software data processing.

Олег Игоревич ДАЦЕНКО – младший научный сотрудник с 2022 года. Сфера научных интересов: ветропесчаный поток, естественные и антропогенные аэрозоли, загрязнение атмосферы, пространственно-временная изменчивость аэрозоля.

Oleg Igorevich DATSENKO - junior researcher since 2022. Area of scientific interests: wind-sand flux, natural and anthropogenic aerosols, atmospheric pollution, spatiotemporal variability.

DOI: 10.15514/ISPRAS-2023-35(5)-19

Использование метода декомпозиции области для распараллеливания моделирования течения вязкой несжимаемой среды методом LS-STAG и дополнительного предобуславливания

[1] *И.К. Марчевский,* ORCID: *0000-0003-4899-4828 <iliamarchevsky@mail.ru>*
[2] *В.В. Пузикова,* ORCID: 0000-0003-0712-4519 *<v.puzikova@yadro.com>*

[1] *Московский государственный технический университет им. Н.Э. Баумана, 105005, Россия, г. Москва, ул. 2-я Бауманская, д. 5.*
[2] ООО *YADRO, 123022, Россия, г. Москва, ул. Рочдельская, д. 15с15.*

Аннотация. В ходе численного решения задач механики сплошной среды основная часть вычислительных затрат, как правило, приходится на решение больших разреженных систем линейных алгебраических уравнений. По этой причине эффективное распараллеливание именно этой процедуры может значительно ускорить моделирование. Наиболее простой подход к решению этой задачи, заключающийся в распараллеливании матрично-векторных операций в обычном итерационном решателе, требует нескольких точек синхронизации и обменов коэффициентами на каждой итерации метода, что не позволяет значительно ускорить расчет в целом. Поэтому предпочтительнее оказываются методы декомпозиции области, которые подразумевают разбиение расчетной области на подобласти, построение и решение отдельных задач в них, а также некоторую процедуру согласования решения между подобластями для обеспечения глобальной сходимости. Подобласти могут перекрываться, как в методе Шварца, использующемся в OpenFOAM, или разделяться интерфейсными участками, для которых решается своя собственная интерфейсная задача, как в методе дополнения Шура. Последний метод используется в данной работе для построения параллельного алгоритма моделирования течений вязкой несжимаемой среды методом погруженных границ LS-STAG. Полученная матрица интерфейсной системы имеет блочную трехдиагональную структуру. Для ускорения прототипирования в программной реализации разработанного алгоритма использована технология параллельного программирования OpenMP, поэтому вычислительные эксперименты проводятся только на системах с общей памятью, в частности на отдельных узлах учебно-экспериментального кластера кафедры «Прикладная математика» МГТУ им. Н. Э. Баумана. Для верификации и оценки эффективности разработанного алгоритма рассмотрена хорошо исследованная тестовая задача о моделировании плоского обтекания неподвижного кругового профиля. Расчеты на последовательности сеток при их разделении на разное количество подобластей показывают, что параллельный алгоритм сходится к тому же решению, что и исходный алгоритм, а рассчитанные значения числа Струхаля и коэффициента лобового сопротивления хорошо согласуются с известными в литературе экспериментальными и расчетными данными. Эксперименты демонстрируют, что разработанный алгоритм с декомпозицией области позволяет ускорить моделирование даже в последовательном режиме за счет уменьшения количества итераций, то есть метод декомпозиции области действует как дополнительный предобуславливатель. Благодаря этому свойству при расчетах в параллельном режиме ускорение оказывается сверхлинейным до некоторого числа подобластей, зависящего от размера задачи.

Ключевые слова: метод декомпозиции области; метод погруженных границ; метод LS-STAG; вязкая несжимаемая среда; система линейных алгебраических уравнений; предобуславливание; метод FGMRES; параллельные вычисления; OpenMP; OpenFOAM.

Для цитирования: Марчевский И.К., Пузикова В.В. Использование метода декомпозиции области для распараллеливания моделирования течения вязкой несжимаемой среды методом LS-STAG и дополнительного предобуславливания. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 287–302. DOI: 10.15514/ISPRAS–2023–35(5)–19.

Domain Decomposition Method Usage for Parallelization and Extra Preconditioning of Viscous Incompressible Flow Simulation by Using the LS-STAG Method

[1] *I. K. Marchevsky*, ORCID: *0000-0003-4899-4828 <iliamarchevsky@mail.ru>*
[2] *V.V. Puzikova*, ORCID: 0000-0003-0712-4519 *<v.puzikova@yadro.com>*

[1] *Bauman Moscow State Technical University,*
5, 2nd Baumanskaya st., Moscow, 105005, Russia.
[2] *YADRO,*
15 bld.15, Rochdelskaya st., Moscow, 123022, Russia.

Abstract. As a rule, the main part of the computational costs in the numerical solution of problems in continuum mechanics consists in the solving of large sparse systems of linear algebraic equations. For this reason, efficient parallelization of this particular procedure can significantly speed up the simulation. To solve this problem, two main approaches can be used. The simplest approach consists in parallelizing of matrix-vector operations in a usual iterative solver. It requires several synchronization points and exchanges of coefficients at each iteration of the solver, which does not significantly speed up the simulation process as a whole. Therefore, domain decomposition methods are preferable. These methods involve dividing the computational domain into subdomains, constructing and solving separate problems in them, as well as some procedure to coordinate the solution between subdomains to ensure global convergence. Subdomains can overlap, as in the Schwartz method used in OpenFOAM, or they can be separated by interface sections, on which their own interface task is built, as in the Schur complement method. The latter method is used in this research to construct a parallel algorithm for viscous incompressible flow simulation by using the immersed boundary method LS-STAG with cut-cells and level-set functions. The resulting matrix of the interface system has a block tridiagonal structure. To speed up prototyping, OpenMP parallel programming technology is used in the software implementation of the developed algorithm, so computational experiments are carried out only on systems with shared memory, in particular on individual nodes of the educational and experimental cluster of the Applied Mathematics Department, Bauman Moscow State Technical University. To verify and evaluate the effectiveness of the developed parallel algorithm, a well-studied test problem about simulation of two-dimensional flow around a stationary circular airfoil is considered. Computations on a sequence of meshes with different numbers of subdomains show that the parallel algorithm allows one to obtain the same numerical solution as the original algorithm of the LS-STAG method, and the computed values of the Strouhal number and drag coefficient are in good agreement with the experimental and computational data known in the literature. Experiments demonstrate that the developed algorithm with domain decomposition allows to accelerate simulation even in sequential mode by reducing the number of solver iterations, i.e. the domain decomposition method acts as an additional preconditioner. Due to this property, the acceleration is superlinear when simulating in parallel mode with developed algorithm. This effect persists up to a certain number of subdomains, which depends on the size of the problem.

