

ТРУДЫ

**ИНСТИТУТА СИСТЕМНОГО
ПРОГРАММИРОВАНИЯ РАН**

**PROCEEDINGS OF THE INSTITUTE
FOR SYSTEM PROGRAMMING OF THE RAS**

ISSN Print 2079-8156
Том 38 Выпуск 1

ISSN Online 2220-6426
Volume 38 Issue 1

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2026

ИСП **РАН**

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 областям 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:



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 институт Technion (Хайфа, Израиль)

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/>

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)

[Andrew Voronkov](#), Dr. Sci. (Phys.–Math.), Professor, University of Manchester (Manchester, United Kingdom)

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>

С о д е р ж а н и е

Поиск максимального взвешенного независимого набора вершин на гиперграфе логических ограничений. <i>Рыжова Д.И., Касаткин В.И.</i>	7
Учет временных ограничений на этапе размещения элементов на ПЛИС. <i>Шокарев Д.Б., Чочаев Р.Ж., Гаврилов С.В.</i>	17
Декомпозиция Ашенхёрста для таблично заданных булевых функций. <i>Емельянов П.Г.</i>	33
Поиск ошибок в исходном коде на C# на основе статического анализа помеченных данных. <i>Беляев М.В., Рагозина П.И., Игнатъев В.Н.</i>	45
Версионирование кода как развитие оптимизации методом Распространения констант. <i>Зинин И.А., Черноног В.В.</i>	61
Escape-анализ с экзистенциальными метками времени жизни. <i>Стоян А.С.</i>	71
Легковесная проверка согласованности файловой системы после сбоя с помощью дифференциального фаззинга. <i>Ковалевский В.М., Кечин В.В., Янин А.С., Ицыксон В.М.</i>	79
Использование аугментации при обучении на некомплектной выборке. <i>Лазарев Д.О., Шокуров А.В., Фомин С.А.</i>	93
Оценка паразитной ёмкости цепей и быстродействия блока СБИС с помощью машинного обучения на этапе размещения. <i>Желудков Н.В., Кулемина Ю.А.</i>	113
Новая интерпретации временной системы вах-васюганского хантыйского на современных корпусных данных платформы ЛингвоДок. <i>Воробьева В.В., Федоринова З.В.</i>	133
Автоматизация обработки речевого сигнала: аннотация и акустический анализ гласных в русском языке. <i>Гончарова О.В.</i>	149
Исследование методов оптимизации вычислительной сложности моделей машинного обучения с использованием тензорных поездов. <i>Лапина М.А., Лапин В.Г., Караева А.Д., Парфентьев М.Е., Селиванов М.Е.</i>	163
Анализ и оптимизация задержек видеотрансляции удалённых рабочих столов. <i>Проскуряков И.С., Бадалян Д.А.</i>	183
Комплексное применение компьютерного зрения и распознавания речи для повышения качества обработки видеозаписей производственных работ. <i>Смолин И.Ю., Любченко А.А., Исаков М.В., Вожадаева Д.А.</i>	201

Сбор данных для обучения и тестирования моделей сверхразрешения, работающих в реальном времени.
Богатырев Е.Н., Молодецких И.А., Абуд Х.Н., Ватолин Д.С......221

АпАТ: состязательное дообучение с использованием якорей для метрик оценки качества изображений.
Гущин А.Е., Анциферова А.В., Ватолин Д.С......241

Адаптивное восстановление фрагментов видеопотока на основе машинного обучения с учётом динамики сцены.
Думкин Н.А., Александров Д.В., Прозорский М.А. 255

Извлечение информации из новостных веб-страниц, содержащих множество записей.
Кустенков А.А., Яцков А.К., Варламов М.И. 275

Table of Contents

Calculation of the maximum weighted independent set in a logical constraint hypergraph. <i>Ryzhova D.I., Kasatkin V.I.</i>	7
Timing constraints during FPGA placement. <i>Shokarev D.B., Chochaev R.Z., Gavrilov S.V.</i>	17
Ashenhurst decomposition for table-defined Boolean functions. <i>Emelyanov P.G.</i>	33
Data augmentation for machine learning on missing data. <i>Belyaev M.V., Ragozina P.I., Ignatyev V.N.</i>	45
Code versioning as an evolution of Constant Propagation optimization. <i>Zinin I.A., Chernonog V.V.</i>	61
Type-based Escape Analysis with Existential Lifetimes. <i>Stoyan A.S.</i>	71
Lightweight file system crash-consistency checking with differential fuzzing. <i>Kovalevsky V.M., Kechin V.V., Yanin A.S., Itsykson V.M.</i>	79
Data augmentation for machine learning on missing data. <i>Lazarev D.O., Shokurov A.V., Fomin S.A.</i>	93
Machine learning-based estimation of interconnect capacitance and block performance in VLSI design at placement stage. <i>Zheludkov N.V., Kulemina Y.A.</i>	113
A new interpretation of the tense system of Vakh-Vasyugan Khanty on the up-to-date corpus data of the LingvoDoc Lab. <i>Vorobeva V.V., Fedorinova Z. V.</i>	133
Automating speech signal processing: annotation and acoustic analysis of Russian vowels. <i>Goncharova O.V.</i>	149
Research into methods for optimizing the computational complexity of machine learning models using tensor trains. <i>Lapina M.A., Lapin V.G., Karaeva A.D., Parfentyev M.E., Selivanov M.E.</i>	163
Analysis and optimization of video transmission delays in remote desktops. <i>Proskuriakov I.S., Badalyan D.A.</i>	183
Integrated application of computer vision and speech recognition to improve the quality of processing video recordings of production works. <i>Smolin I.U., Lyubchenko A.A., Isakov M.V., Vozhdaeva D.A.</i>	201

Methodology for collecting and analyzing subjective prominence of artifacts in learning-based super-resolution.
Bogatyrev E.N., Molodetskikh I.A., Abud K.N., Vatolin D.S. 221

AnAT: Anchor-Based Adversarial Training for Image Quality Assessment.
Gushchin A.E., Antsiferova A.V., Vatolin D.S. 241

Machine learning-based adaptive reconstruction of video stream fragments taking into account scene dynamics.
Dumkin N.A., Alexandrov D.V., Prozorskiy M.A. 255

Multi-record web page information extraction from news websites.
Kustnikov A.A., Yatskov A.K., Varlamov M.I. 275

DOI: 10.15514/ISPRAS-2026-38(1)-1



Поиск максимального взвешенного независимого набора вершин на гиперграфе логических ограничений

¹ Д.И. Рыжова, ORCID: 0009-0005-1975-6195 <darrlight@gmail.com>

² В.И. Касаткин, ORCID: 0009-0003-6454-6379 <kasatkin.vlad@icloud.com>

¹ Национальный исследовательский центр “Курчатовский институт”,
Россия, 123182, г. Москва, пл. Академика Курчатова, д. 1.

² Национальный исследовательский университет МИЭТ,
Россия, 124498, Москва, Зеленоград, пл. Шокина, д. 1.

Аннотация. Задача о поиске независимого множества максимального веса на графе относится к классу NP-полных задач и используется в задачах математического моделирования и оптимизации, которые можно формализовать в терминах теории графов. Разработанный алгоритм поиска максимального взвешенного независимого набора вершин можно применять в рамках оценки пикового тока в комбинационных интегральных схемах и сложнофункциональных блоках небольшой размерности – в пределах нескольких сотен вентилях. В таком случае связанный граф будет являться представлением списка соединений между узлами тестовой схемы, а максимальное взвешенное независимое множество вершин будет соответствовать множеству узлов, которые переключатся одновременно и дадут максимальный мгновенный ток. Для ускоренного поиска независимого множества максимального веса используется комбинация метода ветвей и границ и жадного алгоритма.

Ключевые слова: максимальный взвешенный независимый набор вершин (МВННВ); сверхбольшие интегральные схемы (СБИС); пиковый ток; гиперграф; метод ветвей и границ; жадный алгоритм.

Для цитирования: Рыжова Д.И., Касаткин В.И. Поиск максимального взвешенного независимого набора вершин на гиперграфе логических ограничений. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 7–16. DOI: 10.15514/ISPRAS-2026-38(1)-1

Благодарности: Работа проведена в рамках выполнения государственного задания НИЦ «Курчатовский институт».

Calculation of the Maximum Weighted Independent Set in a Logical Constraint Hypergraph

¹ D.I. Ryzhova, ORCID: 0009-0005-1975-6195 <darrlight@gmail.com>

² V.I. Kasatkin, ORCID: 0009-0003-6454-6379 <kasatkin.vlad@icloud.com>

¹ National Research Center «Kurchatov institute»,
Bld. 1, Academician Kurchatov Square, Moscow, 123182, Russia.

² National Research University of Electronic Technology – MIET,
Bld.1, Shokin Square, Zelenograd, Moscow, 124498, Russia.

Abstract. The problem of finding a maximum weighted independent set on a graph is NP-complete and is used in mathematical simulation and optimization problems that can be formalized in terms of graph theory. We have developed an algorithm for finding a maximum weighted independent set, which can be applied to peak current estimation in combinational integrated circuits and small-scale complex functional blocks (within a few hundred gates). In this case, the connected graph represents a list of connections between the test circuit nodes, and the maximum weighted independent set of nodes corresponds to the set of nodes that switch simultaneously and produce the maximum instantaneous current. To accelerate the search for a maximum weighted independent set, a combination of the branch-and-bound method and a greedy algorithm is used.

Keywords: maximum weighted independent set (MWIS); very large-scale integrated circuits (VLSI); peak current; hypergraph; branch-and-bound method; greedy algorithm.

For citation: Ryzhova D.I., Kasatkin V.I. Calculation of the maximum weighted independent set in a logical constraint hypergraph. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 7-16 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-1.

Acknowledgements. The work was carried out within the state assignment of NRC «Kurchatov institute».

1. Введение

Задача о независимом множестве максимального веса используется для решения различных задач в области теории графов. Она относится к классу NP-полных задач и обычно используются для доказательства вычислительной сложности теоретических задач, а также для моделирования задач оптимизации [1-2].

Задача поиска максимального взвешенного независимого набора вершин (МВННВ) формулируется следующим образом. Пусть дан неориентированный граф $G = (V, E)$, где каждая вершина из V соответствует некоторому выбору/варианту, а ребро $(u, v) \in E$ означает, что варианты u и v несовместимы и не могут быть выбраны одновременно. Каждой вершине v соответствует неотрицательный вес $w(v)$, интерпретируемый как ценность или выгода этого выбора. Требуется найти подмножество $S \subseteq V$, которое, во-первых, является независимым (то есть для любых $u, v \in S$ верно, что $(u, v) \notin E$, конфликтов внутри S нет), а во-вторых, максимизирует суммарный вес $\sum_{v \in S} w(v)$. Правила вида “mw_term <комбинация 0 или 1> <X>” понимаются как задание конфликтов внутри набора X : для любых различных u, v из X в граф добавляется ребро (u, v) ; тем самым список таких запретов эквивалентен построению E и полностью задаёт допустимые совместные выборы [3-4].

В данной статье рассматривается применение этой задачи к проблеме оценки пикового тока (тока на шине питания), возникающего в цифровых КМОП-схемах вследствие синхронного переключения множества вентилях в один и тот же временной промежуток [5]. Ключевая идея состоит в том, чтобы еще на логическом уровне заранее определить, какие наборы выходных узлов не могут менять свое состояние одновременно по законам функционирования самой схемы. Руководствуясь этим принципом, можно сильно сузить множество возможных комбинаций, приводящих к переключению выхода при переключении верхних (pull-up) цепочек вентилях и получить уточненное верхнее ограничение на оценку пикового тока.

2. Формализация задачи поиска МВННВ

2.1 Метод резолюций

Классический метод резолюций представляет собой метод вывода новых булевых соотношений из заданного множества. Задана пара (M, V) , где $M = \{a, b, \dots\}$ – множество булевых переменных; а V – множество булевых соотношений типа $A=B$; A, B – выражения на множестве M . В базовой версии метода резолюций множество V преобразуется в одно соотношение $F=1$, где F представляет собой конъюнктивную нормальную форму (КНФ), то есть произведение сумм переменных или их отрицаний (литералов).

Пусть F – характеристическая функция системы булевых соотношений. Тогда для двухвходового вентиля II со входами a, b и выходом y логическая функция описывается булевым соотношением $y = a * b$. Оно может быть преобразовано в:

$$(x + (\sim a) + (\sim b)) * (\sim x + a) * (\sim x + b) = 1,$$

где характеристическая функция вентиля II находится в левой части уравнения.

Для получения новых соотношений используется следующее правило резолюции:

$$a + B = 1, \sim a + C = 1 \rightarrow B + C = 1,$$

где B, C – суммы литералов.

В модифицированном методе резолюций [6] множество V преобразуется в одно соотношение $d=0$, где d – это дизъюнктивная нормальная форма или ДНФ (сумма произведений литералов). Например, для вентиля II имеем:

$$\sim x * a * b + x * (\sim a) + x * (\sim b) = 0.$$

Этот метод связан с системой логических импликаций. Каждый терм – это запрет на определенную комбинацию переменных. Например, член $x * (\sim a)$ соответствует запрету на комбинацию ($x=1, a=0$). Эта комбинация эквивалентна простым логическим импликациям (ПЛИ) $\sim a \rightarrow \sim x, x \rightarrow (\sim a)$. Аналогично, член $\sim x * a * b$ описывает логическую импликацию $a * b \rightarrow x$.

Для определения новых логических ограничений используется правило резолюции:

$$a * B = 1, \sim a * C = 1 \rightarrow B * C = 0,$$

где B, C – произведения литералов.

В стандартной записи « $=0$ » обычно опускается. Система ограничений для вентиля II будет записываться как $(\sim x * a * b, x * (\sim a), x * (\sim b))$, а правило резолюции будет записываться как $(a * B, \sim a * C \rightarrow B * C)$.

Таким образом, модифицированный метод резолюций позволяет конвертировать сложную электрическую задачу в комбинаторную, которую можно решить быстрыми графовыми алгоритмами. Это дает разработчикам надежную оценку пикового тока потребления уже на этапе логического синтеза, позволяя, к примеру, заранее вносить коррективы в топологию сетей/колец питания.

2.2 Задача оптимизации на графе

В результате применения модифицированного метода резолюций к нетлисту КМОП-схемы мы получаем графовую модель в виде гиперграфа логических ограничений. В нем вершины графа соответствуют выходам вентиля; гиперребра (связи между несколькими вершинами) задают запреты на одновременное переключение: ребро между двумя вершинами запрещает состояние 0-0 или 1-1, ребро из трех вершин запрещает состояние 0-0-0 или 1-1-1 и т. д.; вес вершины соответствует значению тока на шине питания при переключении конкретного вентиля. Метод позволяет оценить пиковый ток в схеме, учитывая корреляции высокого порядка между переключениями вентиля. Основная сложность – NP-полнота задачи,

поэтому на практике ограничиваются гиперребрами небольшого размера (обычно гиперребро соединяет не более пяти вершин).

Оценка максимального тока сводится к поиску МВННВ в гиперграфе. Независимое множество – это подмножество вершин, не связанных ни одним гиперребром. Цель – найти набор с максимальной суммой весов.

Пусть задан неориентированный граф $G = (V, E)$, где V – множество вершин (узлов схемы), а E – множество ребер (логических ограничений). Каждой вершине $v \in V$ приписан вес $w(v)$, отражающий ток, потребляемый при переключении. Требуется найти подмножество вершин $S \subset V$, такое что:

- 1) Вершины из S попарно не соединены ребрами ($\forall u, v \in S, (u, v) \notin E$), что соответствует отсутствию логических ограничений на их одновременное переключение.
- 2) Суммарный вес $\sum_{v \in S} w(v)$ максимален, что означает вычисление пикового тока.

Логические ограничения представляют собой набор запретов вида “mw_term <комбинация 0 или 1> <X>”, где X – множество узлов, которые не могут одновременно переключаться. При этом, в качестве узлов следует выбирать только выходы вентилях. Это и позволяет находить одновременное переключение. В терминах графа эти ограничения формируют дополнительные ребра, уменьшая допустимое множество независимых вершин.

3. Программная реализация алгоритма поиска МВННВ на гиперграфе

В работе была выбрана и реализована комбинация метода ветвей и границ и жадного алгоритма. Это обосновано хорошей и точной работой первого алгоритма для небольших схем, и способностью быстро и достаточно точно обрабатывать большие схемы вторым алгоритмом [7-8].

В качестве входных данных программы используются: файл, содержащий описание схемы в формате Verilog, и файл логических ограничений (сгенерированный программой, предоставленной отделением проблем проектирования в микроэлектронике Центра перспективной микроэлектроники). Далее происходит трансляция нетлиста во внутренний формат, и выходы логических элементов становятся вершинами в гиперграфе ограничений. Токи каждого элемента вычисляются как результат деления мощности на напряжение питания, при этом значения внутренней мощности выбираются из библиотечного Liberty файла, исходя из заданных временных ограничений (файл формата SDC) и выходных емкостей. Далее происходит построение гиперграфа логических ограничений, гиперребра которого означают запрет на состояния, указанные в виде логических ограничений. В зависимости от размера схемы, выбранным методом производится расчет максимального взвешенного набора вершин, однако в списке операций жадного алгоритма нет взаимодействия с гиперграфами, поэтому они там не используются. В качестве результата программа выводит МВННВ и пиковый ток для заданной схемы.

3.1 Метод ветвей и границ

Основные операции метода ветвей и границ (рис. 1):

- Ветвление: Разбиение всего пространства решений на части. В задачах комбинаторной оптимизации принимается один частичный выбор с последующим рассмотрением двух вариантов: «включить» или «не включить» элемент. Узлами дерева являются подобласти (подмножества) значений.
- Границы: для каждой подзадачи оцениваем максимум или минимум, которого можно достичь, развивая эту подзадачу. Такое значение называется верхней или нижней границей. Если приходит понимание, что даже в наилучшем случае подзадача не даст лучший результат, ее можно отбросить.

- Поиск: Алгоритм рекурсивно углубляется по дереву, запоминая лучшее решение, найденное в ходе поиска.

Применительно к задаче анализа максимального тока потребления КМОП схем метод ветвей и границ включает следующие этапы:

- 1) Инициализация: Значение пикового тока приравняется к нулю.
- 2) Определение подмножества: Нахождение подмножества вентилях, способных на одновременное переключение. Если оно увеличивает ток, такое значение принимается как новая максимальная оценка.
- 3) Обход множества: Алгоритм анализирует все возможные подмножества. Завершение работы происходит после анализа всех узлов дерева.

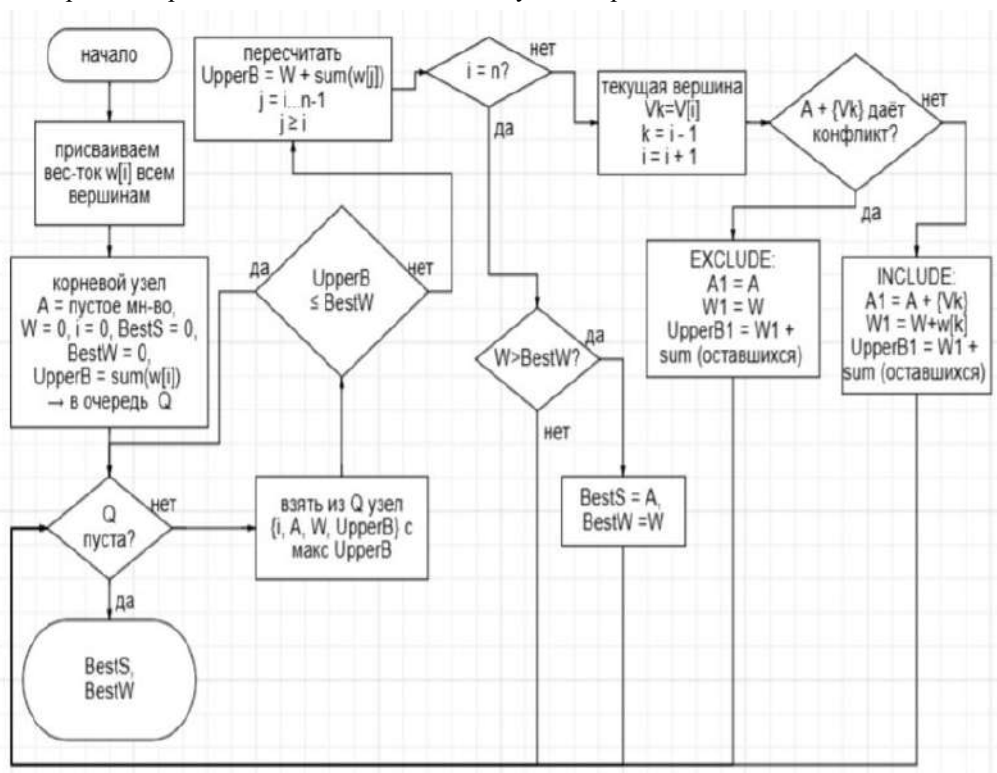


Рис. 1. Блок-схема метода ветвей и границ.