Keywords: domain decomposition method; immersed boundary method; the LS-STAG method; viscous incompressible flow; system of linear equations; preconditioner; FGMRES method; parallel computations; OpenMP; OpenFOAM.

For citation: Marchevsky I.K., Puzikova V.V. Domain Decomposition Method Usage for Parallelization and Extra Preconditioning of Viscous Incompressible Flow Simulation by Using the LS-STAG Method. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 5, 2023. pp. 287-302 (in Russian). DOI: 10.15514/ISPRAS-2023-35(5)-19.

1. Введение

В ходе численного решения задач механики сплошной среды вычислительные ресурсы могут стать ограничивающим фактором при использовании больших и сложных моделей. Распараллеливание решения разреженных систем линейных алгебраических уравнений (СЛАУ), на долю которого, как правило, приходится основная часть вычислительных затрат при проведении моделирования, является одним из основных способов ускорения таких расчетов. Кроме того, использование параллельных методов решения СЛАУ становится обязательным при решении задач на распределенных системах.

Существует несколько подходов к параллельному решению СЛАУ. Во-первых, может использоваться обычная итерационная процедура (например, такие крыловские методы [1] как CG, BiCG, BiCGStab и т.д.) с распределенными вычислениями матрично-векторных произведений или операций над независимыми частями векторов и межпроцессорными обменами в точках вычислениях необходимых коэффициентов алгоритма с использованием скалярных произведений [2]. В этом случае, каждый процессор работает только со своим участком расчетной области, но при этом на каждой итерации решения СЛАУ требуется делать несколько точек синхронизации и обменов коэффициентами. С точки зрения реализации этот подход наиболее прост, но требует много коммуникационных затрат. Во-вторых, могут использоваться методы декомпозиции области [3], которые подразумевают решение на каждом процессоре отдельных специальным образом построенных задач в подобластях и некоторую межпроцессорную процедуру согласования решения между этими подобластями с целью обеспечения глобальной сходимости.

Методы декомпозиции области можно разделить на две группы [1]: подходы, основанные на методе дополнения Шура, и подходы, основанные на методе Шварца. В первом случае область разделяется на подобласти без перекрытий, в которых формируются локальные СЛАУ, и интерфейсные границы между ними, для которых составляется собственная СЛАУ. Решение последней позволяет вычислить значения неизвестных на интерфейсных границах и использовать их как условия первого рода для задач в подобластях. Во втором случае (для методов Шварца) область разделяется на подобласти с перекрытиями, решения пересылаются между процессорами, обрабатывающими соседние подобласти. Этот подход требует больше коммуникационных затрат, но является более робастным. Похожий подход используется в пакете OpenFOAM [4] для проведения гидродинамических расчетов с использованием MPI, декомпозиция области при этом строится при помощи таких библиотек, как Scotch [5] и METIS [6], а решение задач в подобластях осуществляется при помощи обычных решателей СЛАУ.

Целью данной работы является разработка, реализация и верификация параллельного алгоритма для моделирования течений вязкой несжимаемой среды методом погруженных границ [7] с усеченными ячейками и функциями уровня [8] LS-STAG [9]. Этот метод развивается с 2010 г. и еще не реализован в крупных библиотеках для решения задач вычислительной гидродинамики, хотя обладает рядом привлекательных особенностей:

- использование прямоугольных разнесенных сеток, не связанных с границами обтекаемых тел, в т.ч. в случае их движения;
- сохранение пятиточечного шаблона дискретизации в 2D и семиточечного в 3D [10];
- корректная аппроксимация уравнений и граничных условий на усеченных ячейках;
- наличие модификаций для моделирования турбулентных течений с использованием подходов RANS/LES/DES [11];
- наличие модификаций для моделирования течений неньютоновских жидкостей [12].

2. Постановка тестовой задачи

Для верификации разработанного алгоритма используется хорошо изученная тестовая задача о моделировании двумерного обтекания неподвижного кругового профиля. Постановка задачи включает в себя уравнение неразрывности и уравнения Навье-Стокса, а также необходимые начальные и граничные условия. Задача представляется в безразмерном виде; в качестве характерной скорости используется скорость набегающего потока, а в качестве характерного размера – диаметр обтекаемого кругового профиля.

Рассматривается расчетная область, представленная на рис. 1. На левой границе расчетной области для скорости стоит граничное условие первого рода – набегающий поток со скоростью $V_\infty = 1$, на остальных границах – стандартное для практики условие выхода потока (равенство нулю нормальной производной скорости).

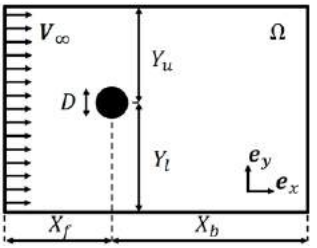


Рис. 1. Расчетная область.
Fig. 1. Computational domain.

В данную расчетную область помещен неподвижный круговой профиль диаметром $D = 1$. Центр профиля находится на расстоянии $X_f = 8$ от левой границы расчетной области и $X_b = 15$ от правой. Верхняя и нижняя границы расчетной области равноотстоят от центра профиля на расстояние $Y_u = Y_l = 12$.

3. Основные идеи метода LS-STAG

В двумерной расчетной области вводится прямоугольная сетка, в общем случае неравномерная, называемая «основной». Относительно нее строятся еще две сетки: «x-сетка», смещенная относительно основной на половину ячейки по горизонтали, и «y-сетка», смещенная относительно основной на половину ячейки по вертикали.

В центрах ячеек основной сетки вычисляются давление и нормальные напряжения, в углах – касательные напряжения. В серединах жидких частей граней ячеек – компоненты скорости, причем горизонтальная составляющая скорости вычисляется в середине восточной грани, а вертикальная – в середине северной. Таким образом, контрольные объемы для построения дискретного аналога уравнения неразрывности совпадают с ячейками основной сетки, ячейки x-сетки являются контрольными объемами для построения дискретного аналога уравнения импульса в проекции на ось Ox, а ячейки y-сетки – контрольными объемами для построения дискретного аналога уравнения импульса в проекции на ось Oy [9].

Для описания положения погруженной границы вводится знакопеременная функция уровня [8], ее значения вычисляются в углах ячеек основной сетки. Функция уровня равна нулю на границе профиля, отрицательна снаружи профиля и положительна внутри него. По четырем значениям функции уровня для каждой ячейки можно вычислить два коэффициента заполнения ячейки – доли восточной и северной грани ячейки, заполненные жидкостью. Эти коэффициенты позволяют однозначно определить тип усеченной ячейки: в 2D случае возможны трапециевидные, треугольные и пятиугольные ячейки; названия отражают форму твердой части ячейки. Коэффициенты заполнения ячеек активно используются при построении дискретных аналогов определяющих соотношений и позволяют добиться

единообразной формы из записи на усеченных ячейках и ячейках, полностью заполненных жидкостью.

Для интегрирования по времени в случае неподвижных погруженных границ используется схема предиктор-корректор второго порядка точности. Шаг предиктора приводит к решению разностного аналога уравнения Гельмгольца для прогноза скоростей, а шаг корректора – к решению разностного аналога уравнения Пуассона для поправки давления. Затем определяются новые значения скоростей и давления.

Таким образом, на каждом шаге по времени решается три СЛАУ:

- для горизонтальных составляющих скоростей на x-сетке;
- для вертикальных составляющих скоростей на y-сетке;
- для функции давления на основной сетке.

Отметим, что третья СЛАУ является плохообусловленной, поскольку возникает при построении дискретного аналога уравнения Пуассона. По этой причине время счета может сильно различаться в зависимости от выбора предобуславливателя.

4. Распараллеливание алгоритма

В работе [10] имеется упоминание о распараллеливании метода LS-STAG, однако никаких деталей алгоритма не представлено. Кроме того, в указанной работе не было вополнено никакого исследования эффективности параллельного алгоритма: рассматривалась только точность решения при проведении расчетов с использованием двух и четырех потоков.

Других работ по построению параллельных алгоритмов для методов усеченных ячеек, в которых построение дискретных аналогов и решение уравнений происходят в том числе и на погруженной границе (на усеченных ячейках), найти не удалось. Однако для классических методов погруженных границ, в которых решение на погруженной границе лишь интерполируется, известен, например, алгоритм [13], основанный на расщеплении системы уравнений по направлениям, сведении к трехдиагональным СЛАУ и их параллельном решении с помощью дополнения Шура. Кроме того, для этих классических методов погруженных границ также известна работа [14], в которой использован алгоритм Эйткена-Шварца.

В данной работе в качестве метода декомпозиции области рассматривается метод дополнения Шура [1, 3]. Он позволяет заменять решение задачи в исходной расчетной области на решение серии независимых задач в подобластях, которые не имеют перекрытий. Интерфейсные условия (граничные условия на границах между подобластями) при этом учитываются неявно. На каждом шаге по времени при использовании данного метода решаются по две задачи для каждой подобласти и одна задача для вычисления неизвестных, соответствующих интерфейсам между соседними подобластями.

4.1 Декомпозиция области

Рассмотрим СЛАУ

$$Ax = b, \quad det\, A \neq 0 \tag{1}$$

Значения компонент вектора неизвестных x вычисляются в центрах ячеек введенной в расчетной области Ω сетки. Разделим расчетную область на непересекающиеся подобласти, разделенные интерфейсными рядами контрольных объемов:

$$\Omega = \cup_{p=0}^{P-1}\Omega_p \cup_{i=0}^{P-2}\Gamma_i.$$

Здесь Ω_p – непересекающиеся подобласти, $p = \overline{0, P-1}$; Γ_i – интерфейсные ряды между подобластями, $i = \overline{0, P-2}$. Обозначим число контрольных объемов в интерфейсном ряду как N. Предположим для определенности, что подобласти представляют собой горизонтальные

полосы. Для случая разбиения расчетной области на вертикальные полосы выкладки могут быть проведены аналогично.

В результате такого разбиения вектор неизвестных также разбивается на непересекающиеся подвекторы – векторы неизвестных из подобластей и вектор неизвестных со всех интерфейсных границ. Введем следующие обозначения: x_p – вектор неизвестных из p-й подобласти, $p = \overline{0, P-1}$; x_s – вектор неизвестных со всех интерфейсных границ, пронумерованных подряд слева направо снизу вверх. Теперь перенумеруем контрольные объемы еще раз – сначала во всех подобластях подряд сверху вниз, потом на всех интерфейсных рядах. После такой перенумерации СЛАУ (1) примет следующий блочный вид:

$$\begin{bmatrix} A_{0,0} & \Theta & \cdots & \Theta & A_{0,s} \\ \Theta & A_{1,1} & \cdots & \Theta & A_{1,s} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Theta & \Theta & \cdots & A_{P-1,P-1} & A_{P-1,s} \\ A_{s,0} & A_{s,1} & \cdots & A_{s,P-1} & A_{s,s} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{P-1} \\ x_s \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{P-1} \\ b_s \end{bmatrix}. \quad (2)$$

Применяя к СЛАУ (2) блочный метод Гаусса, получим СЛАУ только для интерфейсных неизвестных:

$$\tilde{A}_{s,s} x_s = \tilde{b}_s. \quad (3)$$

Таким образом, решив СЛАУ (3), можно вычислить неизвестные на интерфейсных границах до решения задач в подобластях и использовать полученные значения интерфейсных неизвестных как граничные условия первого рода для задач в подобластях. В результате задачи в подобластях становятся полностью независимыми и могут решаться параллельно. Для расчета правой части $\tilde{b}_s$ интерфейсной задачи (3) также необходимо решить независимые задачи в подобластях. Таким образом, на каждой итерации построенного метода декомпозиции области необходимо решить по две задачи в каждой подобласти и интерфейсную систему (3):

for $p = \overline{0, P-1}$ **do**
  $A_{p,p} t = b_p$
  $\tilde{b}_s^p = A_{s,p} t$
end for
$\tilde{b}_s = b_s - \sum_{p=0}^{P-1} \tilde{b}_s^p$
$\tilde{A}_{s,s} x_s = \tilde{b}_s$
for $p = \overline{0, P-1}$ **do**
  $A_{p,p} x_p = b_p - A_{p,s} x_s$
end for