Fig. 1. Branch-and-bound flowchart.

3.2 Жадный алгоритм

Основные операции жадного алгоритма (рис. 2):

- Определение ценности: определяем меру полезности или стоимость всех элементов множества.
- Сортировка по жадному критерию: элементы, из которых будет строиться решение сортируются по убыванию или возрастанию, в зависимости от ценности.
- Формирование набора: наполняем набор лучшими элементами, проверяя допустимость их включения.

Применительно к задаче анализа максимального тока потребления КМОП схем жадный алгоритм включает следующие этапы:

- 1) Инициализация: создается пустое множество.
- 2) Оценка и сортировка: присваиваем элементам вес и сортируем их по убыванию.
- 3) Выбор кандидата: выбираем элементы с наибольшим весом, следуя указанным ограничениям. Таким образом формируем итоговый набор.

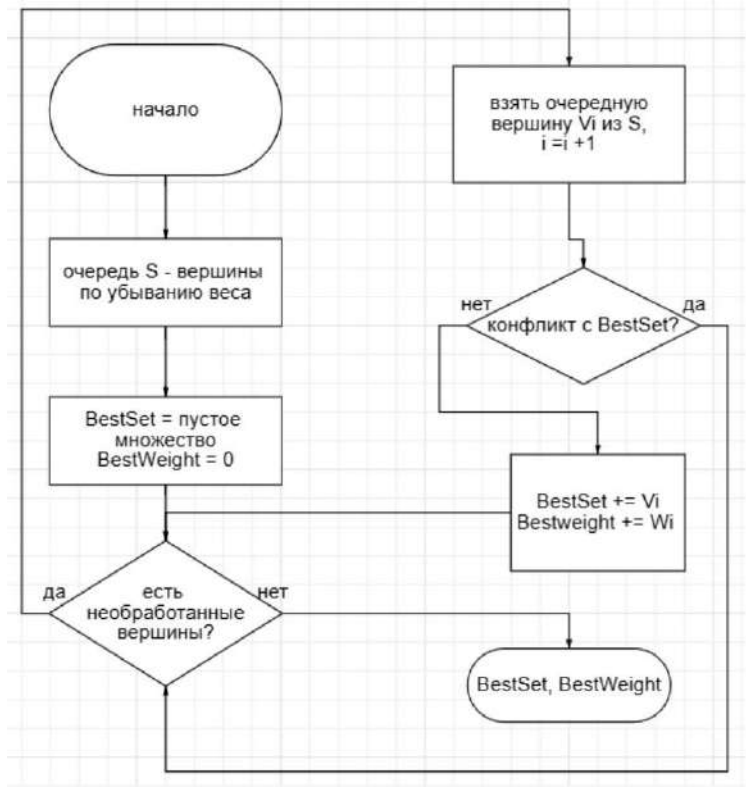


Рис. 2. Блок-схема жадного алгоритма.
Fig. 2. Greedy algorithm flowchart.

4. Пример работы алгоритма

Рассмотрим работу алгоритма в рамках выполнения задачи расчета пикового тока для схемы C17 из тестового набора ISCAS 85.

Verilog-описание схемы представлено на рис. 3, фрагмент описания логических ограничений приведен на рис. 4.

Граф парных ограничений строится на основе изученного ранее метода резолюций, для нахождения суммы весов вентилей, переключающихся одновременно в одном направлении. Только выходы являются вершинами, поэтому в иллюстрации всего 6 вершин – N10, N11, N16, N19, N22, N23 (рис. 5).

Файл временных ограничений приведен на рис. 6, фрагмент файла библиотеки стандартных ячеек приведен на рис. 7. Поскольку схема C17 построена на шести элементах nand2, то в библиотеке sky130_fd_sc_hd__tt_025C_1v80.lib нужно смотреть таблицы характеристики для ячейки sky130_fd_sc_hd__nand2_2. Так как значения фронта и нагрузочной емкости, указанные в файле временных ограничений, совпадают со значениями сетки характеристики, то можно использовать в расчетах табличное значение мощности (то есть значение 0.17571 мВт).

Для значений входных фронтов и нагрузочных емкостей, не совпадающих с узлами сетки, необходимо проводить процедуру интерполяции. Значение пикового тока на MBHNB, полученное по результатам расчетов и деления на напряжение питания (1.8 В), равно 0.293 мА. Значение моделирования пикового тока в Synopsys HSPICE равно 0.313 мА, то есть абсолютная погрешность ускоренной оценки пикового тока для схемы C17 равна 6,4%.

В рамках работы также необходимо было показать, что алгоритм эффективно решает задачи на графах с размерностью в несколько сотен вершин за приемлемое время, поэтому также было проведено тестирование алгоритма для схем из тестового набора ISCAS 85 по параметру времени работы программы (табл. 1). Результаты тестирования показывают, что программа выполняет поиск MBHNB на графах до 4000 вершин достаточно быстро.

```
module c17 (N1,N2,N3,N6,N7,N22,N23);
input N1,N2,N3,N6,N7;
output N22,N23;
wire N10,N11,N16,N19;
nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22, N10, N16);
nand NAND2_6 (N23, N16, N19);
endmodule
```

Рис. 3. Описание схемы C17 в формате Verilog.
Fig. 3. Description of the C17 circuit in Verilog format.

```
mw_gates { n10 n11 n16 n19 n22 n23 }
mw_term 0 vdd
mw_term 1 gnd
mw_term 00 n1 n10
mw_term 111 n1 n10 n3
mw_term 010 n1 n22 n23
mw_term 0111 n1 n3 n6 n22
mw_term 110 n1 n2 n22
mw_term 1111 n1 n10 n19 n7
mw_term 1110 n1 n10 n7 n23
mw_term 1111 n1 n10 n2 n16
mw_term 1110 n1 n10 n2 n23
...
mw_term 111 n16 n19 n23
mw_term 011 n16 n19 n7
mw_term 101 n16 n7 n23
```

Рис. 4. Содержание файла логических ограничений для схемы C17.
Fig. 4. Logic constraint file for the C17 circuit.

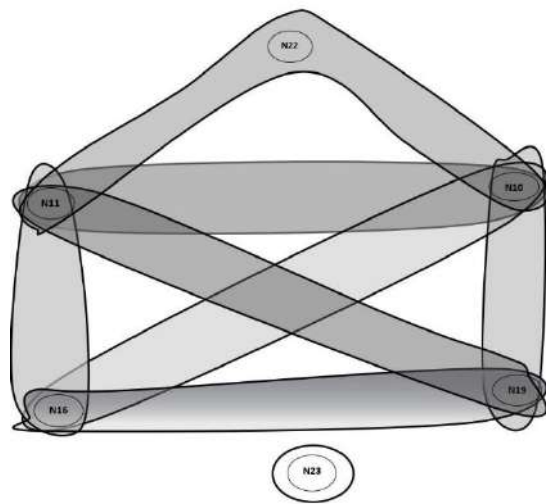


Рис. 5. Гиперграф тройных ограничений схемы C17.
Fig. 5. Triple constraint hypergraph for the C17 circuit.

```
set sdc_version 1.7
set_load 0.102 [get_ports N22]
set_load 0.102 [get_ports N23]
set_input_transition 650
current_design c17
```

Рис. 6. Содержание файла временных ограничений для схемы C17.
Fig. 6. Timing constraints file for the circuit C17.

```
related_pin : "B";
rise_power ("power_outputs_1") {
index_1("0.0100000000, 0.0230505800, 0.0531329300, 0.1224745000, 0.2823108000,
0.6507428000, 1.5000000000");
index_2("0.0005000000, 0.0014486260, 0.0041970370, 0.0121598800, 0.0352302400,
0.1020709000, 0.2957252000");
values("0.0131740000, 0.0147706000, 0.0193188000, 0.0322856000, 0.0692454000,
0.1761525000, 0.4850417000", \
"0.0126719000, 0.0143115000, 0.0188792000, 0.0319361000, 0.0691396000, 0.1762759000,
0.4854702000", \
"0.0122513000, 0.0138694000, 0.0183436000, 0.0315773000, 0.0688111000, 0.1762427000,
0.4850414000", \
"0.0121634000, 0.0136488000, 0.0180699000, 0.0312011000, 0.0682199000, 0.1757919000,
0.4856277000", \
"0.0122867000, 0.0138020000, 0.0182162000, 0.0310540000, 0.0681211000, 0.1750580000,
0.4836663000", \
"0.0138289000, 0.0153922000, 0.0196239000, 0.0326541000, 0.0691547000, 0.1757100000,
0.4841492000", \
"0.0173075000, 0.0186911000, 0.0229938000, 0.0348008000, 0.0715592000, 0.1778563000,
0.4857931000");
}
```

Рис. 7. Фрагмент библиотеки для определения значений мощности, соответствующих схеме
Fig. 7. Library fragment for determining the power values corresponding to the circuit

Табл. 1. Результаты поиска МВННВ для набора тестовых схем ISCAS 85.

Table 1. Search results for the ISCAS 85 test set.

Название схемы	Кол-во вершин	МВННВ (кол-во вентиляй)	Время работы, мс
C17	6	3	6
C432	205	37	25
C499	277	79	36
C880	471	101	80
C1355	621	154	115
C1908	940	123	132
C2670	1644	193	175
C3540	1743	210	225
C5315	2610	295	310
C7552	3830	328	395

Список литературы / References

- [1]. Wang P., Bohacek S. On the practical complexity of solving the maximum weighted independent set problem for optimal scheduling in wireless networks. Proceedings of the 4th Annual International Conference on Wireless Internet, WICON 2008, Maui, Hawaii, USA, November 17-19, 2008. DOI: 10.1145/1554126.1554145.
- [2]. Dong Y., Goldberg A. V., Noe A., Parotsidis N., Resende M. G. C., Spaen Q. New Instances for Maximum Weight Independent Set from a Vehicle Routing Application. Operations Research Forum, 2021, vol. 2. DOI: 10.1007/s43069-021-00084-x.
- [3]. Borowitz J., Großmann E., Schulz Ch. Optimal Neighborhood Exploration for Dynamic Independent Sets. 2024. 10.48550/arXiv.2407.06912.
- [4]. 1. Zhou L., Wang M., Chen Y. Graph-Based Methods for Simultaneous Switching Noise Analysis and Peak Current Estimation in CMOS Circuits // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023, Vol. 42, No. 4, pp. 789–798.
- [5]. Гаврилов С. В., Рыжова Д. И. Алгоритм оценки пикового тока на логическом уровне проектирования на основе анализа распространения логических корреляций в схеме. Вестник Рязанского государственного радиотехнического университета, 2015, выпуск 52, № 2, стр. 56-64. / Gavrilov S. V., Ryzhova D. I. An algorithm for estimating peak current at the logic level of design based on the analysis of the propagation of logical correlations in the circuit. Vestnik of Ryazan State Radio Engineering University, 2015, vol. 52, no. 2, pp. 56-64 (in Russian).
- [6]. Гаврилов С. В. Методы анализа логических корреляций для САПР цифровых КМОП СБИС. М.: РИЦ Техносфера, 2011, 136 с. / Gavrilov S. V. Methods for Logical Correlation Analysis for Digital CMOS VLSI Design. Moscow, Technosfera Research Center, 2011, 136 p. (in Russian).
- [7]. Korhonen T. Lower Bounds on Dynamic Programming for Maximum Weight Independent Set. Available at: <https://arxiv.org/pdf/2102.06901>, accessed 08.11.2025.
- [8]. Andrade D. V., Resende, M. G. C., Werneck R. F. Fast local search for the maximum independent set problem. Journal of Heuristics, 2012, vol. 18, pp. 525-547. DOI: 10.1007/s10732-012-9196-4.

Информация об авторах / Information about authors

Дарья Игоревна РЫЖОВА — кандидат технических наук, начальник научно-исследовательской лаборатории “Перспективные САПР СБИС” института Интегральной

электроники НИУ МИЭТ, лаборант-исследователь НИЦ “Курчатовский институт”. Сфера научных интересов: автоматизированные системы проектирования, логический и логико-временной анализ схем, ускоренный анализ пикового тока КМОП схем, характеристика интегральных схем, физический синтез микро- и нанoeлектронных схем.

Daria Igorevna RYZHOVA – Cand. Sci. (Tech.), head of the Research laboratory “Advanced VLSI CAD” at the Institute of Integrated Electronics, National Research University of Electronic Technology – MIET, research assistant at the NRC “Kurchatov Institute”. Research interests: automated design systems, logical and timing analysis of circuits, accelerated peak current analysis of CMOS circuits, integrated circuit characterization, physical synthesis of micro- and nanoelectronics.

Владислав Игоревич КАСАТКИН является инженером научно-исследовательской лаборатории “Перспективные САПР СБИС” института Интегральной электроники НИУ МИЭТ. Его научные интересы включают разработку алгоритмов автоматизированного проектирования интегральных микросхем.

Vladislav Igorevich KASATKIN is an engineer of the Research laboratory “Advanced VLSI CAD” at the Institute of Integrated Electronics, National Research University of Electronic Technology – MIET. His research interests include the development of algorithms for the automated design of integrated circuits.

DOI: 10.15514/ISPRAS-2026-38(1)-2



Учет временных ограничений на этапе размещения элементов на ПЛИС

Д.Б. Шокарев, ORCID: 0009-0009-2806-1570 <Shokarev_DB@nrcki.ru>

Р.Ж. Чочаев, ORCID: 0009-0007-0304-0496 <Chochaev_RZh@nrcki.ru>

С.В. Гаврилов, ORCID: 0000-0003-0566-4482 <s00v@yandex.ru>

*НИЦ «Курчатовский институт»,
123182 Россия, Москва, пл. Академика Курчатова, д. 1.*

Аннотация. Данная статья посвящена разработке метода учета временных ограничений при оптимизации быстродействия схем на этапе размещения элементов на программируемых логических интегральных схемах (ПЛИС). Поскольку ресурсы ПЛИС ограничены, ключевой задачей разработки систем автоматизированного проектирования (САПР) для ПЛИС является оптимизация эффективности использования этих ресурсов. Решение этой задачи подразумевает разработку новых алгоритмов, позволяющих учитывать архитектурные особенности проектируемых схем. В данной работе рассматриваются основные проблемы, связанные с задачей размещения на ПЛИС с оптимизацией задержек, и существующие подходы к их решению. В данной работе предложен новый метод оценки критичности соединений для оптимизации проектируемых схем на этапе размещения с учетом пользовательских временных ограничений. Представлен алгоритм размещения на основе предложенного метода, а также результаты анализа его эффективности. Эксперименты показали, что применение предложенного метода позволяет сократить задержки критических путей в схеме на 4,32% относительно классического подхода к оценке критичности соединений.

Ключевые слова: СБИС; ПЛИС; топологический синтез; размещение.

Для цитирования: Шокарев Д.Б., Чочаев Р.Ж., Гаврилов С.В. Учет временных ограничений на этапе размещения элементов на ПЛИС. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 17–32. DOI: 10.15514/ISPRAS-2026-38(1)-2.

Благодарности: Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт».

Timing Constraints during FPGA Placement

D.B. Shokarev, ORCID: 0009-0009-2806-1570 <Shokarev_DB@nrcki.ru>
R.Z. Chochaev, ORCID: 0009-0007-0304-0496 <Chochaev_RZh@nrcki.ru>
S.V. Gavrilov, ORCID: 0000-0003-0566-4482 <s00v@yandex.ru>

*NRC «Kurchatov institute»,
1, Kurchatov Square, Moscow, 123182, Russia.*

Abstract. In this paper, we propose a criticality estimation method that takes into account timing constraints during the delay optimization in the FPGA placement. Since FPGA resources are limited, the optimization of the utilization efficiency of these resources is a major challenge in FPGA CAD design. It requires the development of new algorithms that take into account the architectural features of the customers designs. In this paper, existing approaches to the FPGA timing-driven placement are shown. A new method for the criticality estimation considering user-defined timing constraints and a new timing-driven placement algorithm based on this method are proposed. Experimental results show that the proposed method allows to reduce critical path delays in the circuits by 4.32% compared to the classical method.

Keywords: VLSI; FPGA; layout synthesis; placement.

For citation: Shokarev D.B., Chochaev R.Z., Gavrilov S.V. Timing constraints during FPGA placement. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 17-32 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-2.

Acknowledgements: The work was carried out within the state assignment of NRC «Kurchatov institute».

1. Введение

В течение последних лет российская электронная промышленность сталкивается с необходимостью импортозамещения широкого спектра микросэлектронных компонентов, одним из которых являются программируемые логические интегральные схемы (ПЛИС). Устройства данного класса позволяют эффективно решать задачи прототипирования и разработки некоторых типов электронных устройств благодаря возможности программирования логики непосредственно пользователем. В настоящее время маршрут проектирования в базе ПЛИС полностью автоматизирован и требует использования специализированных систем автоматизированного проектирования (САПР).

Для обеспечения максимальной эффективности использования ограниченных ресурсов ПЛИС в современных САПР применяются специализированные алгоритмы оптимизации, адаптированные для архитектур поддерживаемых схем. Однако алгоритмы, применяемые в коммерческих САПР, недоступны, тогда как открытые алгоритмы не адаптированы для эффективной работы с архитектурными особенностями российских ПЛИС. Поэтому возникает необходимость разработки новых методов и алгоритмов для оптимизации различных характеристик проектируемых схем, например, быстродействия.

Одним из ключевых этапов в маршруте проектирования на ПЛИС является этап размещения. От результатов данного этапа напрямую зависит как время выполнения последующих этапов, так и качество итогового решения. Для оптимизации быстродействия проектируемых схем размещение элементов необходимо проводить с учетом задержек, минимизируя критические пути. Существуют две основные проблемы, связанные с задачей размещения на ПЛИС с оптимизацией задержек. Первая из них заключается в том, что на этапе размещения отсутствует информация о реальных задержках путей, которая может быть получена только после трассировки. Это вынуждает использовать различные модели задержек, которые на основе информации о размещении элементов на ПЛИС дают адекватную оценку значения задержки. Вторая проблема заключается в необходимости многократного проведения временного анализа схемы в процессе проектирования для оценки изменения качества размещения. Это требует значительных затрат времени и накладывает ограничения на инструменты временного анализа.

Помимо этого, классические общедоступные алгоритмы оптимизации задержек на этапе размещения на ПЛИС не позволяют в полной мере учитывать пользовательские временные ограничения. В основном, они направлены на оптимизацию комбинаций общих параметров, например, минимизацию длины межсоединений и задержки пути. К данному классу можно отнести алгоритмы, основанные на методе оценки критичности цепей, представленном в VPR [1]. Недостаток данного подхода заключается в том, что вместо реально необходимых пользователю критериев используются обобщенные, не позволяющие во многих случаях получить оптимальное решение. С такой ситуацией можно столкнуться при работе со схемами с несколькими тактовыми сигналами, в том числе локальными, использовании блоков управления тактовыми сигналами (англ. clock gating), использовании ПЛИС как блока в составе устройства, что налагает различные временные ограничения на разные выходы схемы. Таким образом, для оптимизации быстродействия проектируемых схем в базе ПЛИС необходима разработка новых методов и алгоритмов, позволяющих проектировать схемы в соответствии с пользовательскими временными ограничениями.