(4)

Для расчета матрицы $\tilde{A}_{s,s}$ интерфейсной задачи (3) необходимо решить задачи в подобластях для каждого столбца матрицы:

for $p = \overline{0, P-1}$ **do**
  for $c = \overline{0, N-1}$ **do**
    $A_{p,p} t = \left[A_{p,s}\right]_c$
    $\left[\tilde{A}_{s,s}^p\right]_c = A_{s,p} t$
  end for
end for
$\tilde{A}_{s,s} = A_{s,s} - \sum_{p=0}^{P-1} \tilde{A}_{s,s}^p$

(5)

Для задач, в которых необходимо перестраивать сетку на каждом шаге расчета, перестроение вместе с ней матрицы $\tilde{A}_{s,s}$ по алгоритму (5) может оказывать серьезное влияние на время

счета. Но для задач с неподвижными погруженными границами процедуру (5) достаточно провести один раз – на этапе построения сетки.

4.2 Вычисление матриц для задач в подобластях и на интерфейсах

Теперь учтем, что в методе LS-STAG в 2D случае используется пятиточечный шаблон (в обозначениях сторон света – S, W, P, E, N), поэтому матрицы $A_{s,s}$, $A_{s,p}$ и $A_{p,s}$ формируются достаточно просто – ненулевые элементы в них появляются только благодаря вкладам от интерфейсных рядов контрольных объемов. В итоге получаем, что элементы W, P, E с интерфейсного ряда Γ_p идут в p-й диагональный блок $A_{s,s}^p$ матрицы $A_{s,s}$; элемент S идет в блоки $A_{s,p}^S$ и $A_{p,s}^N$ матриц $A_{s,p}$ и $A_{p,s}$; элемент N – в блоки $A_{s,p+1}^N$ и $A_{p+1,s}^S$ матриц $A_{s,p+1}$ и $A_{p+1,s}$. Отметим, что для первой и последней подобластей, имеющих интерфейсную границу только с одной стороны, половины перечисленных блоков не существует:

$$A_{s,0}^N = A_{s,P-1}^S = A_{0,s}^S = A_{P-1,s}^N = 0;\ x_s^{-1} = x_s^{P-1} = \tilde{b}_s^{-1} = \tilde{b}_s^{P-1} = 0;$$
$$\tilde{A}_{s,s}^{-1,k} = \tilde{A}_{s,s}^{P-1,k} = 0; \quad k = \overline{0, P-2}.$$

С учетом описанного вида матриц $A_{s,p}$ и $A_{p,s}$ алгоритм (4) принимает следующий вид:

```
1:  b̃_s = b_s
2:  for p = 0, P − 1 do
3:      A_{p,p} t = b_p
4:      b̃_s^{p−1} −= A_{p,s}^N t^0
5:      b̃_s^p −= A_{p,s}^S t^{M_p − 1}
6:  end for
7:  Ã_{s,s} x_s = b̃_s
8:  for p = 0, P − 1 do
9:      A_{p,p} x_p = b_p − A_{p,s}^S x_s^{p−1} − A_{p,s}^N x_s^p
10: end for
```
(6)

Алгоритм (5) расчета матрицы $\tilde{A}_{s,s}$ интерфейсной задачи (3) после подстановки матриц $A_{s,p}$ и $A_{p,s}$ для метода LS-STAG может быть представлен следующим образом:

```
1:  Ã_{s,s} = A_{s,s}
2:  for p = 1, P − 1 do
3:      for c = 0, N − 1 do
4:          A_{p,p} t = [A_{p,s}]_c
5:          [Ã_{s,s}^{p−1,p−1}]_c −= A_{s,p}^N t^0
6:          [Ã_{s,s}^{p,p−1}]_c −= A_{s,p}^S t^{M_p − 1}
7:      end for
8:  end for
9:  for p = 0, P − 2 do
10:     for c = 0, N − 1 do
11:         A_{p,p} t = [A_{p,s}]_{N_d − N + c}
12:         [Ã_{s,s}^{p−1,p}]_c −= A_{s,p}^N t^0
13:         [Ã_{s,s}^{p,p}]_c −= A_{s,p}^S t^{M_p − 1}
14:     end for
15: end for
```
(7)

Таким образом, хотя матрица $A_{s,s}$ в случае метода LS-STAG имеет блочно-диагональную структуру, причем каждый блок является трехдиагональной матрицей (ни одна пара интерфейсных рядов не является соседними и от каждого контрольного объема с интерфейсных рядов в эту матрицу идут только три ненулевых элемента), матрица $\tilde{A}_{s,s}$ интерфейсной задачи (3) имеет более сложную, блочно-трехдиагональную, структуру, причем каждый блок из нее – полностью заполненная матрица.

4.3 Метод решения интерфейсной задачи

Итак, в построенном параллельном алгоритме (6)-(7) для моделирования методом LS-STAG с описанной декомпозицией расчетной области, матрица $\tilde{A}_{s,s}$ интерфейсной СЛАУ (3) имеет блочно-трехдиагональную структуру. Соответственно, в программной реализации для ее хранения используется блочный формат, а не CSR как для всех остальных рассматриваемых матриц.

Заметим, что эту трехдиагональную блочную матрицу $\tilde{A}_{s,s}$ можно представить в виде произведения нижнетреугольной и верхнетреугольной блочных матриц. Такую LU-факторизацию матрицы интерфейсной СЛАУ можно хранить компактно в виде одной трехдиагональной блочной матрицы, так как блоки на главной блочной диагонали нижнетреугольной матрицы из разложения являются единичными матрицами и хранить их нет смысла.

Поскольку для решения интерфейсной СЛАУ (3) на каждом шаге расчета требуется только наличие LU-факторизации матрицы $\tilde{A}_{s,s}$, а сама матрица интерфейсной СЛАУ после построения этой факторизации в расчетах больше не требуется, выделенную изначально для хранения матрицы $\tilde{A}_{s,s}$ область памяти можно переиспользовать для хранения ее LU-факторизации.