В данной работе представлен метод учета временных ограничений на этапе размещения элементов на ПЛИС. Также описана реализация алгоритма размещения на его основе, интегрированного в маршрут проектирования в САПР X-CAD [2]. Содержание работы организовано следующим образом. Во втором разделе рассматриваются существующие подходы к решению задачи размещения на ПЛИС с оптимизацией задержек. Третий раздел посвящен разработанному методу учета временных ограничений при анализе быстродействия на этапе размещения на ПЛИС. В нем представлен алгоритм размещения на основе представленного метода и приведена оценка его эффективности относительно реализации на основе классического подхода к оценке критичности.

2. Задача размещения на ПЛИС с оптимизацией задержек

Этап размещения элементов в маршруте проектирования на ПЛИС играет определяющую роль в формировании конечных характеристик разрабатываемого устройства: занимаемой площади, задержек межсоединений, помехоустойчивости. На данном этапе на основе синтезированного описания схемы и геометрического описания блока ПЛИС определяются позиции всех элементов проектируемой схемы, исходя из которых, в дальнейшем проводится трассировка. Поэтому качество итогового размещения можно точно оценить исключительно после проведения трассировки и анализа конечных характеристик. От качества размещения зависят как сложность трассировки, так и предельные значения характеристик схемы. Таким образом, задача размещения заключается в поиске такой конфигурации элементов, которая позволит добиться наилучших характеристик схемы и упростить последующую трассировку. Для того, чтобы добиться одновременного удовлетворения обоих требований, применяемые алгоритмы размещения должны учитывать следующие ограничения:

1. Архитектурные ограничения ПЛИС. К таким ограничениям относятся: количество элементов в группах блоков ПЛИС для иерархических схем, число доступных шин данных и синхронизации, топология соединений в случае отсутствия возможности оттрассировать любое теоретически возможное соединение.
2. Налагаемые пользователем ограничения. К таким можно отнести требования к занимаемой площади, рабочей частоте или помехоустойчивости.

Так как предельная рабочая частота конечного устройства определена заранее, одной из основных целей проектирования становится достижение данной частоты. Для этого в процессе проектирования необходимо многократно проводить оценку задержек критических путей, чтобы удостовериться в удовлетворении заданных пользователем временных ограничений. Проблема в том, что на этапе размещения отсутствует информация о задержках соединений между элементами. Поэтому для оценки задержек путей приходится применять различные приближенные модели, позволяющие получить значение задержки, зная только позиции элементов.

2.1 Статический временной анализ

Все существующие методы временного анализа подразделяются на динамические и статические [3]. Динамические методы основаны на моделировании работы всей схемы, поэтому не подходят для использования на этапе размещения, где основным критерием выбора инструмента временного анализа является минимизация временных затрат на проведение анализа. Поэтому методы статического временного анализа (СВА), в которых рассматриваются изолированные переключения отдельных элементов, становятся единственными применимыми для использования в алгоритмах оптимизации быстродействия проектируемых схем.

Методология СВА основана на построении временного графа, представляющего собой направленный взвешенный граф, отражающий структуру блока комбинационной логики синхронной СБИС. Вершины графа соответствуют элементам схемы, тогда как ребра – цепям между ними. Графы такого типа просты и наглядны, но ввиду малого количества узлов требуют использования упрощенных оценок задержек отдельных элементов. Альтернативный подход предполагает, что вершинам графа соответствуют не элементы, а их входные и выходные контакты, тогда как ребра отражают связи между ними. Именно данный подход применяется при построении временного графа в используемом в данной работе инструменте СВА OpenSTA [4].

Процесс работы с временным графом включает в себя четыре основных шага. На первом шаге для каждого комбинационного блока схемы на основе информации из библиотеки стандартных ячеек в формате Liberty происходит расчет задержек всех ребер и длительности переключений всех сигналов в каждой из вершин. После этого определяется фактическое время прибытия сигнала (англ. Actual Arrival Time, AAT), то есть временной промежуток, за который сигнал проходит путь от входа до выбранной точки. На третьем шаге, исходя из установленных временных ограничений, для каждой вершины задается требуемое время прибытия сигнала (англ. Required Arrival Time, RAT), представляющее собой максимальное допустимое время, при соответствии которому гарантируется правильное функционирование схемы. В конце вычисляется разница между требуемым и фактическим временем, представляющая собой запас по времени (англ. Slack). Отрицательное значение запаса по времени свидетельствует о нарушении временных ограничений.

Существует два подхода к проведению СВА:

1. На основе анализа путей. Данный подход подразумевает применение алгоритмов обхода графа в глубину. Сначала определяется анализируемая группа путей, после чего для каждого из них вычисляется общая задержка. Недостаток данного подхода заключается в том, что с увеличением размера схемы скорость анализа резко снижается, поскольку общее число путей увеличивается экспоненциально.
2. На основе блочного анализа. В отличие от подхода на основе анализа путей, данный подход подразумевает применение алгоритмов обхода графа в ширину. Их преимущество заключается в линейной вычислительной сложности относительно размера схемы, однако они не способны учитывать корреляции между различными путями, например, между тактовыми путями и путями данных.

В используемом в данной работе инструменте СВА OpenSTA совмещаются оба подхода. Сперва проводится блочный анализ, позволяющий определить критичность путей. Однако, в отличие от классического подхода, информация о задержках на некритических путях не удаляется, а привязывается к сигналу. Так формируются группы из одного сигнала и связанных с ним путей. После этого становится возможным провести детальный анализ выбранных путей, но без существенной потери производительности, так как нет необходимости перебирать все возможные переключения.

2.2 Модели задержек

Фундаментальной проблемой оптимизации задержек на этапе размещения является то, что точные значения задержек можно определить только после завершения этапа трассировки. Вследствие этого возникает необходимость получения приблизительных величин задержек, обеспечивающих достаточную точность. Для простых гомогенных архитектур ПЛИС, не имеющих иерархии, приемлемым решением является замена реальных задержек линейной функцией от расстояния между элементами. Тем не менее, введение иерархии приводит к тому, что такая модель дает неадекватные результаты, поскольку не способна учесть архитектурные особенности ПЛИС. В таком случае задержка будет зависеть не только от физического расстояния между логическими блоками, но и от принадлежности их различным группам блоков ПЛИС, специфики используемых элементов и особенностей реализации архитектуры конкретного кристалла.

Одним из способов преодоления указанной проблемы является разделение общей задержки на составляющие компоненты, каждый из которых рассчитывается отдельно. Например, в работе [5] предлагается следующая формула расчета задержки $D(i, j)$ ребра (i, j) между двумя логическими блоками:

$$D(i, j) = I(i, j) + P(i, j) + C(i, j),$$

где $I(i, j)$ – задержка между выходным контактом источника цепи и первым коммутационным блоком ребра, $C(i, j)$ – компенсирующая задержка, позволяющая учитывать различие задержек различных функциональных блоков, $P(i, j)$ – задержка пути, вычисляемая следующим образом:

$$P(i, j) = B(i, j) + L(i, j),$$

где $B(i, j)$ – базовая задержка, $L(i, j)$ – задержка, зависящая от количества задействованных фрагментов цепей ПЛИС, вычисляемая по следующей формуле:

$$L(i, j) = h \times \text{WireDelayH} + v \times \text{WireDelayV},$$

где WireDelayH и WireDelayV – задержки горизонтальных и вертикальных фрагментов цепей ПЛИС, h и v – количество задействованных фрагментов цепей ПЛИС по горизонтали и вертикали соответственно.

Другая реализация данного подхода была предложена в работе [6]. Предлагаемое решение заключается в замене пути эквивалентной электрической схемой и определении элморовских задержек для полученных цепочек транзисторов. Такое решение должно дать наиболее близкий к реальным значениям результат, однако требует, по сути, характеристики путей в ПЛИС, что является чрезвычайно трудоемким подготовительным процессом, который должен быть повторен для каждой новой ПЛИС.

Общим недостатком подобных моделей является то, что предсказанные ими значения будут соответствовать действительности только в том случае, если инструмент трассировки также направлен на минимизацию задержек. В противном случае оптимальные пути, полученные на этапе трассировки, могут отличаться от тех, которые ожидались на этапе размещения. Тогда, даже имея реальную информацию о задержках, может не получиться добиться наилучших характеристик схемы.

Другим типом моделей, которые уже много лет находят применение в различных инструментах, являются эвристические. Это могут быть как сложные обобщенные модели с множеством параметров, так и узкоспециализированные варианты. К первым можно отнести модель, представленную в работе [7], которая учитывает не только горизонтальное и вертикальное расстояние между элементами, но и размер схемы, перегруженность и количество контактов цепи. Примером вторых может служить модель, используемая в открытой САПР VPR [1]. В ней значения задержек получены экспериментальным путем по

результатам работы трассировщика, используемого в САПР. Такой подход позволяет одновременно учесть как все архитектурные особенности, так и особенности используемого инструмента трассировки. Однако из-за привязанности модели задержек к конкретному трассировщику изменение алгоритма трассировки приведет к необходимости времязатратного перерасчета всех значений.

Ввиду простоты реализации для использования с целевой ПЛИС была выбрана эвристическая модель задержки VPR. Из-за архитектурных особенностей целевой ПЛИС задержка между выходным контактом источника цепи и первым коммутационным блоком любого ребра временного графа всегда одинакова, поэтому была включена в задержку самих элементов. Так были выделены следующие четыре типа связи между элементами, для которых необходимо получить задержки:

1. Между элементами одной группы блоков ПЛИС;
2. Между элементами различных групп блоков ПЛИС;
3. Между входным блоком ПЛИС и элементом группы блоков ПЛИС;
4. Между выходным блоком ПЛИС и элементом группы блоков ПЛИС.

Для каждого типа связи была составлена своя таблица задержек в зависимости от расстояния между элементами по горизонтали и вертикали. Эти таблицы построены на основе статистической информации, для получения которой проводилось несколько сотен запусков трассировки с заданным заранее размещением для перебора всех возможных взаиморасположений элементов. Сперва происходила генерация файлов размещения для всех запусков, а после проводилась трассировка и сбор информации о задержках путей. Для 1 и 2 типов связи использовалась схема, включающая в себя два инвертора, а для 3 и 4 – один. Это позволило исключить возможное влияние других цепей на результат трассировки.

2.3 Учет временных ограничений на этапе размещения

Одним из основных классов алгоритмов размещения являются эвристические алгоритмы. Такие алгоритмы принимают на вход предварительно сформированное начальное размещение и оптимизируют его, последовательно приближая к наиболее оптимальному варианту. В качестве критерия оптимальности служит целевая функция, на минимизацию которой и направлен алгоритм. Но, хотя алгоритмы данного класса и позволяют достигать глобального оптимума целевой функции, при увеличении размеров проектируемых схем скорость поиска решения значительно падает ввиду резко возрастающего числа потенциальных перестановок.

Одним из классических эвристических алгоритмов размещения элементов на ПЛИС является алгоритм имитации отжига [8]. На его основе разработан алгоритм размещения, используемый в открытой САПР VPR. Для работы алгоритма проектируемая схема представляется в виде ориентированного графа $G = (V, E)$, где V – множество вершин, представляющих элементы, а E – множество направленных дуг, представляющих связи между двумя контактами элементов. Каждой дуге присваиваются два значения: вес, соответствующий длине соединения, и вес, связанный с задержкой ребра. Задача алгоритма размещения VPR сводится к минимизации суммы этих двух величин.

Изменение целевой функции ΔC при перестановке элементов вычисляется следующим образом [1]:

$$\Delta C = \lambda \cdot \frac{\Delta C_T}{Prev_C_T} + (1 - \lambda) \cdot \frac{\Delta C_W}{Prev_C_W},$$

где C_T – стоимость задержек, C_W – стоимость длины цепей, λ – балансирующий коэффициент от 0 до 1, для задания степени влияния задержек, $Prev_C_T$ и $Prev_C_W$ – значения C_T и C_W на предыдущей итерации температуры.

Стоимость длины цепей вычисляется следующим образом:

$$C_W = \sum_{i=1}^{N_{nets}} q[i] \times (bb_x(i) + bb_y(i)),$$

где N_{nets} – общее количество цепей в схеме, $bb_x(i)$, $bb_y(i)$ – размеры охватывающего прямоугольника для i -ой цепи, $q[i]$ – коэффициент для компенсации снижения оценки длины цепи из-за использования модели охватывающего прямоугольника.

Для того, чтобы определить, задержки каких ребер следует оптимизировать в первую очередь, вводится понятие критичности ребра, которая помогает оценить степень влияния каждого соединения на задержку критического пути. Минимизация задержек наиболее критичных ребер является ключевым фактором сокращения длины критического пути всей схемы.

Стоимость задержек с учетом критичности ребер вычисляется по следующим формулам:

$$C_T = \sum_{\forall i,j \in circuit} C_T(i,j),$$

$$C_T(i,j) = Delay(i,j) \cdot Crit(i,j)^\beta,$$

где $C_T(i,j)$ – это стоимость задержки ребра (i,j) , $Delay(i,j)$ – задержка ребра (i,j) , $Crit(i,j)$ – критичность ребра (i,j) , β – степенной коэффициент.

Коэффициент β вводится для того, чтобы увеличить вес критических цепей. Во время работы алгоритма данный коэффициент постепенно возрастает от начального значения 1 до максимального значения, установленного пользователем, которое по умолчанию составляет 8. Дальнейшее увеличение перестает оказывать влияние на итоговый результат.

2.4 Классический подход к расчету критичности

Классический подход к расчету критичности заключается в определении соотношения между запасом по времени для ребер и максимальной задержкой пути в схеме. Это позволяет количественно оценить, насколько задержка конкретных ребер близка к тому, чтобы ограничивать быстродействие схемы. Классическая формула расчета критичности выглядит следующим образом [1]:

$$Crit(i,j) = 1 - \frac{Slack(i,j)}{D_{max}},$$

где $Slack(i,j)$ – запас по времени для ребра (i,j) , D_{max} – максимальная задержка пути в схеме.

При использовании данной формулы расчета критичности ребер временного графа для всех путей должно быть установлено единое требуемое время прибытия сигнала, равное D_{max} для всех маршрутов в проекте. Таким образом, для всех ребер, принадлежащих путям с максимальным временем распространения сигнала, величина запаса времени $Slack(i,j)$ будет равна нулю, в то время как критичность принимает максимальное значение 1. То есть, чем ближе задержка пути к максимальной задержке в схеме, тем выше критичность всех ребер на этом пути. Такой подход эффективен для уменьшения общей задержки критического пути, однако в реальных проектах путь с максимальной задержкой не всегда оказывается критическим. Фактическая критичность путей определяется временными ограничениями, накладываемыми проектировщиком. Нередко бывает так, что требования по времени прибытия сигнала варьируются в зависимости от участка схемы. Например, в случаях, когда используются локальные тактовые сигналы или несколько глобальных, или если максимально допустимые задержки на различных выходах устройства неодинаковы и другие подобные ограничения.

В качестве примера подобной ситуации рассмотрим тестовую схему, изображенную на рис. 1. Она состоит из двух триггеров (FF1 и FF2) и трех комбинационных элементов (LE1, LE2, LE3) с задержками 4 нс, 2 нс и 2 нс соответственно. Все ребра схемы изначально имеют одинаковые задержки, равные 1 нс. Временные ограничения выглядят следующим образом:

- Максимальная задержка на выходе out – 8 нс;
- Период синхросигнала clk – 6,5 нс;
- Время предустановки триггеров – 0,5 нс.

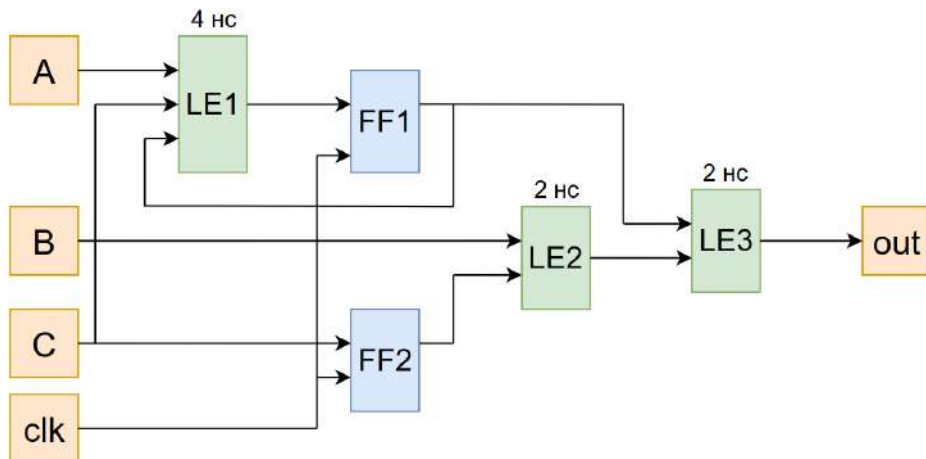


Рис. 1. Тестовая схема.

Fig. 1. Test circuit.

Для данной схемы был построен временной граф и вычислены ААТ, РАТ и запас по времени. На рис. 2 показаны два временных графа: используемый при расчете критичности ребер и получаемый при СВА с учетом заданных ограничений. На каждом из графов красным цветом выделен критический путь.

Как можно заметить, в соответствии с классической функцией критичности, веса, присваиваемые ребрам, пропорциональны задержкам соответствующих путей, однако пользовательские временные ограничения полностью игнорируются. Вследствие этого при проведении СВА с учетом вышеуказанных ограничений критические пути отличаются, причем даже если пути будут совпадать, то реальная критичность ребер может существенно отличаться от вычисляемой с помощью классической функции. Поэтому применение классической функции критичности на схемах со сложной структурой может привести к минимизации задержек не критических путей и, как следствие, к получению неоптимального результата размещения.

3. Метод учета временных ограничений на этапе размещения

Для того, чтобы решить проблему учета пользовательских временных ограничений, предлагается объединять пути с аналогичными временными ограничениями в группы для последующего анализа. Основная идея заключается в следующем: пути объединяются в группы таким образом, чтобы оценка критичности производилась исключительно в пределах каждой отдельной группы, исключая некорректные сравнения запасов по времени, относящихся к различным требуемым временам прибытия сигнала.

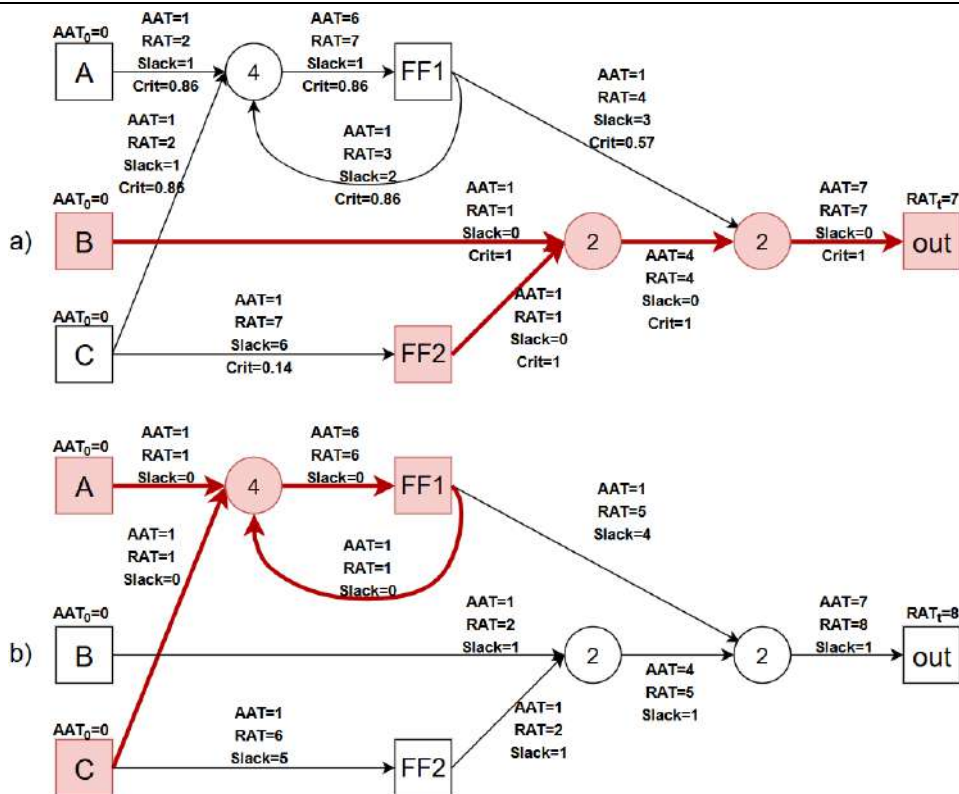


Рис. 2. (a) Пример расчета критичности по классической функции для тестовой схемы, (b) СВА с учетом заданных пользователем ограничений. RAT_t – требуемое время прибытия в терминальной точке t пути, а AAT_0 – фактическое время прибытия в начальной точке пути.

Fig. 2. (a) Example of calculating criticality using the classical function for the test circuit, (b) STA taking into account user-specified constraints. RAT_t – required arrival time at the terminal point t of the path, AAT_0 – actual arrival time at the starting point of the path.

Формирование групп происходит согласно следующим двум критериям:

1. Источник ограничения – временное ограничение, определяющее требуемое время прибытия сигнала для пути. К источникам ограничений относятся период синхросигнала, а также максимально допустимая задержка на выходе схемы.
2. Источник сигнала – элемент или вход, выступающий в роли источника рассматриваемого сигнала. Им может являться входной контакт схемы или выход триггера.

Однако, если учет источника ограничения критически необходим, то деление в зависимости от источника сигнала не столь важно. Деление было введено, чтобы, используя упомянутую в разделе 2.1 особенность OpenSTA по привязке путей к сигналам, повысить быстродействие алгоритма размещения. Рассмотрим применение данного подхода на примере рассмотренной ранее тестовой схемы. Данная схема содержит семь путей, которые могут быть разделены на четыре группы, представленные в табл. 1 и на рис. 3.

Исходя из представленного деления на группы, итоговая критичность ребер формируется из двух составляющих:

- Критичности ребра в пределах собственной группы;
- Критичности группы относительно других групп.

Табл. 1. Группы путей в тестовой схеме.

Table 1. Path groups in the test circuit.

№	Путь	Источник ограничения	Источник сигнала
1	A → LE1 → FF1 B → LE1 → FF1 C → FF2	Период синхросигнала clk	Первичный вход
2	FF1 → LE1 → FF1	Период синхросигнала clk	Триггер, тактируемый сигналом clk
3	B → LE2 → LE3 → out	Первичный выход out	Первичный вход
4	FF1 → LE3 → out FF2 → LE2 → LE3 → out	Первичный выход out	Триггер, тактируемый сигналом clk

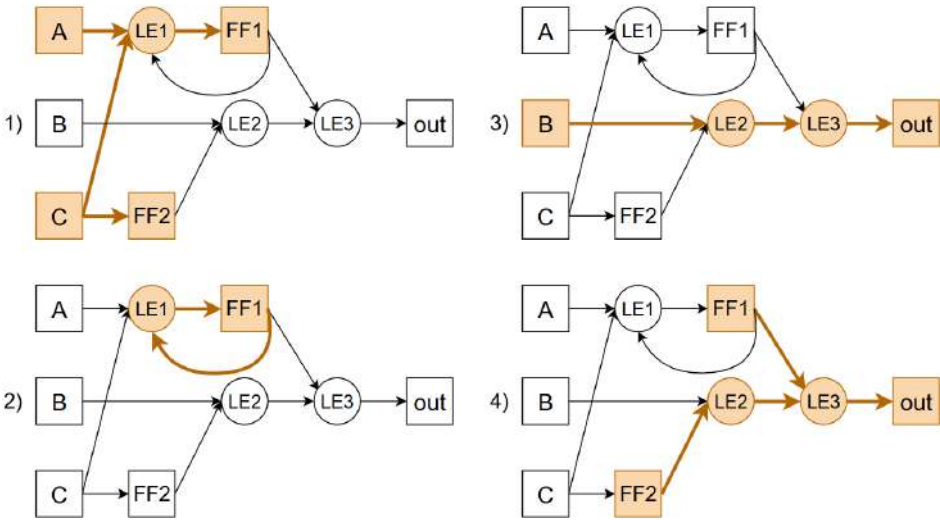


Рис. 3. Группы путей в тестовой схеме.

Fig. 3. Path groups in the test circuit.

С учетом этого, предлагается новая функция оценки критичности для ребра (i, j) :

$$Crit(i, j) = PgCrit_g \cdot \left(1 - \frac{Slack(i, j) - PgMinSlack_g}{PgMinRAT_g - PgMinSlack_g} \right),$$
$$PgCrit_g = \frac{1}{PgMinSlack_g - GlobalMinSlack + 1'}$$

где g – индекс наиболее критической группы, в которую входит ребро, $Slack(i, j)$ – запас по времени для ребра, рассчитанный для сигнала и ограничения группы g , $PgCrit_g$ – критичность группы g , $PgMinSlack_g$ – минимальный запас по времени в группе g , $PgMinRAT_g$ – минимальное требуемое время прибытия в терминальных точках путей в группе g , $GlobalMinSlack$ – минимальный запас по времени во всей схеме.

Так как через одно ребро может проходить множество путей, из-за чего оно может принадлежать одновременно нескольким группам, при расчете критичности используется наибольшее значение критичности среди всех групп, к которым оно принадлежит. Величина критичности группы путей лежит в интервале $[0, 1]$ и возрастает по мере уменьшения минимального запаса времени в группе. Общая критичность ребра получается путем

умножения критичности наиболее критичной группы, включающей это ребро, на критичность ребра в пределах данной группы. По сути, данный метод расширяет область применимости классической функции, позволяя сравнивать временные характеристики изолированных фрагментов схемы посредством введения дополнительного уровня иерархии. В качестве примера применения разработанной функции рассмотрим представленную ранее тестовую схему. Временные ограничения остаются неизменными:

- Максимальная задержка на выходе out – 8 нс;
- Период синхросигнала clk – 6,5 нс;
- Время предустановки триггеров – 0,5 нс.

На рис. 4 представлен временной граф, для которого был произведен расчет критичности с применением разработанной функции. Рассмотрим ребро, ведущее к вершине FF1, соответствующей триггеру. Согласно классической функции (рис. 2, а), критичность данного ребра составляет 0,86, поскольку она принадлежит пути с задержкой 6 наносекунд при максимальной задержке схемы 7 наносекунд. Тем не менее при выполнении СВА с учетом временных ограничений оказывается, что запас по времени для рассматриваемого ребра минимален в схеме и равен 0. Поэтому, согласно разработанной функции, его критичность принимается равной единице. Этот пример показывает преимущество предлагаемого метода перед классическим, поскольку позволяет определять степень критичности ребер с учетом пользовательских временных ограничений.

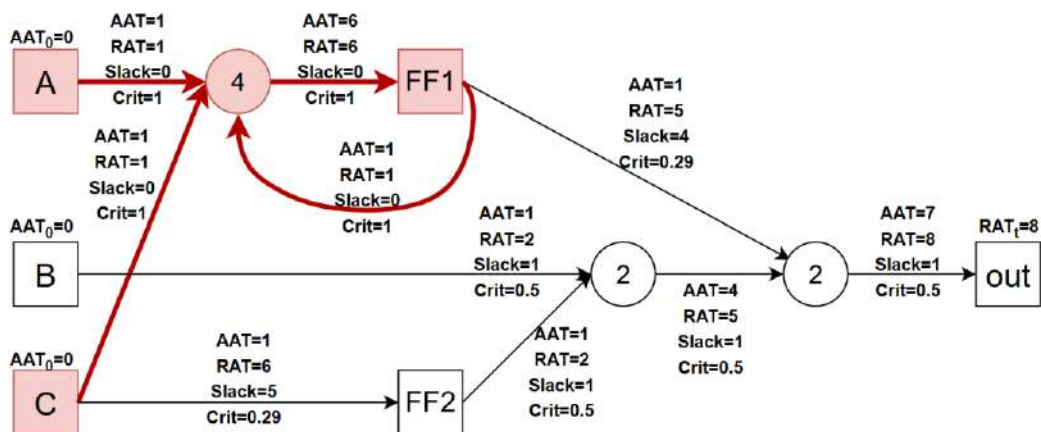


Рис. 4. Расчет критичности по разработанной функции для тестовой схемы.
Fig. 4. Calculation of criticality based on the developed function for the test circuit.

3.1 Алгоритм размещения на основе метода группировки путей

Для оценки эффективности предложенного метода учета временных ограничений разработан алгоритм размещения с оптимизацией задержек. Данный алгоритм относится к классу эвристических и базируется на рассмотренном ранее алгоритме размещения VPR. Блок-схема данного алгоритма представлена на рис. 5. Задача разработанного алгоритма состоит в минимизации следующей целевой функции:

$$C = \lambda \cdot \frac{C_T}{\text{Min_}C_T} + (1 - \lambda) \cdot \frac{C_W}{\text{Min_}C_W},$$

где C_T – стоимость задержек, C_W – стоимость длины цепей, λ – балансирующий коэффициент от 0 до 1, для задания степени влияния задержек, $\text{Min_}C_T$ и $\text{Min_}C_W$ – минимальные значения C_T и C_W за все время работы алгоритма.

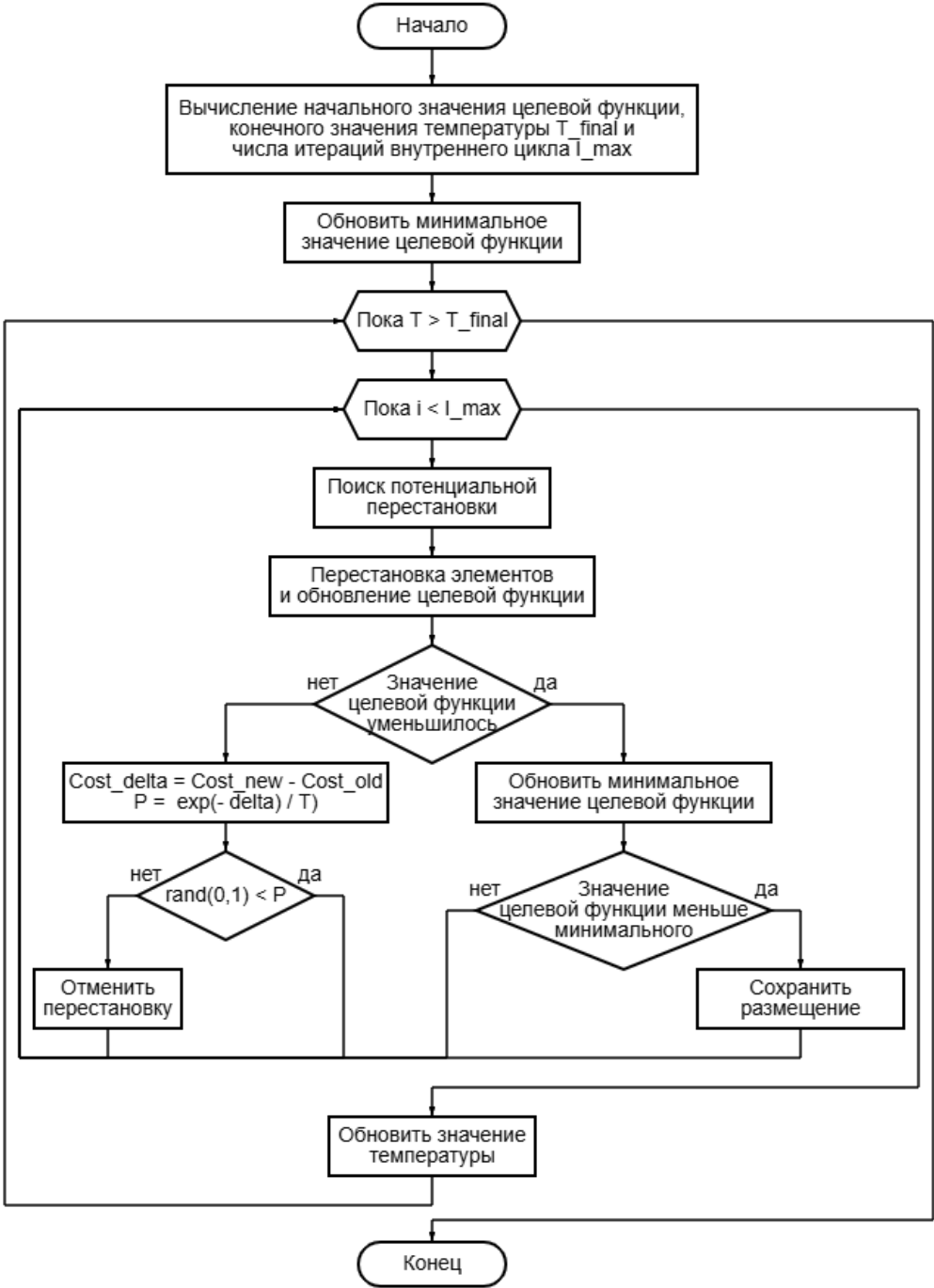


Рис. 5. Блок-схема алгоритма размещения на ПЛИС с учетом временных ограничений.

Fig. 5. Diagram of the FPGA placement algorithm taking into account timing constraints.

Функции стоимости совпадают с применяемыми в VPR, тогда как для определения задержек используются упомянутые в разделе 2.2 таблицы задержек, составленные для целевой ПЛИС.

Помимо этого, в реализации предусмотрена возможность выбора используемой функции критичности, что позволяет избежать влияния иных факторов при сравнении методов.

В качестве инструмента СВА используется OpenSTA, описанный в разделе 2.1. Данный инструмент может быть использован в качестве подключаемой библиотеки на языке C++ и встроен как часть САПР. В качестве отдельной программы OpenSTA обладает собственным Tcl интерфейсом для управления работой и загрузки временных ограничений в формате Synopsys Design Constraints (SDC) [9]. Однако при работе с ним как с библиотекой данный интерфейс отсутствует, поэтому необходима разработка собственного C++ интерфейса для работы с ним. Таким образом, для реализации алгоритма размещения с оптимизацией задержек было сделано следующее:

- инструмент СВА OpenSTA собран в виде C++ библиотеки, реализован программный интерфейс, обеспечивающий вызов функций инструмента, передачу данных из OpenSTA и загрузку временных ограничений;
- разработана реализация алгоритма размещения, в которой OpenSTA используется для задания задержек, обновления временного графа и получения временных характеристик схемы.

Поскольку любая перестановка элементов влияет на задержки путей, проходящих через перемещенные элементы, временные характеристики, полученные в ходе предыдущих расчетов, утрачивают свою актуальность. Поэтому возникает проблема выбора оптимальной частоты проведения СВА. Исследования разработчиков САПР VPR показали, что выполнение СВА чаще, чем изменяется температура, практически не улучшает качество итогового решения, но значительно увеличивает затраты по времени [1]. Эксперименты на целевой ПЛИС показали, что данное утверждение справедливо, и более частое проведение СВА даст выигрыш в среднем до 2%, но многократно увеличивает время работы алгоритма. По этой причине СВА выполняется только при каждом обновлении температуры.

3.2 Оценка эффективности

Реализация разработанного алгоритма размещения с учетом временных ограничений была протестирована на схемах различной размерности, включая схемы из наборов ISCAS-85 [10], ISCAS-89 [11] и IWLS 2005 [12]. Проведен анализ задержек критических путей для трех вариантов размещения:

- без учета временных ограничений ($\lambda=0$);
- с учетом временных ограничений и стандартной функцией критичности;
- с учетом временных ограничений и разработанной функцией критичности.

Для экспериментов с двумя последними вариантами размещения коэффициент λ устанавливался равным 0,7, поскольку именно такое значение позволило получить наилучшие результаты на исследуемом диапазоне значений от 0 до 1 с шагом 0,1. Значения задержек для сравнения реализаций алгоритма были взяты для наиболее критического пути в схеме после прохождения всего маршрута. Результаты тестирования представлены в табл. 2.

Эксперименты показали, что применение разработанной функции критичности, учитывающей пользовательские временные ограничения, обеспечило сокращение длины критического пути примерно в среднем на 5,6%. Тогда как классическая функция дала сокращение только на 1,28%. Вместе с тем в отдельных случаях наблюдается существенный рост задержки критического пути при использовании разработанного метода. Это обусловлено тем, что в используемом трассировщике [2] нет возможности выполнять трассировку цепей с оптимизацией задержек. Поэтому трассировка выполнялась с

оптимизацией длины цепей, что в ряде случаев негативно сказывается на конечных значениях задержек.

Табл. 2. Результаты тестирования.

Table 2. Testing results.

		Без оптими- зации	Оптимизация с классической функцией критичности		Оптимизация с разработанной функцией критичности	
Схема	Кол-во элемен- тов	Задержка, нс	Задержка, нс	Δ, %	Задержка, нс	Δ, %
ehl_spi_ssr	16	4,12	3,93	-4,45	3,64	-11,47
s382	49	8,33	7,49	-10,05	7,51	-9,77
s526	50	6,08	5,51	-9,37	5,41	-11,04
GRAY_24_BIT	54	3,75	3,76	0,08	3,75	0,00
synchronizer	64	4,11	4,11	0,00	4,11	0,00
ehl_uart_rx	74	7,88	8,03	1,95	8,35	6,04
s713	77	18,17	17,77	-2,19	16,33	-10,12
s641	79	17,06	16,69	-2,17	14,75	-13,52
c1355	85	16,64	15,32	-7,90	14,51	-12,77
c499	86	16,23	15,75	-2,96	15,09	-7,01
b13	102	3,85	3,84	-0,09	3,84	-0,09
s832	127	3,69	3,71	0,51	3,69	0,01
s820	128	3,69	3,69	-0,01	3,73	0,83
timer_module	146	8,87	7,89	-11,04	7,57	-14,64
arbiter	164	57,62	56,78	-1,46	56,57	-1,82
b04	190	3,42	3,89	13,71	3,42	0,05
s1423	206	3,80	3,80	-0,09	3,81	0,26
s1238	240	3,73	3,73	-0,22	3,73	-0,18
b05	262	3,78	3,75	-0,88	3,74	-0,98
srio_arb_tx_mux	268	23,94	26,86	12,22	24,48	2,25
s1488	280	3,72	3,72	0,00	3,22	-13,44
simple_spi	304	9,26	8,37	-9,56	8,86	-4,29
c3540	418	28,12	29,72	5,66	24,08	-14,38
fifo	429	19,65	21,27	8,23	21,08	7,27
i2c_master_top	448	13,48	16,00	18,74	12,86	-4,62
b12	514	3,88	3,83	-1,30	3,88	0,07
test_2	633	12,38	9,44	-23,76	9,86	-20,35
info_handler_top	675	17,41	16,47	-5,42	13,91	-20,10
decimator3_1_1	954	38,38	36,38	-5,20	35,03	-8,73
Средние значения:				-1,28		-5,60

4. Заключение

Учет временных ограничений на этапе размещения элементов на ПЛИС позволяет повысить быстродействие проектируемых схем за счет увеличения эффективности использования ограниченных ресурсов ПЛИС. В данной работе были рассмотрены основные проблемы, возникающие при решении задачи размещения на ПЛИС с оптимизацией задержек. Среди них: выбор подходящего инструмента временного анализа и формирование подходящей модели задержек. Также рассмотрены классические подходы к решению задачи размещения и представлен метод группировки путей, позволяющий учитывать пользовательские временные ограничения при оптимизации быстродействия проектируемых схем.

На основе представленного метода разработан алгоритм размещения с оптимизацией задержек, проведено его тестирование и произведена оценка эффективности предложенного метода. Как показали эксперименты, представленный метод позволил сократить длину критических путей в среднем на 5,6%, что на 4,32% больше, чем позволяет достичь классический подход. Так, метод группировки путей расширяет возможности в оценке критичности цепей, тем самым совершенствуя основу, на которой базируется множество существующих алгоритмов. Также, поскольку представленный метод, в отличие от классического подхода, не требует каких-либо вмешательств в работу инструмента СВА или специальных функций, то позволяет использовать любые инструменты СВА, в том числе и открытые.