Отметим, что при построении LU-факторизации требуется вычисление обратных матриц для блоков, составляющих главную диагональ верхнетреугольной матрицы разложения. Вместо непосредственного обращения этих матриц так же строим их LU-факторизации и сохраняем вместо исходных блоков на главной блочной диагонали.

5. Вычислительные эксперименты

Предложенный параллельный алгоритм реализован в разработанном программном комплексе "LS-STAG_DDM" [15] на языке C++. Для ускорения прототипирования в качестве технологии распараллеливания вычислений использовалась библиотека OpenMP, поэтому в рамках данной работы представлены вычислительные эксперименты, проведенные только на системах с общей памятью. В дальнейшем будет разработана реализация с использованием технологии параллельного программирования MPI, что позволит моделировать течения на системах с распределенной памятью.

В качестве решателя СЛАУ, соответствующих задачам в подобластях, используется итерационный метод FGMRES [1] с ILU-предобуславливателем для уравнения Гельмгольца и с многосеточным [16] предобуславливателем для уравнения Пуассона. Для решения СЛАУ, соответствующей задаче на интерфейсных границах между подобластями, реализован описанный выше блочный прямой решатель.

Верификация и оценка эффективности разработанного параллельного алгоритма проводятся на описанной выше тестовой задаче о моделировании двумерного обтекания неподвижного кругового профиля при значении числа Рейнольдса, равном 200, в течение 100 безразмерных единиц времени. Эксперименты проводились на последовательности из четырех неравномерных сеток (табл. 1). Каждая сетка содержит равномерный блок в окрестности рассматриваемого профиля, причем на диаметр профиля приходится ND ячеек. В качестве примера на рис. 2 показана сетка G_1.

Табл. 1 Основные характеристики используемых сеток
Table 1. Main characteristics of used meshes

Сетка	ND	$N \times M$	Δt
G_1	16	120 × 148	0.0100
G_2	32	240 × 296	0.0050
G_3	64	480 × 592	0.0010
G_4	128	960 × 1184	0.0005

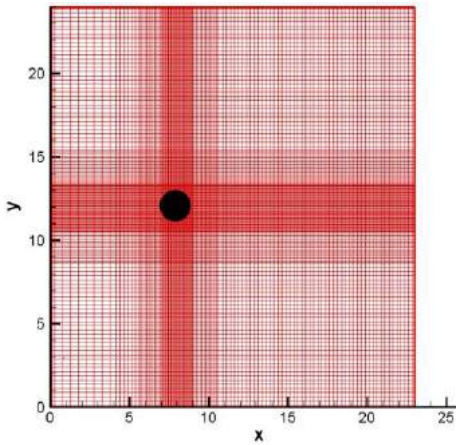


Рис. 2. Сетка G_1 .
Fig. 2. G_1 mesh.

5.1 Валидация алгоритма

Эксперименты, проведенные на сетках из табл. 1 при разном количестве подобластей, показали, что построенный и реализованный параллельный алгоритм позволяет получить такое же численное решение, что и исходный алгоритм без декомпозиции. При моделировании были рассчитаны нестационарные нагрузки $C_{xa}(t)$ и $C_{ya}(t)$. На рис. 3 и 4 представлены полученные зависимости для рассматриваемой последовательности сеток.

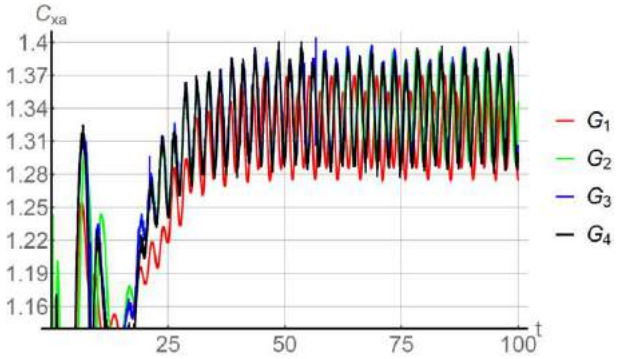


Рис. 3. Нестационарные нагрузки $C_{xa}(t)$, рассчитанные на сетках G_1 и G_2 без деления на подобласти и на сетках G_3 и G_4 с разбиением на 18 подобластей.
Fig. 3. Unsteady loads $C_{xa}(t)$, computed on G_1 and G_2 meshes without division into subdomains and on G_3 and G_4 meshes with division into 18 subdomains.

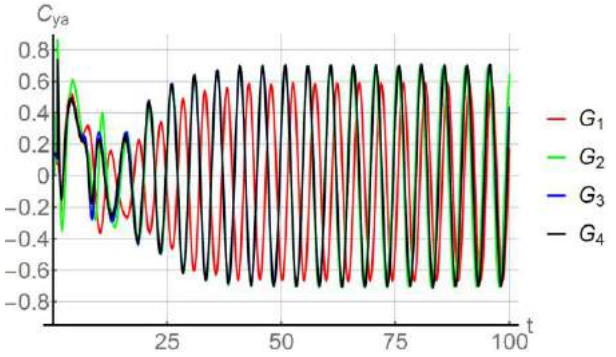


Рис. 4. Нестационарные нагрузки $C_{ya}(t)$, рассчитанные на сетках G_1 и G_2 без деления на подобласти и на сетках G_3 и G_4 с разбиением на 18 подобластей .

Fig. 4. Unsteady loads $C_{ya}(t)$, computed on G_1 and G_2 meshes without division into subdomains and on G_3 and G_4 meshes with division into 18 subdomains.

Видно, что по мере сгущения сетки результаты начинают сходиться несмотря на то, что представленные зависимости для подробных сеток получены при использовании декомпозиции расчетной области на 18 подобластей.

Кроме того, для каждого вычислительного эксперимента было рассчитано значение безразмерного стационарного аэродинамического коэффициента лобового сопротивления C_{xa} (табл. 2). Для этого смоделированные нестационарные нагрузки $C_{xa}(t)$ осреднялись по промежутку от 75 до 100 безразмерных единиц. Кроме того, в этом же временном диапазоне были проанализированы рассчитанные нестационарные зависимости $C_{ya}(t)$ с целью определения числа Струхаля Sh.