Список литературы / References

- [1]. Marquardt A., Betz V., Rose J. Timing-driven placement for FPGAs / Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays, 2000, pp. 203-213. DOI: 10.1145/329166.329208.
- [2]. 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 базе гетерогенных ПЛИС. Труды ИСП РАН, том 35, вып. 5, 2023 г., стр. 107–126. DOI: 10.15514/ISPRAS-2023-35(5)-8. / 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.
- [3]. Forzan C., Pandini D. Statistical static timing analysis: A survey / Integration, 2009, T. 42, №. 3, pp. 409-435. DOI: 10.1016/j.vlsi.2008.10.002.
- [4]. Ajayi T., Blaauw D. OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain / Proc. Government Microcircuit Applications and Critical Technology Conference, 2019, pp. 1105-1110.
- [5]. Lin Z., Xie Y., Qian G., Chen J., Wang S., Yu J., Chang Y.W. Timing-driven placement for FPGAs with heterogeneous architectures and clock constraints / 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2021, pp. 1564-1569. DOI: 10.23919/DATE51398.2021.9474054.
- [6]. Hung E., Wilton S. J. E., Yu H., Chau T. C. P., Leong. P. H. W. A detailed delay path model for FPGAs / 2009 International Conference on Field-Programmable Technology, IEEE, 2009, pp. 96-103. DOI: 10.1109/FPT.2009.5377673.
- [7]. Karnik T., Kang S.M. An empirical model for accurate estimation of routing delay in FPGAs / Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), IEEE, 1995, pp. 328-331. DOI: 10.1109/ICCAD.1995.480136.
- [8]. Ghannadi P., Kourehli S.S., Mirjalili S. A review of the application of the simulated annealing algorithm in structural health monitoring (1995-2021) / Frattura ed Integrità Strutturale, 2023, T. 17, № 64, pp. 51-76. DOI: 10.3221/IGF-ESIS.64.04.
- [9]. Using the Synopsys Design Constraints Format Application Note, Version 2.1 / Mountain View, CA, USA, 2017.
- [10]. Brglez F., Fujiwara H. A neutral netlist of 10 combinational benchmark circuits and a target simulator in Fortran / International Symposium on Circuits and Systems, 1985, 1985, pp. 695-698.

- [11]. Brglez F., Bryan D., Kozminski K. Combinational profiles of sequential benchmark circuits / 1989 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 1989, pp. 1929-1934. DOI: 10.1109/ISCAS.1989.100747.
- [12]. Albrecht C. IWLS 2005 benchmarks / International Workshop for Logic Synthesis (IWLS), 2005, т. 9.

Информация об авторах / Information about authors

Дмитрий Борисович ШОКАРЕВ – инженер-исследователь отдела САПР ОППМ ЦПМ НИЦ «Курчатовский институт». Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, РСнК.

Dmitry Borisovich SHOKAREV – Research Engineer of Department of Design Problems in Microelectronics of CPM of NRC «Kurchatov institute». Research interests: electronic design automation, VLSI computer-aided design, FPGA, SoC FPGA.

Рустам Жамболатович ЧОЧАЕВ – научный сотрудник отдела САПР ОППМ ЦПМ НИЦ «Курчатовский институт». Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС, ПЛИС, РСнК.

Rustam Zhambolatovich CHOCHAEV – Research Associate of Department of Design Problems in Microelectronics of CPM of NRC «Kurchatov institute». Research interests: electronic design automation, VLSI computer-aided design, FPGA, SoC FPGA.

Сергей Витальевич ГАВРИЛОВ – доктор технических наук, профессор, руководитель отделения проблем проектирования в микроэлектронике ЦПМ НИЦ «Курчатовский институт». Область научных интересов: автоматизация проектирования микроэлектроники, САПР ИС.

Sergey Vitalievich GAVRILOV – Dr. Sci. (Tech.), Prof., Head of Department of Design Problems in Microelectronics of CPM of NRC «Kurchatov institute». Research interests: electronic design automation, VLSI computer-aided design.



Декомпозиция Ашенхёрста для таблично заданных булевых функций

П.Г. Емельянов, ORCID: 0000-0001-8747-0600 <emelyanov@iis.nsk.su>

Институт систем информатики им. А.П. Еришова СО РАН,
Россия, 630090, г. Новосибирск, пр. Лаврентьева, д. 6.

Новосибирский государственный университет,
Россия, 630090, г. Новосибирск, ул. Пирогова, д. 1.

Аннотация. Декомпозиция сложных систем – широко распространённый метод анализа и преобразований, применяемый в различных областях. Декомпозиция в менее сложные подсистемы, помимо того, что уменьшает в целом представление исходной системы (при этом, возможно, оптимизируются и другие её характеристики), позволяет выявить скрытую структуру системы, что, в частности, позволяет провести её более глубокий анализ и преобразования. В статье рассмотрен подход к декомпозиции Ашенхёрста для таблично заданных булевых функций (это могут быть таблицы истинности, характеристические вектора нулей и единиц и так далее), основанный на интерпретации композиции таких функций как их соединения-с-равенством (аналогично SQL оператору JOIN ON с равенством). Задача декомпозиции сведена к задаче решения с использованием базисов Грёбнера системы квадратных уравнений над конечным полем порядка 2 для некоторых специально введённых параметров. Решение системы индуцирует разбиение входных переменных исходной функции на связанные и свободные переменные, что, в свою очередь, позволяет определить функции-компоненты декомпозиции, при этом сами вычисленные параметры связывают их, выступая для одной в качестве значений результата, а для другой значений аргумента. Приведены результаты экспериментов по декомпозиции в системе компьютерной алгебры Maple. Рассмотрены варианты обобщения данного подхода.

Ключевые слова: декомпозиция Ашенхёрста булевых функций; таблицы истинности; обращение декартова соединения таблиц; обращение соединения-с-равенством; системы квадратных уравнений над конечным полем $GF(2)$; базисы Грёбнера.

Для цитирования: Емельянов П.Г. Декомпозиция Ашенхёрста для таблично заданных булевых функций. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 33–44. DOI: 10.15514/ISPRAS-2026-38(1)-3.

Ashenhurst Decomposition for Table-defined Boolean Functions

P.G. Emelyanov, ORCID: 0000-0001-8747-0600 <emelyanov@iis.nsk.su>

A.P. Ershov Institute of Informatics Systems of the Siberia Branch of the RAS,
6, Lavrentiev ave., Novosibirsk, 630090, Russia.

Novosibirsk State University,
1, Pirogova st., Novosibirsk, 630090, Russia.

Abstract. Decomposition of complex systems is a widely used method of analysis and transformation in various fields. Decomposition of complex systems into smaller ones can reduce the overall complexity of them while potentially optimizing other characteristics. This process also allows for the explication of the hidden structure within the system, facilitating deeper analysis and possible transformations. This article is aimed to establish a new approach to Ashenhurst decomposition for table-defined Boolean functions (truth tables, characteristic vectors of zeros/units, etc.). This approach is based on the idea of considering the composition of these functions in terms of join-on-equality of their tables (similar to the SQL JOIN ON operator equipped with equality of the joined attributes). The decomposition process involves solving a system of quadratic equations for some newly introduced parameters using Gröbner basis techniques over the finite field of the order 2. These equations are derived using a certain condition on a polynomial that is constructed from the given table. Processing the solution of the system results in the splitting of the input variables of the original function into bound and free variables. This, in turn, enables us to determine the individual functions-components of the decomposition. The computed parameters then connect these functions, acting as output values for one component and input values for the other component. The results of decomposition experiments conducted using the Maple computer algebra system are presented. Cases for improving method complexity and for generalizing this approach are also discussed.

Keywords: Ashenhurst decomposition of Boolean functions; truth tables; inversion of Cartesian join of tables; inversion of JOIN ON with equality; systems of quadratic equations over the finite field $GF(2)$; Gröbner bases.

For citation: Emelyanov P.G. Ashenhurst decomposition for table-defined Boolean functions. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026. pp. 33-44 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-3.

1. Введение

Декомпозиция сложных систем, моделируемых булевыми функциями, – широко распространённый метод анализа и преобразований, применяемых в различных областях (логический синтез микросхем, надёжность систем, теория голосования, машинное обучение и системы принятий решений). Декомпозиция в менее сложные подсистемы, кроме того, что уменьшает в целом представление исходной системы (при этом, возможно, оптимизируются и другие её характеристики), позволяет выявить скрытую структуру системы, что, в частности, позволяет провести более глубокий анализ и преобразования.

Декомпозиция Ашенхёрста булевых функций – один из старейших методов функциональной декомпозиции, появившийся в 1950-х годах [1] в связи с исследованиями по оптимизации переключательных схем. Он состоит в отыскании для n -местной булевой функции $F : \mathbf{B}^n \rightarrow \mathbf{B}$, где $\mathbf{B} = \{0,1\}$ и $\mathbf{B}^n = \mathbf{B} \times \dots \times \mathbf{B}$ (n раз), представления в виде $F(X) = H(G(X_1), X_2)$, где $X_1 \cap X_2 = \emptyset$ (называемые соответственно множествами *связанных* и *свободных переменных*), $G : \mathbf{B}^m \rightarrow \mathbf{B}$, $H : \mathbf{B}^{n-m+1} \rightarrow \mathbf{B}$, $2 \leq m \leq n-1$ (функцию H ещё называют функцией композиции). Вскоре Кёртис [2] обобщил эту декомпозицию на случай $G : \mathbf{B}^m \rightarrow \mathbf{B}^l$, $H : \mathbf{B}^{n-m+l} \rightarrow \mathbf{B}$, при подходящих m и $l \geq 1$. Декомпозиция вида $F(X) = H(G_1(X_1), G_2(X_2))$, где функция композиции H двухместна, называется (*алгебраической*) *би-декомпозицией* и может рассматриваться обобщением декомпозиции Ашенхёрста в том смысле, что произвольная функция композиции, используемая в случае Ашенхёрста, выражается с помощью двухместной композиционной функции и функции G_2 , которая также может быть выражена композицией двухместных функций. В общем случае множества X_1 и X_2 могут пересекаться и данное обобщение декомпозиции Ашенхёрста-Кёртиса было рассмотрено Ротом и Карпом

в [3]. Би-декомпозиция с разделяемыми между компонентами переменными называется *булевой*.

Обширный обзор работ к середине 1990-х годов по данной теме, всегда игравшей значительную роль в многоуровневой декомпозиции, дан в [4] (см. также [5] и [6]), недавние – в [7] и [8]. В последнее время интерес к ней стимулируется активным развитием *ПЛИС-технологий*, где данный вид декомпозиции используется на нескольких технологических этапах решения разных задач синтеза схем (краткий обзор в [9]). Его важность подчёркивается существующим уже сейчас интересом к реализации на квантовых компьютерах [10].

Первые методы отыскания декомпозиции Ашенхёрста опирались на карты Карно, являющиеся некоторым вариантом *таблиц истинности* булевых функций. В данной работе также используются таблицы истинности (это покрывает и другие аналогичные варианты представления булевых функций: черный ящик, характеристический вектор значений, формулы в СДНФ/СКНФ, характеристические векторы множеств единиц/нулей). Обзор последних работ см. в [9] и [8]. Рассматриваются не только полностью специфицированные таблицы истинности, задающие *тотальные* булевы функции, но и *частичные*, то есть специфицированные не на всех возможных входах из $\mathbf{B}^n$ (в английской терминологии – *don't care inputs*, *DC-inputs*). Отметим, что частично специфицированные функции, причём с малым числом единиц, – типичная ситуация в задачах машинного обучения и системах принятия решений (см., например, [11]).

Фундаментальной проблемой декомпозиции Ашенхёрста является разбиение входных переменных на множество связанных и свободных (для декомпозиции Рота-Карпа ещё рассматривается множество разделяемых между компонентами переменных). Когда это разбиение предписано по каким-то причинам (например, если имеется задержка в вычислениях части переменных, то их целесообразно считать свободными), тогда задача сводится к оптимальному дизайну функций-компонент декомпозиции [12:гл.3]. Однако в общем случае задача разбиения переменных является сложной. Для представления булевых функций в виде бинарных диаграмм решений (БДР), для которого практически значимые методы были разработаны первыми, она сводится либо к отысканию «хорошего» упорядочивания переменных [13], либо к построению специального вида доминаторов [14], либо использованию дополнительных графовых конструкций [15]. Широко известные проблемы БДР (чувствительность к количеству и порядку переменных, большие размеры для некоторых классов функций) стимулировали исследования в области эвристик [16], позволяющих за разумное время построить удовлетворительное по качеству разбиение, а также представление задачи в виде задачи выполнимости логических формул [17, 12:гл.5-6, 18]. В работах [19] (эвристический) и [20] (с учётом некоторых свойств обрабатываемых булевых функций) би-декомпозиция строится посредством вычислений над графами, сконструированными из нормальных форм.

В [21-24] был разработан детерминированный полиномиальной временной сложности алгоритм факторизации мультилинейных полиномов над конечным полем $GF(2)$, который, по сути, даёт метод алгебраической би-декомпозиции булевых функций в АНФ (полиномы Жегалкина). В [25] на его основе был разработан алгоритм булевой би-декомпозиции с предписанным множеством разделяемых переменных. В [26] эти подходы были использованы для *декомпозиции таблиц реляционных СУБД* (обращения SQL оператора соединения JOIN ON с равенством), где была предложена *конъюнктивная би-декомпозиция* таблично заданных булевых функций с разделяемой между компонентами переменной.

На рис. 1 слева показана функция семи переменных F , заданная таблицей, представляющей её единицы (то есть на выход выдаётся ответ 1, если соответствующая комбинация входных параметров присутствует в таблице, 0 иначе; для функций F_1 и F_2 аналогично). Входная переменная x_4 описывает «соединение», подходящее для данной декомпозиции и функция F представима в виде $F(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = AND(F_1(x_1, x_2, x_3, x_4), F_2(x_4, x_5, x_6, x_7))$.

Декомпозиция состоит в отыскании разбиения множества атрибутов всей таблицы на подмножества атрибутов компонент посредством декартовой декомпозиции (обращения SQL оператора соединения CROSS ON) подтаблиц, соответствующих строкам, где переменная соединения равна 0 и 1 соответственно. Отметим, что эта задача имеет *полиномиальную временную сложность* относительно размера таблицы.

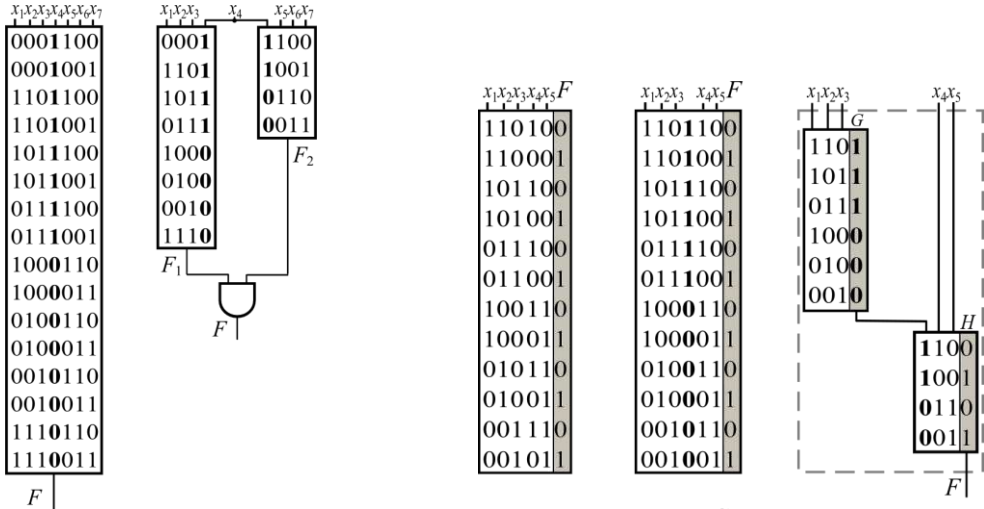


Рис. 1. Декомпозиция таблично заданных булевых функций.
Fig. 1. Decomposition of table-defined Boolean functions.

Цель исследования – применить данную технику декомпозиции таблиц к отысканию декомпозиции Ашенхёрста. Действительно, если даны две табличные функции (частичные или тотальные), то оператор соединения-с-равенством *атрибута-результата* одной функции и *атрибута-входа* другой функции (при подходящей типизации этих атрибутов) выдаёт табличную функцию, являющуюся композицией исходных функций. Случай, когда соответствующий *атрибут-соединение* присутствует в таблице, технически аналогичен случаю, рассмотренному в [26]. Однако получаемая в этом случае декомпозиция $F(x_1, \dots, x_{k-1}, x_k, x_{k+1}, \dots, x_n) = H(G(x_1, \dots, x_{k-1}), x_{k+1}, \dots, x_n)$, порожаемая переменной x_k , демонстрирует избыточность этой переменной, так как x_k описывается функциональной зависимостью G от переменных $x_1, \dots, x_{k-1}$. Такого вида функциональные зависимости широко изучаются для реляционных баз данных (см., например, [27]).

Большой интерес представляет случай, когда этот атрибут отсутствует в таблице. На рис. 1 справа описана декомпозиция для этого случая. В итоге функция F представима в виде $F(x_1, x_2, x_3, x_4, x_5) = H(G(x_1, x_2, x_3), x_4, x_5)$ (все действия проводятся с таблицами истинности). Таким образом, эта декомпозиция – процесс, обратный композиции – состоит в отыскании значений атрибута-соединения, разбиения множества имеющихся атрибутов по компонентам декомпозиции и реконструкции самих таблиц-компонент. Более точно, в данной работе декомпозиция Ашенхёрста строится на основе решения следующих задач.

- По таблице истинности строится некоторый полином, включающий в том числе переменные, которые описывают значения атрибута-соединения.
- Формулированием некоторых условий, относящих пару атрибутов-входов к разным компонентам декомпозиции, для переменных атрибута-соединения порождается система квадратных уравнений над полем $GF(2)$.
- Система решается (на данный момент с использованием техники базисов Грёбнера), что даёт значения атрибута-соединения, которые разбивают строки исходной таблицы истинности на две части.

- В этих частях проводится декомпозиция на подтаблицы, непересекающиеся по атрибутам-входам: выполняется обращение SQL оператора CROSS JOIN [26], являющегося, по сути, декартовым произведением таблиц. В случае успеха, полученные подтаблицы представляют собой компоненты декомпозиции.

В разделе 2 представлены используемые в данной работе определения и основные результаты о декомпозиции таблиц. В разделе 3 дано описание метода построения декомпозиции Аштенхёрста и иллюстрирующие построения примеры. Дальнейшие направления исследований изложены в заключении.

2. Основные определения и результаты по декомпозиции таблиц

Декомпозицию таблиц можно рассмотреть в чисто «комбинаторных» терминах манипуляций над таблицами [26]. Однако для наших целей больше подойдёт оригинальный «алгебраический» подход, изложенный в [21] и [22], послуживший основой табличному методу. Ниже даётся его краткое изложение.

Для таблицы, в общем случае не обязательно булевой, можно построить *мультилинейный однородный полином* (все переменные в первой степени, все мономы одной длины), в котором переменные, соответствующие атрибутам, проиндексированы значениями этих атрибутов, встречающимися в таблице. Полиномы рассматриваются в кольце $GF(2)[x_1, \dots, x_n]$, где n равно сумме мощностей областей (использованных) значений атрибутов таблицы. Например, для таблиц, приведённых на рис. 2), соответствующими полиномами будут:

$$A_0B_0C_0D_0E_0 + A_0B_0C_0D_1E_0 + A_0B_0C_0D_0E_1 + A_0B_1C_1D_0E_0 + A_0B_1C_1D_1E_0 + A_0B_1C_1D_0E_1 + A_1B_1C_0D_0E_0 + A_1B_1C_0D_1E_0 + A_1B_1C_0D_0E_1 + A_1B_1C_1D_0E_0 + A_1B_1C_1D_1E_0 + A_1B_1C_1D_0E_1 = (A_0B_0C_0 + A_0B_1C_1 + A_1B_1C_0 + A_1B_1C_1)(D_0E_0 + D_1E_0 + D_0E_1).$$

A	B	C	D	E
0	0	0	0	0
0	0	0	1	0
0	0	0	0	1
0	1	1	0	0
0	1	1	1	0
0	1	1	0	1
1	1	0	0	0
1	1	0	1	0
1	1	0	0	1
1	1	1	0	0
1	1	1	1	0
1	1	1	0	1

=

A	B	C
0	0	0
0	1	1
1	1	0
1	1	1

⊗

D	E
0	0
1	0
0	1

Рис. 2. Декартова декомпозиция – обращение SQL оператора CROSS JOIN.

Fig. 2. Cartesian decomposition is the inverse of the SQL CROSS JOIN operator.

Назовём такой полином *табличным*. Очевидно, декартово произведение малых таблиц (CROSS JOIN) соответствует перемножению табличных полиномов, результат которого – табличный полином некоторой таблицы. Отметим, что все табличные полиномы мультилинейны, так как все атрибуты считаются различными независимо от имён и типов. Таким образом, *декартова декомпозиция* некоторой *таблицы* – это факторизация её табличного полинома, при этом полиномы-факторы (*компоненты декомпозиции*) будут иметь непересекающиеся множества переменных.

В общем случае задача факторизации полиномов над конечными полями является алгоритмически сложной. Только в начале 2010-х были получены результаты о полиномиальной временной сложности этой задачи для мультилинейных полиномов в

разных представлениях. В [21] и [22] это было показано для полиномов из $GF(2)[x_1, \dots, x_n]$, представленных в виде списка мономов (арифметических схем из класса $\Sigma_m \Pi_n$). Этот алгоритм основывается на отыскании разбиения множества переменных исходного полинома на непересекающиеся подмножества переменных факторов с последующей «проекцией» полинома на эти множества для построения самих факторов. Для формулировки необходимого свойства потребуется несколько определений.

Для полинома P $vars(P) = \{x_1, \dots, x_n\}$ – множество переменных. Мультилинейные полиномы $GF(2)[x_1, \dots, x_n]$ образуют область однозначной факторизации, а значит неприводимые факторы P индуцируют однозначное разбиение $vars(P)$. Означивание полинома P по переменной x значением v обозначается $P_{x=v}$. Проекцией полинома P на подмножество переменных $V \subseteq vars(P)$, обозначаемая $P \downarrow V$, – это новый полином, образованный по следующим правилам: рассматривается множество мономов исходного полинома, во всех элементах этого множества переменные $vars(P) \setminus V$ означиваются в 1, а затем из получившегося множества мономов формируется результат (например, $(xu + xv + xy) \downarrow \{x, y\} = x + xy$, в частности, $P \downarrow vars(P) = P$ и $P \downarrow \emptyset = 1$). Длина полинома m – это количество его ненулевых мономов. Размер полинома – это сумма длин его мономов, где длина монома равна мощности его множества переменных. Размер монома можно оценить как $O(nm)$. Формальная производная полинома P по переменной x , обозначаемая через P'_x , определяется следующим образом: $P'_x = \sum_P (\dots x^d \dots)'_x = \sum_P d(\dots x^{d-1} \dots)$. Для мультилинейных полиномов $GF(2)[x_1, \dots, x_n]$ $P'_x = P_{x=0} + P_{x=1}$. Будем предполагать, что полином не имеет тривиальных факторов, то есть переменных, которые входят во все мономы полинома (для табличных полиномов это отсутствие константных атрибутов). Для полинома P из $K[X, Y]$ соответствующий полином из $K[Y][X]$ будем обозначать $Coll(P, Y)$.

Следующая теорема является основой алгоритма факторизации мультилинейных полиномов из $GF(2)[x_1, \dots, x_n]$ (см. [21–24]).

Теорема 1. Если $P = GH$, тогда для любой переменной $x \in vars(P)$ верно:

1. $(P_{x=0}P'_x)'_y \neq 0$, если x и y принадлежат одному фактору, при этом он неразложим;
2. $(P_{x=0}P'_x)'_y = 0$, если x и y принадлежат разным факторам, при этом фактор, содержащий y , может быть разложимым.

Полином $P_{x=0}P'_x$ называется сортирующим (он не обязательно мультилинейный), переменная x – сортирующей. Если P неразложим, то $\forall x, y \in vars(P)$, $x \neq y$, верно $(P_{x=0}P'_x)'_y \neq 0$.

Проверка этого условия, то есть сортировки переменных полинома по двум множествам относительно некоторой сортирующей переменной, требует $O(n^2m^2)$. Ту же сложность имеет и проекция полинома на эти множества. Таким образом, общая временная сложность алгоритма факторизации мультилинейных полиномов является квадратичной относительно его размера.

Декомпозиция таблиц, полученных с помощью соединения-с-равенством по некоторому атрибуту (JOIN ON с проверкой равенства, см. пример на рис. 1 слева), требует некоторых дополнительных шагов. Во-первых, факторизация выполняется на табличных полиномах, соответствующих подтаблицам, где рассматриваемый атрибут имеет одно и тоже значение (таким образом, для булевых таблиц таких подтаблиц две). Так как в этих полиномах этому атрибуту всегда соответствует одна и та же переменная, она не включается в полином. Во-вторых, выполняется полная факторизация полиномов, то есть вычисляются все неразложимые факторы. Это необходимо, так как полученные для подтаблиц разбиения атрибутов могут быть разными (см. пример на рис. 3), что не всегда позволяет выбрать корректные подмножества атрибутов для выполнения проекции. Поэтому, в-третьих, выполняется согласование полученных разбиений. Если оно завершилось успешно (см. процедуру в [26]), то строятся компоненты декомпозиции.

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	1

$$P_{D_0=1, D_1=0} = A_0 B_0 C_0 + A_0 B_0 C_1 + A_0 B_1 C_0 + A_0 B_1 C_1 + A_1 B_0 C_0 + A_1 B_0 C_1 \\ = (A_0 B_0 + A_0 B_1 + A_1 B_0)(C_0 + C_1)$$

$$P_{D_0=0, D_1=1} = A_0 B_0 C_0 + A_0 B_0 C_1 + A_0 B_1 C_0 + A_1 B_0 C_0 + A_1 B_0 C_1 + A_1 B_1 C_0 \\ = (A_0 + A_1)(B_0 C_0 + B_0 C_1 + B_1 C_0)$$

Для $D = 0$ разбиение $\{A, B\} \{C\}$.

Для $D = 1$ разбиение $\{A\} \{B, C\}$.

Рис. 3. Пример декомпозиции подтаблиц с несогласованными разбиениями атрибутов.

Fig. 3. Example of decomposition of sub-tables with incompatible attribute partitions.

3. Декомпозиция Ашенхёрста

Как уже отмечалось ранее, в [26] рассмотрен подход, когда соответствующий атрибут-соединение присутствует в таблице в явном виде. Однако интересен случай декомпозиции Ашенхёрста, когда соответствующего атрибута-входа нет. Он вычисляется в процессе декомпозиции на основе предположения, что часть исходных атрибутов-входов задаёт некоторую функцию, с помощью значений которой можно расширить другую часть исходных атрибутов-входов так, чтобы составленная из них композиция соответствовала определению исходной функции. Декомпозицию можно выполнять и для таблиц истинности (атрибут-результат опускается при рассмотрении, см. обсуждение в конце раздела), и для таблиц, описывающих множество единиц/нулей (см. рис. 1). Ниже описаны шаги построения декомпозиции.

Шаг 1. Дополним каждый моном табличного полинома P соответствующей переменной из нового множества переменных $J = \{J_1, \dots, J_m\}$, где m – длина полинома (см. рис. 4). Обозначим новый полином PJ . Означивание некоторой переменной из J в 0 или 1 будет относить соответствующий ей моном к одной или другой подтаблице, внутри которых нужно выполнять декартову декомпозицию (факторизацию табличных полиномов, соответствующих этим подтаблицам). Это означивание будет найдено как решение некоторой системы уравнений, порождённой с использованием **Теоремы 1** (построение системы на **Шагах 2-4**).

Шаг 2. Выберем из $vars(P)$ сортирующую переменную x . С учётом того, что полином не содержит тривиальных факторов (таблица не содержит константных столбцов), можно выбрать произвольную переменную. Однако, для снижения объёма вычислений следует выбирать переменные, которые встречаются наиболее часто или наиболее редко в исходном полиноме, так как это минимизирует длину полинома-произведения $Q_x = PJ_{x=0}PJ'_x \in GF(2)[vars(P), J]$. Для табличного полинома, таблица которого представлена слева на рис. 4, такой переменной может быть выбрана D с любым индексом. Пусть это будет переменная D_0 . Вычислим полином Q_{D_0} .

Шаг 3. Построим полином $Coll(Q_{D_0}, J) \in GF(2)[J][vars(P)]$. Для того, чтобы какая-то переменная y и переменная D_0 были из разных факторов, необходимо, чтобы производная полинома $Coll(Q_{D_0}, J)$ по переменной y равнялась тождественно нулю. $Coll(Q_{D_0}, J)'_y$ может содержать ненулевые коэффициенты из $GF(2)[J]$, таким образом, чтобы установить, при каких значениях переменных из J переменная y и переменная D_0 будут заведомо из разных факторов, необходимо чтобы полиномы-коэффициенты обращались в ноль. Эти тождества и порождают искомую систему уравнений. На рис. 4 приведены 16 уравнений для коэффициентов полинома $Coll(Q_{D_0}, J)'_{C_0}$ (переменная C_0 выбрана из иллюстративных соображений – она даёт наименьшее количество уравнений). Если декомпозиция существует, то

достаточно решить систему, порождённую одной подходящей переменных из J . Однако, так как соответствующее декомпозиции разбиение как раз и строится, то эта подходящая переменная нам неизвестна и в общем случае необходимо испытать все переменные из $vars(P) \setminus D_0$.

A	B	C	D	E	F	R
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	0	1	1	1
0	1	1	1	0	1	1
1	0	0	0	0	1	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	0	1	1	1
1	0	1	1	0	1	1
1	1	0	0	0	0	1
1	1	0	0	1	1	1
1	1	0	1	0	1	1

A	B	C	J	D	E	F	R
0	1	0	J_1	0	0	1	0
0	1	0	J_2	0	1	0	1
0	1	0	J_3	1	1	0	0
0	1	1	J_4	0	0	0	1
0	1	1	J_5	0	1	1	1
0	1	1	J_6	1	0	1	1
1	0	0	J_7	0	0	1	0
1	0	0	J_8	0	1	0	1
1	0	0	J_9	1	1	0	0
1	0	1	J_{10}	0	0	0	1
1	0	1	J_{11}	0	1	1	1
1	0	1	J_{12}	1	0	1	1
1	1	0	J_{13}	0	0	0	1
1	1	0	J_{14}	0	1	1	1
1	1	0	J_{15}	1	0	1	1

$$\begin{aligned} J_1 J_6 &= 0, \\ J_2 J_6 &= 0, \\ J_3 J_4 &= 0, \\ J_3 J_5 &= 0, \\ J_7 J_{12} &= 0, \\ J_8 J_{12} &= 0, \\ J_9 J_{10} &= 0, \\ J_9 J_{11} &= 0, \\ J_1 J_{12} + J_6 J_7 &= 0, \\ J_2 J_{12} + J_6 J_8 &= 0, \\ J_3 J_{10} + J_4 J_9 &= 0, \\ J_3 J_{11} + J_5 J_9 &= 0, \\ J_4 J_{15} + J_6 J_{13} &= 0, \\ J_5 J_{15} + J_6 J_{14} &= 0, \\ J_{10} J_{15} + J_{12} J_{13} &= 0, \\ J_{11} J_{15} + J_{12} J_{14} &= 0 \end{aligned}$$

Рис. 4. Исходная частичная таблица истинности функции R , таблица полинома PJ , коэффициенты полинома $Coll(Q_{D_0}, J)_{C_0}'$, составляющие исходную систему уравнений.

Fig 4. The original partial truth table of the function R , the table of PJ polynomial, the coefficients of the polynomial $Coll(Q_{D_0}, J)_{C_0}'$ giving the original system of equations.

Шаг 4. Исходная система довольно компактна, однако в данном случае это не является достоинством, так как с большой вероятностью имеет много решений, не все из которых полезны, но которые нужно будет обрабатывать на следующих шагах. Было решено изучить влияние пополнения системы новыми уравнениями на сложностные (временные и ёмкостные) характеристики вычислений.

Так как означивание переменных из множества J может быть произвольным подходящим под условия, то система будет иметь два противоположных решения (одно отрицание другого – все равно какие конкретные значения будут приписаны связывающему параметру). Таким образом, переменные, входящие в уравнения вида $J_s J_t = 0$, так же удовлетворяют и условию $(J_s + 1)(J_t + 1) = 0$, а значит эти два квадратных уравнения могут быть заменены одним линейным $J_s + J_t = 1$ (иными словами – эти параметры должны быть не равны друг другу, в алгебре логики они связаны неравенством). В квадратные уравнения, содержащие суммы произведений, можно выполнить такую же подстановку. На рис. 5 представлена расширенная система с выполненными подстановками (см. продолжение обсуждения подстановок на следующем шаге).

Шаг 5. Для отыскания решений расширенной системы сначала строится её базис Грёбнера [27], а затем приведённая система решается решателем для модулярных систем уравнений (по модулю 2). Эксперименты проводились в системе компьютерной алгебры Maple 2020 [28]. Параметры вызова метода Groebner[Basis]: характеристика поля равна 2, выбор порядка переменных на усмотрение системы, метод построения базиса по умолчанию (системой всегда выбирался алгоритм Фожера F_4 [29]). Это самый затратный шаг в работе алгоритма декомпозиции. Было установлено, что от количества переменных время работы алгоритма зависит слабо (тестировались таблицы с количеством атрибутов от 10 до 50). В то же время было обнаружено, что при минимальном расширении системы (только линейными уравнениями) уже при сравнительно небольшом количестве мономов (порядка 700+ строк в таблице) вычислительное ядро Maple перестаёт функционировать без особо содержательной диагностики. Вероятно, это связано с спецификой реализации данного алгоритма, который оптимизирован [28] на высокую производительность использованием вычислений,

ориентированных на размер машинного слова. Вычисление базиса с использованием явного указания классического алгоритма Бухбергера или неоптимизированной версии алгоритма F_4 позволяет строить базисы для больших таблиц (хотя и медленнее, чем с помощью других алгоритмов, предлагаемых Maple). Однако для работы с большим количеством мономов они требуются все больше и больше памяти, а это приводит к активному использованию диска для виртуальной памяти (использовался компьютер с 48 Гб ОЗУ), в следствии чего производительность радикально деградирует. Пополнение системы уравнениями, рассмотренное на **Шаге 4**, даёт возможность строить декомпозицию для функций, определяемых таблицами со 1000 и более строк. Затраты памяти – от нескольких сотен мегабайт до пары гигабайт. Время – от десятков до сотен секунд, что конечно же не является рекордным на сегодняшний день. Можно также отметить, что чем меньше асимметрия между компонентами по количеству атрибутов и строк (то есть они примерно одинаковые), тем дольше выполняются вычисления.

$$\begin{aligned}
 J_1 + J_6 = 1, J_2 + J_6 = 1, J_3 + J_4 = 1, J_3 + J_5 = 1, & J_1 + J_6 = 1, \\
 J_7 + J_{12} = 1, J_8 + J_{12} = 1, J_9 + J_{10} = 1, J_9 + J_{11} = 1, & J_2 + J_6 = 1, \\
 J_1J_{12} + J_6J_7 = 0, J_2J_{12} + J_6J_8 = 0, J_3J_{10} + J_4J_9 = 0, & J_3 + J_4 = 1, \\
 J_3J_{11} + J_5J_9 = 0, J_4J_{15} + J_6J_{13} = 0, J_5J_{15} + J_6J_{14} = 0, & J_3 + J_5 = 1, \\
 J_{10}J_{15} + J_{12}J_{13} = 0, J_{11}J_{15} + J_{12}J_{14} = 0, & J_3 + J_9 = 0, \\
 J_1J_{12} + J_6J_7 + J_1 + J_6 + J_7 + J_{12} = 0, & J_3 + J_{10} = 1, \\
 J_2J_{12} + J_6J_8 + J_2 + J_6 + J_8 + J_{12} = 0, & J_3 + J_{11} = 1, \\
 J_3J_{10} + J_4J_9 + J_3 + J_4 + J_9 + J_{10} = 0, & J_6 + J_7 = 1, \\
 J_3J_{11} + J_5J_9 + J_3 + J_5 + J_9 + J_{11} = 0, & J_6 + J_8 = 1, \\
 J_4J_{15} + J_6J_{13} + J_4 + J_6 + J_{13} + J_{15} = 0, & J_6 + J_{12} = 0, \\
 J_5J_{15} + J_6J_{14} + J_5 + J_6 + J_{14} + J_{15} = 0, & J_3 + J_6 + J_{13} + J_{15} = 1, \\
 J_{10}J_{15} + J_{12}J_{13} + J_{10} + J_{12} + J_{13} + J_{15} = 0, & J_3 + J_6 + J_{14} + J_{15} = 1, \\
 J_{11}J_{15} + J_{12}J_{14} + J_{11} + J_{12} + J_{14} + J_{15} = 0 & J_3J_6 + J_3J_{15} + J_6^2 + J_6J_{15} + J_6 + J_{15} = 0
 \end{aligned}$$

Рис. 5. Расширенная система уравнений (24 уравнения) и её базис Грёбнера (13 уравнений).

Fig. 5. The extended system of equations (24 equations) and its Gröbner basis (13 equations).

Шаг 6. Следующим шагом является построение согласованного разбиения атрибутов исходной таблицы. Для этого перебираются решения системы квадратичных модулярных уравнений для переменных J_k , полученные на предыдущих шагах, и с их помощью формируются полиномы, соответствующие подтаблицам для случаев $J = 0$ и $J = 1$. Выполняется полная факторизация этих полиномов, множества переменных факторов определяют возможное разбиение атрибутов. Эти разбиения для подтаблиц согласовываются (см. [26]). Если согласование прошло успешно, то выполняется реконструкция таблиц для функций, формирующих декомпозицию, с использованием полученных факторов для подтаблиц и значений J . Реконструкцию таблиц можно описать и в терминах табличных манипуляций (см. пример в конце раздела). Решений системы, порождающих декомпозицию, может быть несколько, перебирать их все или нет – зависит от решаемых задач.

При использовании решения (всего в этом случае их 5, в расширенной системе их число сократилось примерно в 3 раза по сравнению с исходной системой)

$$\begin{aligned}
 \{J_1 = 0, J_2 = 0, J_3 = 0, J_4 = 1, J_5 = 1, J_6 = 1, J_7 = 0, J_8 = 0, J_9 = 0, \\
 J_{10} = 1, J_{11} = 1, J_{12} = 1, J_{13} = 1, J_{14} = 1, J_{15} = 1\}
 \end{aligned}$$

полиномы для подтаблиц факторизуются следующим образом:

$$\begin{aligned}
 & \text{для } J_k \text{ означенных в 0: } (A_0B_1 + A_1B_0)C_0(D_0E_0F_1 + D_0E_1F_0 + D_1E_1F_0) \\
 & \text{для } J_k \text{ означенных в 1: } (A_0B_1C_1 + A_1B_0C_1 + A_1B_1C_0)(D_0E_0F_0 + D_0E_1F_1 + D_1E_0F_1)
 \end{aligned}$$

Отметим, что в первом случае отсутствует переменная C_1 . Это произошло по причине её отсутствия в полиноме для подтаблицы для случая $J = 0$, так как при данном решении

атрибут C принимает константное значение 0. Но это не имеет значения, так как, если подходящее разбиение существует, переменные с одним именем и разными индексами заведомо попадают в одну часть. Таким образом, для случая $J = 0$ разбиением атрибутов является $[\{A, B\}, \{C\}, \{D, E, F\}]$, для случая $J = 1$ разбиением является $[\{A, B, C\}, \{D, E, F\}]$. В итоге согласованным разбиением атрибутов является $[\{A, B, C\}, \{D, E, F\}]$, так как отнесение для случая $J = 0$ атрибута $\{C\}$ к части $\{D, E, F\}$ не согласуется с разбиением случая $J = 1$.

A	B	C	G
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1

J	D	E	F	H
0	0	0	1	0
0	0	1	0	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1

Рис. 6. Декомпозиции Ашенхёрста: $R(A, B, C, D, E, F) = H(G(A, B, C), D, E, F)$.
Fig 6. The Ashenhurst decomposition: $R(A, B, C, D, E, F) = H(G(A, B, C), D, E, F)$.