Табл. 2 Сравнение рассчитанных значений коэффициента лобового сопротивления C_{xa} и числа Струхаля Sh с известными в литературе данными

Table 2. Comparison of computed values of the drag coefficient C_{xa} and the Strouhal number Sh with data known in the literature

Источник	C_{xa}	**Sh**
Расчет на G_1 без декомпозиции	1.322	0.191
Расчет на G_1, 8 подобластей	1.320	0.194
Расчет на G_2 без декомпозиции	1.341	0.208
Расчет на G_2, 8 подобластей	1.340	0.208
Расчет на G_3 без декомпозиции	1.342	0.202
Расчет на G_3, 18 подобластей	1.343	0.202
Расчет на G_4 без декомпозиции	1.336	0.201
Расчет на G_4, 18 подобластей	1.336	0.201
Cheny & Botella [9], LS-STAG на сетке 550 × 350	1.327	0.200
Rajani и др. [17], расчет на сетке 386 × 322	1.316	0.196
Qu, Norberg, Davidson, Peng & Wang [18], расчет	1.337	0.196
Williamson [19], эксперимент	–	0.195
Wieselberger [20], эксперимент	1.330	–

В качестве примера в табл. 2 представлены некоторые из полученных значений C_{xa} и Sh для расчетов на разных сетках с разным числом подобластей. Видно, что по мере сгущения сетки рассчитанные значения коэффициентов перестают отличаться в зависимости от используемого числа подобластей. Хотя даже на самой грубой из используемых сеток эти значения отличаются не более чем на 1.5%, что также можно считать незначительным.

Также в табл. 2 приведены для сравнения известные в литературе экспериментальные и расчетные данные других исследователей. Заметим, что рассчитанные на сетке G_4 при разном числе подобластей значения коэффициента лобового сопротивления отличаются от экспериментально определенного на 0.45%, а значения числа Струхаля – на 3.08%. Таким образом, можно сделать вывод о хорошем согласовании смоделированных значений с экспериментальными, как при использовании метода LS-STAG без декомпозиции расчетной области, так и проведении расчетов с разбиением области на подобласти.

5.2 Исследование свойств алгоритма на грубых сетках

Расчеты на сетках G_1 и G_2 проводились на ПК с 8-ядерным процессором AMD Ryzen 7 5800H (Zen 3 с SMT, 16 потоков, 3,2 ГГц, кэш L3 16 Мб, 16 ГБ оперативной памяти DDR4-3200 и 512 ГБ SSD NVMe PCIe). Из графиков на рис. 5 можно сделать вывод, что на сетке G_1 задача недостаточно велика (около 14 рядов контрольных объемов в подобласти при делении на 8 подобластей). Ускорение, полученное на сетке G_2 сравнимо с результатами, представленными в работе [14] для задачи сопоставимого размера (400 × 100 ячеек, 1..8 узлов).

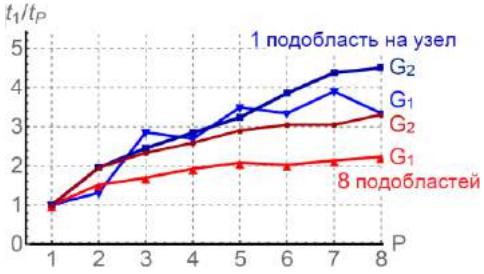


Рис. 5. Ускорение вычислений на грубых сетках.
Fig. 5. Acceleration of computations on coarse meshes.

Необходимо отметить, что разработанный алгоритм с декомпозицией области позволяет ускорить моделирование даже на 1 узле/потоке за счет уменьшения количества итераций, т.е. метод декомпозиции области с дополнением Шура работает как дополнительный предобуславливатель. В случае уравнения Гельмгольца для прогноза скоростей задача во всей расчетной области решается за 1 итерацию для любого числа подобластей больше 1, в то время как для решения задачи без подобластей требуется 2 итерации метода FGMRES с ILU-предобуславливателем. Решение плохообусловленной СЛАУ, возникающей из уравнения Пуассона для функции давления, без подобластей требует около 8-12 итераций метода FGMRES с многосеточным предобуславливанием. Это количество можно сократить всего до 1-2 итераций при оптимальном выборе числа подобластей.

Как показали численные эксперименты на рассматриваемых грубых сетках G_1 и G_2, в случае расчетов в последовательном режиме наилучшее ускорение достигается при количестве уравнений для задач в подобластях в 2 раза большем, чем для интерфейсной системы. Таким образом, лучше использовать 8 подобластей при расчетах в один поток на сетке G_1 и 16 подобластей на сетке G_2. Получаемое при этом ускорение составляет 1.55 раз и 2.32 раза соответственно.

5.2 Исследование свойств алгоритма на подробных сетках

Расчеты на сетках G_3 и G_4 проводились на учебно-экспериментальном кластере кафедры «Прикладная математика» МГТУ им. Н.Э. Баумана. Кластер состоит из управляющего узла и семи вычислительных. Управляющий узел кластера имеет два четырехядерных процессора Intel Xeon E5620 и оборудован 48 Гб ОЗУ. Он оснащен жестким диском 2 Тб (для собственных нужд) и RAID-массивом объемом 12 Тб для данных пользователей, тремя сетевыми контроллерами (Gigabit Ethernet – для связи с внешним миром; 10-Гб Ethernet – для связи с узлами кластера; Mellanox InfiniBand FDR, 2x54 Гбит/с – основной интерконнект), а также видеокартой Nvidia Geforce GTX 970. Каждый вычислительный узел имеет 18-ядерный процессор Intel Core i9-10980XE, 128 Гб ОЗУ, видеоускоритель Nvidia Titan V и два сетевых интерфейса: 10-Гб Ethernet и Mellanox InfiniBand (FDR, 2x54 Гбит/с). На тесте Linpack данный кластер показал для CPU максимальную производительность 8.82 Тфлопс, составляющую около 97 % от пиковой.