Далее восстанавливаем функции G и H , формирующие декомпозицию Ашенхёрста. Например, из второй таблицы, представленной на рис. 4, таблицы функций G и H получаются выбором двух групп атрибутов A, B, C, J , и J, D, E, F, R , соответственно, с последующей проекцией таблицы на эти группы (то есть удалением из результирующих таблиц лишних атрибутов и повторяющихся строк), после чего атрибут J первой таблицы переименовывается в G , а R второй в H (см. рис. 6). Отметим, используемые табличные полиномы подтаблиц не говорят, какой из них определяет функцию G , а какой функцию H (то есть только по ним, без учёта значений атрибута R). Если попытаться построить для групп атрибутов A, B, C, J, R , и J, D, E, F , декомпозицию вида $H(G(D, E, F), A, B, C)$, то для данной функции R это не получится. Однако, если рассмотреть для таблицы истинности табличный полином, включающий атрибут-результат (то есть переменные для R), то связь групп атрибутов с функциями будет предопределена: попавшие к R соответствуют функции H .

Заключение

В статье рассмотрена декомпозиция Ашенхёрста, то есть декомпозиция вида $F(X, Y) = H(G(X), Y)$, для (частичных) булевых функций, заданных в виде таблицы истинности или её вариантов. Задача декомпозиции сведена к задаче решения с использованием базисов Грёбнера системы квадратных уравнений над конечным полем $GF(2)$ для некоторых специально введённых параметров. Решение системы индуцирует разбиение входных переменных функции F на входные (связанные) переменные X функции G и входные (свободные) переменные Y функции H , при этом сами параметры определяют композицию G и H связывая их: для G они выступают в качестве значений результата, а для H – значений аргумента. На данный момент рассматриваемый метод не демонстрирует высокой производительности при его реализации в системе компьютерной алгебры Maple, однако открывает новые способы исследования этого вида декомпозиции. Дальнейшие направления исследований:

- Исследовать комбинаторную структуру модулярных уравнений для параметров связывания таблиц с целью улучшения эффективности работы алгоритмов решения рассматриваемой системы уравнений.
- Исследовать эффективность отыскания параметров связывания путём построения по системе модулярных уравнений эквивалентной логической формулы и применения SAT-решателей, а также другие методы.
- Представляет большой интерес вопрос, можно ли обобщить данный метод на декомпозицию Кёртиса, если допустить означивание параметров связывания натуральными числами. Двоичное представление значений вычисленных

параметров представляли бы выходы функции G и соответствующие входы функции H .

- В данной работе для связывания таблиц использовалась проверка на равенство, что, по сути дела, является тождественной «функцией связывания». Например, если бы использовалось неравенство, то для булевых таблиц это означало бы подстановку в функцию композиции H отрицания функции G . Однако тест в операторе JOIN ON может быть более сложный и в нем можно использовать не один параметр связывания. По сути дела, это построение декомпозиции вида $F(X, Y) = H(T(X, Y, G(X)), Y)$, где функция T , имея возможность использовать весь вход функции F , «усложняет» поведение функции G (сама G зависит только от части входа X). Равно можно отнести T и к определению функции H для «усложнения» поведения её аргумента, принимающего результат G . Это открывает возможности построения декомпозиции типа Рота-Карпа с разделяемыми переменными $F(X, Y, Z) = H(G(X, Z), Y, Z)$.

Список литературы / References

- [1]. Ashenurst R. The Decomposition of Switching Functions. In Proc. of the International Symposium on the Theory of Switching, 1957. pp. 74–116.
- [2]. Curtis H. A. A New Approach to the Design of Switching Circuits. D. van Nostrand company, 1962. 635 p.
- [3]. Roth J. P., Karp R. M. Minimization over Boolean Graphs. IBM Journal of Research and Development, 6(2), 1962, pp. 227–238. DOI: 10.1147/rd.62.0227.
- [4]. Perkowski M.A., Grygiel S. A Survey of Literature on Function Decomposition, Version IV. Rep. of PSU Electrical Engineering Department, Portland State University, Portland, Oregon, USA. Опубликовано: November 20, 1995. 188 p. Available at <https://web.cecs.pdx.edu/~mperkows/=PUBLICATIONS/PER/G1995/survey.pdf>, accessed 25.10.2025.
- [5]. Mishchenko A., Steinbach B., Perkowski M. An Algorithm for Bi-Decomposition of Logic Functions. Proc. of the 38th Design Automation Conference, 2001, pp. 103–108. DOI: 10.1145/378239.378353.
- [6]. Steinbach B., Lang C. Exploiting Functional Properties of Boolean Functions for Optimal Multi-Level Design by Bi-Decomposition. Artificial Intelligence Review, 20(3-4), 2003, pp. 319-360. DOI: 10.1023/B:AIRE.0000006606.01771.8f.
- [7]. Mishchenko A., Brayton R., Calvino A.T., De Micheli G. Boolean Decomposition Revisited. In Proc. of the 32th International Workshop on Logic & Synthesis, 2023. 5 p.
- [8]. Hien B., Walter W., Calvino A.T., Mishchenko A., Wille R. Ashenurst-Curtis Decomposition Using Don't Cares. In Proc. of the 34th International Workshop on Logic & Synthesis, 2025. 8 p.
- [9]. Calvino A. T., De Micheli G., Mishchenko A., Brayton R. Enhancing Delay-Driven LUT Mapping with Boolean Decomposition. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 44(3), 2024, pp. 1017-1030. DOI: 10.1109/TCAD.2024.3457378.
- [10]. Li Y., Tsai E., Perkowski M., Song X. Grover-Based Ashenurst-Curtis Decomposition Using Quantum Language Quipper. Quantum Information and Computation, 19(1-2), 2019, pp. 0035-0066. DOI: 10.26421/QIC19.1-2-4.
- [11]. Miyasaka Y., Mishchenko A., Wawrzynek J., Fraser N.J. Synthesizing a Class of Practical Boolean Functions Using Truth Tables. In Proc. of the 31th International Workshop on Logic & Synthesis, 2022. 7 p.
- [12]. Advanced Techniques in Logic Synthesis, Optimizations and Applications, eds. Khatri, S.P., Gulati, K. Springer, New York, NY, USA, 2011. 423 p. DOI: 10.1007/978-1-4419-7518-8.
- [13]. Stanion T., Sechen C. A Method for Finding Good Ashenurst Decompositions and Its Application to FPGA Synthesis. In Proc. of the 32th Design Automation Conference, 1995. pp. 60-64. DOI: 10.1145/217474.217507.
- [14]. Yang C., Ciesielski M. BDS: a BDD-Based Logic Optimization System. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 21(7), 2002, pp. 866-876. DOI: 10.1109/TCAD.2002.1013899.
- [15]. Choudhury M., Mohanram K. Bi-decomposition of Large Boolean Functions Using Blocking Edge Graphs. Proc. of the 2010 IEEE/ACM International Conference on Computer-Aided Design, 2010, pp. 586–591. DOI: 10.1109/ICCAD.2010.5654210.

- [16]. Bengtsson T., Martinelli A., Dubrova E. A BDD-based fast heuristic algorithm for disjoint decomposition. In Proc. of the 31st Asia and South Pacific Design Automation Conference, 2003. pp. 191–196. DOI: 10.1145/1119772.1119810.
- [17]. Lin H.-P., Jiang J.-H. R., Lee R.-R. To SAT or not to SAT: Ashenhurst decomposition in a large scale. In Proc. of the 2008 IEEE/ACM International Conference on Computer-Aided Design, 2008. pp. 32–37. DOI: 10.5555/1509456.1509476.
- [18]. Chen H., Janota M., Marques-Silva J. QBF-Based Boolean Function Bi-Decomposition. In Proc. of the 2012 *Design, Automation & Test in Europe Conference*, 2012. pp. 816–819. DOI: 10.1109/DATE.2012.6176606.
- [19]. Поттосин Ю.В. Эвристический метод алгебраической декомпозиции частичных булевых функций. Информатика, том 17, № 3, 2020, стр. 44–53. DOI: 10.37661/1816-0301-2020-17-3-44-53. / Pottosin Yu.V. A Heuristic Method for Bi-Decomposition of Partial Boolean Functions. Informatics, 2020, vol. 17, no. 3, pp. 44–53 (in Russian). DOI: 10.37661/1816-0301-2020-17-3-44-53.
- [20]. Поттосин Ю.В. Синтез комбинационных схем с помощью алгебраической декомпозиции булевых функций. Информатика, том 19, № 1, 2022, стр. 7–18. DOI: 10.37661/1816-0301-2022-19-1-7-18. / Pottosin Yu.V. Synthesis of Combinational Circuits by Means of Bi-Decomposition of Boolean Functions. Informatics, 2022, vol. 19, no. 1, pp. 7–18 (in Russian). DOI: 10.37661/1816-0301-2022-19-1-7-18.
- [21]. Emelyanov P., Ponomaryov D. On the Tractability of Disjoint AND-Decomposition of Boolean Formulas. In Proc. of the PSI 2014: Ershov Informatics Conference. Lecture Notes in Computer Science, vol. 8974, 2015. pp. 92–101. DOI: 10.1007/978-3-662-46823-4_8.
- [22]. Емельянов П.Г., Пономарев Д.К. Алгоритмические вопросы конъюнктивной декомпозиции булевых формул. Программирование, № 3, 2015, стр. 62–72. / Emelyanov P.G., Ponomaryov D.K. Algorithmic Issues of AND-Decomposition of Boolean Formulas. Programming and Computer Software, 2015, vol. 41, no. 3, pp. 162–169. DOI: 10.1134/S0361768815030032.
- [23]. Emelyanov P., Ponomaryov D. On a Polytime Factorization Algorithm for Multilinear Polynomials over F_2 . In Proc. of the 20th International Workshop on Computer Algebra in Scientific Computing. Lecture Note in Computer Science, vol. 11077, 2018. pp. 164–176. DOI: 10.1007/978-3-319-99639-4_11.
- [24]. Emelyanov P., Ponomaryov D. The Complexity of AND-decomposition of Boolean Functions. Discrete Applied Mathematics. 280, 2020, pp. 113–132. DOI: 10.1016/j.dam.2019.07.005.
- [25]. Emelyanov P. AND-Decomposition of Boolean Polynomials with Prescribed Shared Variables. In Proc. of the Second International Conference on Algorithms and Discrete Applied Mathematics. Lecture Note in Computer Science, vol. 9602, 2016. pp. 164–175. DOI: 10.1007/978-3-319-29221-2_14.
- [26]. Emelyanov P. On two kinds of dataset decomposition. In Proc. of the 18th International Conference on Computational Science. Part II. Lecture Notes in Computer Science, vol. 10861, 2018. pp. 171–183. DOI: 10.1007/978-3-319-93701-4_13.
- [27]. Papenbrock T., Ehrlich J., Marten J., Neubert T., Rudolph J., Schoenberg M., Zwiener J., Naumann F. Functional dependency discovery: an experimental evaluation of seven algorithms. Proceedings of the VLDB Endowment, 8(10), 2015, pp. 1082–1093. DOI: 10.14778/2794367.2794377.
- [28]. Кокс Д., Литтл Дж., О’Ши Д. Идеалы, многообразия и алгоритмы. Введение в вычислительные аспекты алгебраической геометрии и коммутативной алгебры: Пер. с англ. Мир, 2000. 687 с.
- [29]. Maple Online Help, Groebner Basis (overview). Available at <https://www.maplesoft.com/support/help/view.aspx?path=Groebner%2fBasis>, accessed 25.10.2025.
- [30]. Faugère J.C. A New Efficient Algorithm for Computing Gröbner Bases (F_4). Journal of Pure and Applied Algebra, 139(1–3), 1999, pp. 61–88. DOI: 10.1016/S0022-4049(99)00005-5.

Информация об авторах / Information about authors

Павел Геннадьевич ЕМЕЛЬЯНОВ – кандидат физико-математических наук, старший научный сотрудник лаборатории смешанных вычислений Института систем информатики им. А.П. Ершова СО РАН с 2006 года. Сфера научных интересов: анализ булевых функций и приложения, теория графов, комбинаторная оптимизация, семантический анализ программ, информационные системы.

Pavel Gennadievich EMEL'YANOV – Cand. Sci. (Phys.-Math.), Senior Researcher of the Laboratory of Mixed Computations of the Institute for Informatics Systems of the SB RAS since 2006. Research interests: analysis of Boolean functions and applications, graph theory, combinatorial optimization, semantic analysis of programs, information systems.

DOI: 10.15514/ISPRAS-2026-38(1)-4



Поиск ошибок в исходном коде на С# на основе статического анализа помеченных данных

М.В. Беляев, ORCID: 0000-0003-3489-3508 <mbelyaev@ispras.ru>

П.И. Рагозина, ORCID: 0000-0003-4219-7203 <pragozina@ispras.ru>

В.Н. Игнатьев, ORCID: 0000-0003-3192-1390 <valery.ignatyev@ispras.ru>

*Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

Аннотация. Ошибки в исходном коде, часто являющиеся причиной уязвимостей, могут быть найдены с помощью разных методов статического анализа. Однако методы на основе анализа помеченных данных имеют ряд преимуществ. В работе предложен комплекс методов, которые в совокупности с алгоритмом на основе IFDS для распространения помеченных данных позволяют превзойти результаты существующих промышленных инструментов. Комплекс методов реализован в промышленном статическом анализаторе SharpChecker и протестирован как на наборе тестов для статических анализаторов, таких как Juliet и WebGoat, для оценки полноты, так и на реальных проектах для оценки точности и производительности. Приведены результаты сравнения с популярными инструментами InferSharp, Security Code Scan, а также коммерческими анализаторами. Показаны относительно высокие значения полноты и точности анализатора SharpChecker. Результаты сравнения позволяют заключить, что предложенный комплекс методов имеет высокую практическую значимость при поиске уязвимостей.

Ключевые слова: статический анализ; анализ помеченных данных; язык С#; поиск уязвимостей; задача анализа потоков данных IFDS; символическое выполнение.

Для цитирования: Беляев М.В., Рагозина П.И., Игнатьев В.Н. Поиск ошибок в исходном коде на С# на основе статического анализа помеченных данных. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 45–60. DOI: 10.15514/ISPRAS-2026-38(1)-4.

Благодарности: Результаты получены с использованием услуг Центра коллективного пользования Института системного программирования им. В.П. Иванникова РАН – ЦКП ИСП РАН.

Search for Errors in C# Source Code Based on Static Taint Analysis

M.V. Belyaev, ORCID: 0000-0003-3489-3508 <mbelyaev@ispras.ru>

P.I. Ragozina, ORCID: 0000-0003-4219-7203 <pragozina@ispras.ru>

V.N. Ignatyev, ORCID: 0000-0003-3192-1390 <valery.ignatyev@ispras.ru>

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

Abstract. Various static analysis methods can detect errors in source code that often lead to vulnerabilities. Tainted data analysis, in particular, offers several advantages. This paper proposes a set of methods that, when combined with an IFDS-based taint propagation algorithm, can outperform existing industrial tools. The method set has been implemented in the industrial-grade SharpChecker analyzer and evaluated on both static analyzer test suites – such as Juliet and WebGoat – to assess completeness, and on real-world projects to measure accuracy and performance. A comparison with popular analyzers, including InferSharp, Security Code Scan, and others, demonstrates SharpChecker's comparatively high completeness and accuracy. These results lead to the conclusion that the proposed method set is of high practical value for vulnerability detection.

Keywords: static analysis; taint analysis; C# language; vulnerabilities detection; IFDS; symbolic execution.

For citation: Belyaev M.V., Ragozina P.I., Ignatyev V.N. Search for errors in C# source code based on static taint analysis. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 45-60 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-4.

Acknowledgements. The results were obtained using the services of the Ivannikov Institute for System Programming (ISP RAS) Data Center.

1. Введение

При разработке программ неизбежно появление ошибок и уязвимостей. Их поиск требует применения специальных инструментов. Существуют различные методы поиска уязвимостей, например, тестирование, статический и динамический анализ, фаззинг. Тестирование зачастую не подходит для обнаружения уязвимостей, так как они проявляются на нестандартных, не предусмотренных программистом данных. Поэтому более оправдан поиск уязвимостей при помощи динамического и статического анализа.

Динамический анализ требует запуска программы, а вместе с ним – настройки окружения и подготовки входных данных. Статический анализ не требует запуска и позволяет проверить все возможные пути выполнения программы. Для поиска уязвимостей могут применяться различные методы статического анализа: проверка файлов конфигурации, сигнатурный поиск, методы на основе символьного анализа и анализа помеченных данных (АПД). В работе [1] сравнивались последние два подхода и была отмечена перспективность методов на основе помеченных данных, которые, однако, обладают рядом недостатков. В данной работе предлагается дополнить базовую реализацию комплексом вспомогательных алгоритмов, обеспечивающих уверенное превосходство над другими решениями, в том числе и на основе АПД.

Анализ помеченных данных – один из широко распространенных методов поиска уязвимостей и дефектов безопасности. Он может быть как статическим [1], так и динамическим [2], и нацелен на отслеживание путей распространения чувствительных данных от *истока* – метода или инструкции, помечающей данные, до *стока* – метода или инструкции, использование помеченных данных в которой является уязвимостью или утечкой. Например, если пользовательский ввод без проверки попадает в вызов метода семейства `exec`, то возможна уязвимость *внедрение команд* (Command Injection [3]). В то время как при динамическом АПД требуется при необходимости отследить опасный путь, которым будут распространяться помеченные данные во время выполнения, задача статического – обнаружение всевозможных таких путей.

Для оценки результатов практически любого метода статического анализа используются следующие метрики: *полнота* – отношение числа верно найденных ошибок к их общему количеству в анализируемом проекте; *точность* – отношение числа верно найденных к общему количеству найденных ошибок; *производительность*. Однако, помимо них, промышленный анализатор кода должен соответствовать еще трем ключевым требованиям: детерминизм, масштабируемость и конфигурируемость [4].

- Детерминизм означает, что анализ должен выдавать идентичные результаты при повторных запусках на одном и том же коде.
- Масштабируемость – это способность анализатора эффективно работать с большими кодовыми базами в миллионы строк кода без экспоненциального роста времени анализа, потребления памяти и количества выданных предупреждений.
- Конфигурируемость – это возможность адаптировать анализатор под нужды конкретного проекта или пользователя, что в контексте АПД означает наличие интерфейса для описания собственных наборов истоков, стоков и других параметров поиска ошибок.

В работе описан метод анализа, реализованный в рамках разрабатываемого в ИСП РАН промышленного статического анализатора языка C# SharpChecker. Этот метод позволяет, учитывая указанные выше требования, достичь результатов, превосходящих другие известные решения, такие как Security Code Scan, InferSharp и другие.

Существуют различные методы реализации анализа помеченных данных. Решение, рассмотренное в данной работе, основано на представлении задачи анализа помеченных данных *IFDS-задачи* [1], то есть задачи межпроцедурного (*Interprocedural*) анализа потока данных с конечным (*Finite*) числом фактов и дистрибутивными (*Distributive*) передаточными функциями, оперирующей подмножествами (*Subset*). Алгоритм решения такой задачи, называемый алгоритмом табуляции, был описан уже в 1995 году [5]. Он позволяет свести IFDS-задачу к задаче достижимости на расширенном межпроцедурном графе потока управления (ГПУ). Однако для использования с целью поиска уязвимостей описанный алгоритм требует решения следующих проблем:

- IFDS-решение крайне трудно масштабируется из-за необходимости полного построения и хранения в памяти межпроцедурного ГПУ;
- на определенных шаблонах кода IFDS-алгоритм не может завершить анализ;
- алгоритм не имеет чувствительности к путям выполнения, что снижает точность анализа;
- алгоритм требует дополнительного анализа косвенных и виртуальных вызовов, иначе снижаются полнота и точность.
- алгоритм требует дополнительного моделирования библиотечных функций, иначе помеченность теряется при обработке таких вызовов.

В связи с этим необходима разработка комплекса вспомогательных методов анализа для повышения показателей анализа на основе IFDS. В работе предложен такой комплекс, описаны его преимущества и недостатки. Оценивается вклад как отдельных компонентов комплекса, так и сравнение итоговых результатов с другими инструментами статического анализа – InferSharp, Security Code Scan и другими.

В разделе 2 данной работы будут более подробно рассмотрены конкретные решения этих проблем, выбранные для анализатора помеченных данных в SharpChecker.