Для каждой рассматриваемой в этом разделе сетки расчеты проводились на одном вычислительном узле кластера с использованием числа ядер (потоков OpenMP), равным числу подобластей. Серия экспериментов для каждой сетки состояла из 18 расчетов: расчета методом LS-STAG без декомпозиции в последовательном режиме и расчетов с декомпозицией на 2..18 подобластей. Полученное благодаря декомпозиции ускорение представлено на рис. 6.

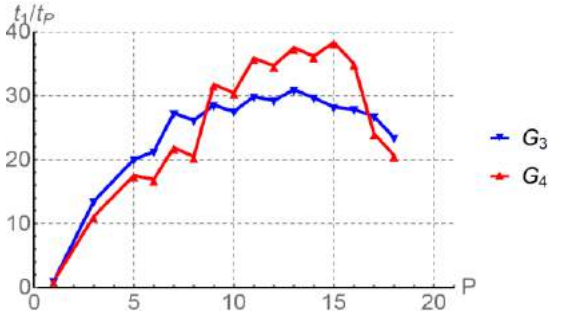


Рис. 6. Ускорение вычислений на подробных сетках.
Fig. 6. Acceleration of computations on fine meshes.

В первую очередь, из рис. 6 видно, что при любом рассматриваемом числе подобластей получаемое ускорение превышает число используемых для распараллеливания потоков OpenMP. Такое сверхлинейное ускорение объясняется хорошими предобуславливающими свойствами метода декомпозиции области с дополнением Шура, которые уже обсуждались выше при рассмотрении вычислительных экспериментов на грубых сетках. Наилучшее ускорение на сетке G_3 составляет 30.9 раз и достигается при использовании 13 подобластей (и потоков), а на сетке G_4 – 38.3 раза при использовании 15 подобластей.

Кроме того, видно, что при многих четных значениях числа подобластей ускорение меньше, чем при соседнем нечетном. Это объясняется тем, что при выбранном способе разбиения на подобласти одна из интерфейсных границ в первом случае будет проходить через ячейки, полностью занятые твердым телом, как показано на рис. 7 на примере разбиения расчетной области на две подобласти. Это делает интерфейсную СЛАУ плохообусловленной. Когда подобластей и, соответственно, интерфейсных границ, немного, численные ошибки при решении такой плохообусловленной системы могут быть особенно критичны – так, например, для случая двух и четырех подобластей на сетках G_3 и G_4 итерационный процесс в итоге перестал сходиться. Вероятно, данная проблема может быть решена заменой описанного выше прямого решателя интерфейсной СЛАУ на более сложный вариант, лучше

подходящий для решения плохообусловленных систем. Этот вопрос требует дальнейшего исследования.

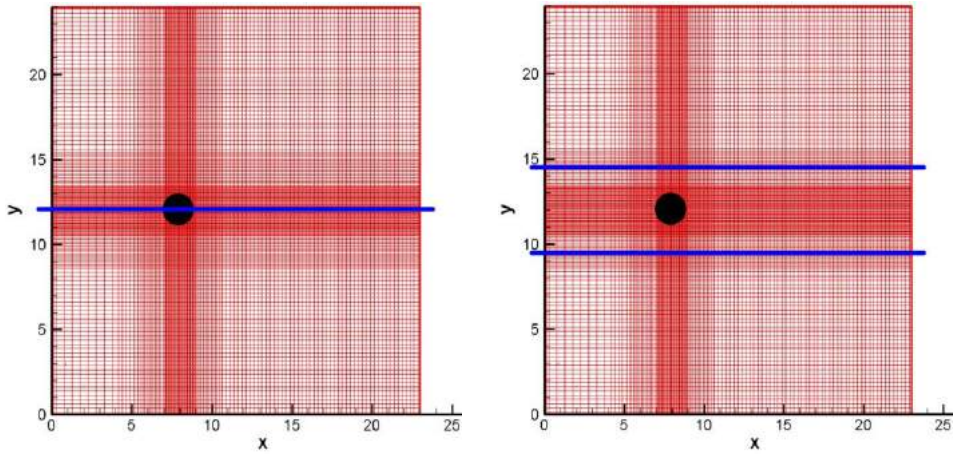


Рис. 7. Разделение сетки G_1 на 2 (слева) и 3 (справа) подобласти.
Fig. 7. The G_1 mesh division into 2 (left) and 3 (right) subdomains.

6. Заключение

С использованием метода декомпозиции области, в частности метода дополнения Шура, построен параллельный алгоритм для моделирования течений вязкой несжимаемой среды методом погруженных границ LS-STAG с усеченными ячейками и функциями уровня. Представлен алгоритм построения задач в подобластях и задачи на интерфейсных границах. Для ускорения прототипирования программная реализация описанного метода разработана с использованием технологии параллельного программирования OpenMP, поэтому вычислительные эксперименты проводились только на системах с общей памятью, в частности на отдельных узлах учебно-экспериментального кластера кафедры «Прикладная математика» МГТУ им. Н.Э. Баумана. В дальнейшем планируется реализовать в разрабатываемом программном комплексе распределенное хранение матриц и векторов и использовать для распараллеливания решения задач в подобластях технологию MPI.

Верификация разработанного метода проводилась на хорошо изученной тестовой задаче об обтекании неподвижного кругового профиля при значении числа Рейнольдса, равном 200. На рассматриваемой последовательности сеток предлагаемый параллельный алгоритм позволяет получить такое же численное решение, что и исходный алгоритм метода LS-STAG без декомпозиции расчетной области. Кроме того, рассчитанные значения коэффициенты лобового сопротивления и числа Струхаля во всех проведенных вычислительных экспериментах хорошо согласуются с известными в литературе экспериментальными и расчетными данными других исследователей.

Как показали проведенные расчеты на грубых сетках, метод декомпозиции области работает также как дополнительный предобуславливатель: даже в последовательном режиме декомпозиция области позволяет ускорить моделирование за счет уменьшения необходимо для достижения заданной точности числа итераций решателя СЛАУ. Например, плохообусловленную СЛАУ, возникающую из уравнения Пуассона для функции давления, удается решить в среднем за 1-2 итерации в каждой подобласти, в то время как в варианте без декомпозиции расчетной области ее решение занимает от 8 до 12 итераций метода FGMRES с многосеточным предобуславливанием.

Благодаря таким хорошим предобуславливающим свойствам используемого метода декомпозиции области в вычислительных экспериментах на подробных сетках было получено сверхлинейное ускорение. В дальнейшем планируется доработать решатель задачи на интерфейсных границах, чтобы снизить влияние обусловленности интерфейсной задачи на сходимость метода во всей расчетной области.

Список литературы / References

[1]. Saad Y., Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2003. 184 p.
[2]. Cools S., Vanroose W. The communication-hiding pipelined BiCGstab method for the parallel solution of large unsymmetric linear systems. Parallel Computing, 2017, vol. 65, pp. 1-20. DOI: 10.1016/j.parco.2017.04.005
[3]. Quarteroni A., Valli A. Domain Decomposition Methods for Partial Differential Equations. Oxford, 1999. 376 p.
[4]. OpenFOAM: API Guide: OpenFOAM®: Open source CFD: API, Available at: https://www.openfoam.com/documentation/guides/latest/api/index.html, accessed 23.10.2023.
[5]. Scotch: a software package for graph and mesh/hypergraph partitioning, graph clustering, and sparse matrix ordering, Available at: https://gitlab.inria.fr/scotch/scotch, accessed 23.10.2023.
[6]. METIS, Available at: https://github.com/KarypisLab/METIS, accessed 23.10.2023.
[7]. Mittal R., Iaccarino G. Immersed boundary methods. Annu. Rev. Fluid Mech, 2005, vol. 37, pp. 239–261. DOI: 10.1146/annurev.fluid.37.061903.175743
[8]. Osher S. J., Fedkiw R. Level set methods and dynamic implicit surfaces. Springer, 2003. 286 p.
[9]. Cheny Y., Botella O. The LS-STAG method: A new immersed boundary/level-set method for the computation of incompressible viscous flows in complex moving geometries with good conservation properties. J. Comput. Phys., 2010, vol. 229, pp. 1043-1076. DOI: 10.1016/j.jcp.2009.10.007
[10]. Nikfarjam F., Cheny Y., Botella O. The LS-STAG immersed boundary / cut-cell method for non-Newtonian flows in 3D extruded geometries. Comp. Phys. Comm., 2018, vol. 226, pp. 67-80. DOI: 10.1016/j.cpc.2018.01.006
[11]. Marchevsky I.K., Puzikova V.V. Modification of the LS-STAG immersed boundary method for simulating turbulent flows. Herald of the Bauman Moscow State Technical University. Natural Sciences, 2017, № 5, pp. 19-34. DOI: 10.18698/1812-3368-2017-5-19-34
[12]. Marchevsky I.K., Puzikova V.V. The modified LS-STAG method application for planar viscoelastic flow computation in a 4:1 contraction channel. Herald of the Bauman Moscow State Technical University. Natural Sciences, 2021, № 3, pp. 46-63. DOI: 10.18698/1812-3368-2021-3-46-63
[13]. Pacull F., Garbey M. A parallel immersed boundary method for blood-like suspension flow simulations. Lect. Notes in Comp. Sci. and Eng., 2010, vol. 74, pp. 1-8. DOI: 10.1007/978-3-642-14438-7_16
[14]. Wiens J. K., Stockie J. M. An efficient parallel immersed boundary algorithm using a pseudo-compressible fluid solver. J. Comp. Phys., 2015, vol. 281, pp. 917-941. DOI: 10.1016/j.jcp.2014.10.058
[15]. Пузикова В. В. Параллельный программный комплекс LS-STAG_DDM для моделирования методом LS-STAG с декомпозицией области; свидетельство о государственной регистрации программы для ЭВМ № 2023665707, зарегистрировано в Реестре программ для ЭВМ 19.07.23 / Puzikova V. V. Parallel software package LS-STAG_DDM for simulation by using the LS-STAG method with domain decomposition; Certificate of state registration of a computer program No. 2023665707, registered in the Register of Computer Programs on 07/19/23 (in Russian).
[16]. Wesseling P. An introduction to multigrid methods. John Willey & Sons Ltd., 1991. 296 p.
[17]. Rajani B. N., Kandasamy A., Majumdar S. Numerical simulation of laminar flow past a circular cylinder. Applied Mathematical Modeling, 2009, vol. 33, pp. 1228-1247. DOI: 10.1016/j.apm.2008.01.017
[18]. Qu L., Norberg C., Davidson L., Peng S.-H., Wang F. Quantitative numerical analysis of flow past a circular cylinder at Reynolds number between 50 and 200. J. Fluids Struct., 2013, vol. 39, pp. 347-370. DOI: 10.1016/j.jfluidstructs.2013.02.007
[19]. Williamson C. H. K. Vortex dynamics in the cylinder wake. Annu. Rev. Fluid. Mech., 1996, vol. 28, pp. 477-539. DOI: 10.1146/annurev.fl.28.010196.002401
[20]. Wieselberger C. New data on the laws of fluid resistance.Technical Notes. National Advisory Committee for Aeronautics, 1923, № 84, pp. 1-16.

Информация об авторах / Information about authors

Илья Константинович МАРЧЕВСКИЙ – доктор физико-математических наук, доцент, профессор кафедры «Прикладная математика» МГТУ им. Н.Э. Баумана. Сфера научных интересов: вычислительная гидродинамика, вихревые методы, теория устойчивости, численные методы, высокопроизводительные вычисления, элементарная математика.

Ilia Konstantinovich MARCHEVSKY – Dr. Sci. (Phys.-Math.), Associate professor, Professor of Applied Mathematics department, Bauman Moscow State Technical University\. Research interests: computational fluid dynamics, vortex particle methods, theory of stability, numerical methods, high performance computing, elementary mathematics.

Валерия Валентиновна ПУЗИКОВА – кандидат физико-математических наук, эксперт по разработке программного обеспечения Департамента разработки высокопроизводительных библиотек компании YADRO. Сфера научных интересов: решатели и предобуславливатели СЛАУ, разработка прикладных математических программ, вычислительная гидродинамика, физические движки для AR/VR, высокопроизводительные вычисления, численные методы.

Valeria Valentinovna PUZIKOVA – Cand. Sci. (Phys.-Math.), Software Development Expert in High Performance Libraries Department, YADRO. Research interests: solvers and preconditioners for SLAE, applied mathematics software development, computational hydrodynamics, physics engines for AR/VR, high performance computations, numerical methods.