Некоторые из элементов этого комплекса, реализованные в рамках статического анализатора, уже были упомянуты в статьях ([6-7]), однако данная работа впервые охватывает их все и даёт совокупное представление о полученной технологии статического анализа помеченных данных.

2. Схема работы анализатора SharpChecker

Многие из перечисленных выше алгоритмов актуальны не только для анализа помеченных данных, но и для других методов статического анализа. В связи с этим, многие из них уже реализованы в промышленном анализаторе для символьного анализа, а их результаты могут повторно использоваться в АПД без дополнительных накладных расходов.

Работа SharpChecker основывается на компиляторной платформе Microsoft .NET Compiler Platform (Roslyn) [8]. Roslyn предоставляет абстрактное синтаксическое дерево (АСД), таблицу символов и основу для графа потока управления, а SharpChecker использует их для построения графов потока управления всех методов и дальнейшего анализа.

Данный процесс включает следующие этапы [9]:

- 1) Анализ синтаксического дерева: выполняется разбор дерева, выявление явных и неявных вызовов, построение иерархии классов и поиск синтаксических ошибок.
- 2) Построение статического графа вызовов.
- 3) Чувствительный к путям и контексту вызова межпроцедурный анализ на основе символьного выполнения. Производится анализ каждой функции в обратном топологическом порядке по графу вызовов с помощью движка символьного выполнения. Масштабируемый межпроцедурный анализ реализуется с использованием резюме методов для хранения собранной информации. Резюме используются при обработке инструкции вызова вместо анализа кода вызываемого метода. Для каждой функции выполняется:
 - a) построение графа потока управления;
 - b) обход базовых блоков в порядке выполнения, в процессе которого вычисляется состояние памяти программы и генерируются события анализа;
 - c) обработка этих событий детекторами для сбора данных и выполнения проверок.
- 4) Анализ помеченных данных: выполняется на основе описанного выше алгоритма;
- 5) Формирование отчета: на заключительном этапе обработчик выдает предупреждения на основе собранной информации.

Третий этап анализа реализует символьное выполнение с объединением состояний. Каждая функция рассматривается как точка входа в программу. Все переменные, используемые функцией, параметризуются в начальной точке символьными значениями. Затем все инструкции ГПУ символьно исполняются, обновляя символьное состояние памяти программы. В точках слияния путей выполняется объединение состояний памяти с учетом предиката каждого пути, что в некоторых случаях может привести к потере точности. Циклы обходятся три раза: моделируется первая итерация и две «произвольные», перед анализом которых всем измененным в цикле переменным присваиваются новые символьные значения. Такой подход позволяет выполнить анализ большинства возможных путей выполнения функции, не запуская её. В анализаторе SharpChecker метод символьного выполнения и анализ помеченных данных могут запускаться как независимо, так и вместе, а при совместном запуске данные, получаемые с помощью первого метода, могут быть использованы для повышения точности и полноты второго. Подробнее эта интеграция будет описана в дальнейших разделах.

Для работы символьного выполнения, как и для АПД, требуется моделирование библиотечных функций. В анализаторе SharpChecker реализовано сразу несколько методов [7]: база данных аннотаций библиотечных функций и их спецификации на C#. Эти методы и созданные спецификации могут использоваться при поиске уязвимостей с помощью АПД. На этапе символьного анализа для каждой инструкции вызова вычисляется список возможных кандидатов с помощью методов девиртуализации, анализа интерфейсов и

делегатов с учетом информации о типах созданных объектов, вычисляемых при символьном исполнении. Данная информация также доступна на следующем этапе. Кроме этого, резюме методов могут быть сериализованы на диск для использования как на более поздних этапах, так и при следующем запуске.

3. Анализ помеченных данных в SharpChecker

Базовая реализация АПД в SharpChecker описана в статьях [4], [9] и [10]. Подсистема анализа состоит из движка анализа, основанного на алгоритме IFDS, описания набора искомых ошибок, которые задаются в виде правил, описанных ниже, и вспомогательных подсистем, необходимых для эффективного сбора данных для движка.

Правило безопасности для АПД – утверждение о программе, нарушение которого означает наличие потенциального дефекта, связанного с безопасностью программы, – например, утечки пароля, SQL-инъекции. Правила безопасности задают наборы истоков и стоков, а задача проверки таких правил заключается в поиске путей от истоков до стоков путем межпроцедурного анализа помеченных данных.

В общем случае существует два направления анализа путей распространения помеченных данных: прямое и обратное. Нарушения разных правил безопасности могут обнаруживаться более эффективно при выборе того или иного направления. Например, SQL-инъекцию (CWE-89 [11]) проще находить в прямом направлении, отслеживая распространение данных из истока – функции чтения пользовательского ввода – до одного или нескольких SQL-запросов, количество которых в проекте обычно больше. В то же время при поиске константных паролей (CWE-798 [12]) эффективнее использовать обратное направление – от стока (использование строки в качестве пароля) к истоку (создание константы), которых в программе на несколько порядков больше.

Также правило безопасности может быть **составным** – в таком случае для его проверки требуется исследование нескольких связанных между собой путей распространения данных в программе. Такие правила задаются как комбинации простых правил безопасности помеченных данных: в качестве истока или стока в одном правиле используется другое правило.

Формализованное описание правил безопасности помеченных данных в SharpChecker задается в текстовом формате (JSON) и для каждого отдельного правила включает в себя:

- название дефекта;
- направление анализа (прямое или обратное);
- множества истоков и стоков;
- множество *пропагаторов* – специальных передаточных функций, определяющих особенности распространения помеченных данных через внешние функции для данного типа ошибки;
- множество *санитайзеров* – функций, прекращающих распространение помеченных данных, например, экранирование или удаление специальных символов, порождение исключений при обнаружении дефекта.

Анализ начинается при построении графа вызовов. На этом этапе в программе ищутся *начальные точки* анализа – инструкции в коде, соответствующие истокам одного из правил безопасности, с которых начинается распространение. В случае обратного направления анализа начальные точки вычисляются на основе стоков правил.

На основном этапе для каждой найденной точки строятся все возможные пути распространения данных из нее. Для этого анализируется семантика каждой инструкции промежуточного представления кода и сопоставляется с заданным для правила набором передаточных функций, санитайзеров и пропагаторов. При обнаружении новой точки

программы, куда могут попасть помеченные данные, она добавляется в очередь анализа. После завершения обработки текущей точки из очереди берется новая. Для сокращения объемов повторного анализа применяются резюме.

При обнаружении для некоторого правила пути от начальной точки до конечной в список предупреждений добавляется сообщение о нарушенном правиле.

В связи с неограниченной длиной путей в программе вводятся **пределы** – это максимальные значения глубины анализа помеченных данных, необходимые для отсеечения слишком длинных и маловероятных путей. Пределы также могут при необходимости использоваться для сокращения времени анализа.

3.1 Дополняющие методы анализа

Как было отмечено ранее, без моделирования окружения, анализа виртуальных и других косвенных вызовов результаты подхода на основе IFDS не удовлетворяют требованиям, предъявляемым к современным статическим анализаторам. В данном разделе будут подробнее описаны изменения и расширения IFDS-алгоритма анализа помеченных данных, внесенные с целью повышения масштабируемости, точности, полноты и конфигурируемости.

3.1.1 Анализ библиотечных функций

В анализаторе SharpChecker реализовано несколько методов моделирования библиотечных функций, описанных в статье [7]. Анализ помеченных данных использует подход на основе спецификаций. Спецификация библиотечной функции – это «сокращенная» реализация этой функции, имеющая идентичный прототип, код которой содержит только важные для анализа инструкции и не содержит ничего избыточного, как например, вызовов других функций, циклов и т. д. Такие спецификации обычно создаются вручную для часто используемых методов, а также с помощью больших языковых моделей [13]. Они позволяют описать все возможные пути распространения помеченных данных в результате выполнения метода. Отсутствие избыточного кода в спецификациях обеспечивает высокую скорость анализа по сравнению с анализом настоящей реализации, если она доступна, и повышает полноту анализа по сравнению с пропуском библиотечных функций.

Многие функции, например, реализующие работу с коллекциями, такие как `Add` и `Union`, имеют большое количество различных реализаций для разных типов данных, однако сохраняют общую семантику. В коде их вызовы представлены соответствующими интерфейсными методами. Поэтому вместо того, чтобы задавать спецификации для конкретных реализаций интерфейсного метода, для анализа помеченных данных реализована возможность указать путь распространения данных через функции-пропагаторы (которыми могут являться и интерфейсные методы) непосредственно в файле описания правил безопасности. Это также позволяет задавать пути распространения, специфичные для конкретного типа дефекта. Например, в случае поиска деления на ноль, метод `int.Parse` является пропагатором, а для SQL-инъекции – нет, так как угрозу представляют только строки.

Таким образом, моделирование окружения является наиболее важным средством, обеспечивающим полноту, так как в противном случае анализ будет страдать от недопомеченности в результате потери информации в многочисленных вызовах внешних функций.

Наряду с пропагаторами реализована возможность указывать санитайзеры – функции или инструкции, снимающие пометку. Это позволяет запретить отдельные пути распространения для специфичных правил безопасности, что необходимо для поддержки корректной проверки данных перед использованием. Например, при поиске ошибок деления на ноль сравнение помеченного значения с константой может снимать с него пометку.

3.1.2 Девиртуализация

Обработка вызовов, в которых вызываемый метод точно не известен на этапе компиляции, может приводить и к недостаточной, и к избыточной помеченности. В таких вызовах могут находиться как стоки, для которых необходимо выдавать предупреждение, так и санитайзеры, обеспечивающие проверку данных, а также возможно их копирование и распространение пометок.

При анализе кода на C#, для которого характерно широкое использование виртуальных и других косвенных вызовов – например, через делегаты и интерфейсы – эта проблема особенно актуальна. Для ее решения в рамках SharpChecker используются преимущества комплексного анализа кода различными методами: на этапе построения графа вызовов работают простые алгоритмы девиртуализации, которые по иерархии классов сокращают множество допустимых кандидатов виртуального вызова. На этапе символьного выполнения вычисляется еще более точная информация о возможных значениях и типах переменных, на основе которой выбираются наиболее подходящие кандидаты и информация о них сохраняется для последующего анализа помеченных данных. Аналогично выполняются уточнение методов, вызываемых из делегатов и интерфейсов. В случае полного отсутствия анализа или в результате менее точных подходов, характерных для других инструментов, список кандидатов может быть как пустым, так и содержать лишние методы, негативно сказываясь на показателях анализатора.

3.1.3 Объединение начальных точек

Многие правила безопасности имеют общие истоки и похожие списки санитайзеров и пропагаторов, а различаются только набором стоков. Например, пользовательский ввод рассматривается в качестве истока помеченных данных многих правил безопасности, т. к. без должной проверки не должен использоваться ни в SQL-запросах, ни при запуске команд. Количество правил безопасности в SharpChecker на данный момент превосходит 25 и непрерывно растет. Независимое выполнение анализа правил приводит к снижению масштабируемости при увеличении числа детекторов из-за избыточных повторных вычислений. Для повышения масштабируемости с ростом числа детекторов был реализован алгоритм объединения анализа общих для нескольких правил начальных точек. При этом у совместного выполнения есть слабые стороны, например, усложнение реализации и трудности в достижении одинаковых результатов между совместным и раздельным анализом – например, в случае, когда правила имеют одинаковые истоки, но различные наборы пропагаторов и санитайзеров.

В качестве примера правил с общими начальными точками можно рассмотреть вышеупомянутые SQL-инъекцию и инъекцию команды ОС. И в первом, и во втором случае начальными точками (истоками) будут команды пользовательского ввода, также похожим образом будет производиться распространение пометок. Разница состоит только в стоках (SQL-запрос в первом случае и выполнение команды во втором). Реализация совместной обработки истоков ускорила этап анализа помеченных данных более чем в 1.5 раза при тестировании на наборе проектов из 3.4 млн. строк кода. Кроме того, эта оптимизация обеспечила возможность добавления новых детекторов ошибок без существенного влияния на время работы. Это особенно важно для поддержки созданных пользователями детекторов.

3.1.4 Фильтрация недостижимого кода

Не весь код программы, поступающей на вход статическому анализатору, является достижимым при каких-либо входных данных программы. Например, сбор отладочной информации может включаться при определенных параметрах сборки программы, а наличие кода, реализующего сохранение отладочной информации, может быть причиной ложного предупреждения об утечке конфиденциальных данных. Алгоритм IFDS не способен

обеспечить чувствительность к путям выполнения и не позволяет избавиться от ложных предупреждений в недостижимом коде. Однако использование информации о достижимости базовых блоков графа потока управления, собранной на этапе символьного выполнения, помогает фильтровать такие ошибки. Проверка достижимости анализируемого кода не только повышает точность, и обеспечивает чувствительность к путям, но и сокращает затраты ресурсов и времени, если завершать анализ в недостижимых блоках.

На этапе символьного выполнения идентификаторы недостижимых базовых блоков сохраняются в резюме методов. Далее ядро анализа помеченных данных завершает распространение пометок при обработке инструкций недостижимых блоков.

Тестирование на наборе из проектов с открытым исходным кодом показало, что использование данных о недостижимом коде, получаемых при помощи символьного выполнения, приводит не только к исчезновению срабатываний, относящихся к недостижимому коду (не играющих роли в проекте и, как правило, ложных), но и к обнаружению новых уязвимостей, достигнуть которых ранее не позволяло достижение пределов.

3.1.5 Фильтрация методов, не взаимодействующих с помеченными данными

Алгоритм IFDS предполагает последовательную обработку всех инструкций всех методов на пути выполнения. Даже если метод никак не взаимодействует с помеченными данными, приходится обходить все его инструкции, чтобы в этом убедиться. Например, метод, принимающий объект с помеченным полем, может не использовать это поле. К подобным случаям также относятся методы класса, не использующие его помеченные поля. Так, в примере, приведенном на рис. 1, `Bad()` – опасный метод, так как может привести к SQL-инъекции. `Good()` – пример метода, анализ которого может быть пропущен без ущерба для результата.

Использование предварительного простого анализа, например, только абстрактного синтаксического дерева, позволяет заранее определить все используемые методом поля и параметры, чтобы было можно пропустить его обработку целиком, когда он не может повлиять на распространение помеченных данных. Подход, реализованный в `SharpChecker`, состоит из трех этапов:

- 1) На этапе анализа абстрактного синтаксического дерева выполняется сбор информации обо всех использованных методом переменных и их сохранение в резюме.
- 2) Далее выполняется распространение данных по графу вызовов, что обеспечивает перенос информации из резюме всех вызываемых методов в вызывающий.
- 3) При обработке вызова метода на этапе анализа помеченных данных список используемых полей из резюме сопоставляется с помеченными данными. Если никакие помеченные переменные им не используются, анализ этого метода пропускается.

Эта оптимизация позволяет находить больше дефектов, что вызвано более поздним достижением пределов, завершающих анализ. Так, на описанном наборе проектов удалось получить 149 новых истинных срабатываний. Кроме того, пропуск этих функций повышает масштабируемость.

3.2 Пользовательские правила безопасности

Для адаптации анализа к конкретной предметной области и кодовой базе полезной является возможность настройки параметров анализа помеченных данных на стороне пользователя. Помимо редактирования существующих в анализаторе правил безопасности, поставляемых в текстовом формате JSON, пользователи могут создавать собственные правила для отслеживания возможных путей распространения чувствительных данных в своем коде.

В рамках SharpChecker это реализовано путем предоставления пользователю формата описания правил безопасности на основе API, реализованного в ядре анализа помеченных данных. Полнота доступного API подтверждается наличием более 25 существующих детекторов, обладающих высокими показателями точности и полноты. Реализованные в SharpChecker оптимизации ядра анализа обеспечивают масштабируемость и точность работы даже при большом числе точек начала анализа.

```
1 class C {
2     SqliteCommand c = new();
3     string taintedStr;
4     int i;
5     public void Bad() {
6         string query = "SELECT* FROM table WHERE name = '" + taintedStr + "'";
7         // переменная query помечена
8         c.CommandText += query; // помеченная переменная query попала в сток
9     }
10    public void Good(int arg) => i = arg;
11    public void F() {
12        taintedStr = Console.ReadLine(); // (!SQL_INJECTION)
13        // переменная this.taintedStr помечена
14        Good(3);
15        Bad(); // c.CommandText помечен
16        c.ExecuteReader(); // выполнение опасного запроса
17    }
18 }
```

Рис. 1. Метод Bad(), распространяющий помеченные данные, и метод Good(), не влияющий на их распространение.

Fig. 1. The Bad() method that propagates the tainted data and the Good() method that does not affect its propagation.

3.3 Поддержка Visual Basic

Язык программирования Visual Basic (VB.NET), несмотря на свою популярность (7-е место в рейтинге TIOBE [14]), остается слабо охваченным промышленными анализаторами, которые фокусируются на C#. Это создает пробел в инструментах для анализа legacy-проектов на VB.NET, которые часто содержат скрытые ошибки и уязвимости. Кроме того, основной компилятор языков C# и VB.NET Roslyn содержит большое количество кода на VB.NET и требует проведения полного анализа для возможности его применения в ответственных проектах, например, при сертификации. В связи с этим, поддержка языка реализована в SharpChecker [15], включая подсистему анализа помеченных данных.

Внедрение поддержки VB.NET в SharpChecker основано на использовании общей компиляторной инфраструктуры Roslyn, поддерживающей оба языка. Это упростило добавление поддержки VB.NET для большей части детекторов уязвимостей, за исключением относящихся к символьному выполнению: для них потребовалось учитывать различия в представлении символов между языками, такие как организация свойств и их внутренних полей.

Поскольку анализ помеченных данных в SharpChecker реализован на унифицированных структурах данных Roslyn, как например IOperation, его адаптация к VB.NET не потребовала значительных изменений ни ядра, ни правил безопасности. После доработки построения графа вызовов подсистема автоматически начала корректно работать с кодом на VB.NET. Провести сравнительное тестирование для этого языка программирования оказалось трудно ввиду его мало распространенной поддержки анализаторами C#. Тем не менее на проектах с открытым исходным кодом он показывает высокую точность. Так, на проекте EVE-IPN [16] (игровая утилита с открытым исходным кодом) обнаружено 342 срабатываний, все они истинные.

Табл. 1. Иллюстрация связи дополнений к классическому IFDS-анализу и аспектов оценки статического анализатора, которые они повышают. Поля, которые соответствуют связанным дополнению и аспекту, отмечены символом «+».

Table 1. An illustration of the relationship between the additions to the classical IFDS analysis and the aspects of the static analyzer evaluation improved by them. Fields that correspond to the associated addition and aspect are marked with a "+" symbol.

	Масштабируемость	Точность	Полнота	Конфигурируемость	Чувствительность к путям
Анализ библиотечных функций	+		+		
Девиртуализация		+	+		
Объединение начальных точек	+				
Фильтрация недостижимого кода		+	+		+
Фильтрация методов, не взаимодействующих с помеченными данными	+		+		
Пользовательские детекторы				+	
Поддержка Visual Basic			+		

4. Сравнение с существующими решениями

Основными метриками качества анализатора являются полнота и точность. Если точность можно оценить после трудоемкой разметки предупреждений, выданных анализатором, то для оценки полноты необходима информация о всех ошибках в коде, которая доступна только для тестов. Поэтому для оценки полноты используются специальные наборы тестов для статических анализаторов. Для языка C# существует версия Juliet [17], состоящая из множества различных небольших тестовых программ с размеченными уязвимостями. Помимо этого, известны проекты с внедренными уязвимостями WebGoat.NET [18], PumaPrey [19] и другие. Для оценки полноты на реальных проектах можно выполнить сравнение с другими инструментами анализа. Наиболее известные анализаторы, заявляющие поддержку поиска ошибок, реализованных в SharpChecker на основе анализа помеченных данных, описаны ниже.

4.1 InferSharp

InferSharp [20-21] – это статический анализатор кода для C#. Он специализируется на межпроцедурном анализе потоков данных для обнаружения уязвимостей безопасности и включает в себя в том числе и анализ помеченных данных, однако он не использует напрямую алгоритмы решения IFDS-задач. Авторы разработали собственную архитектуру статического анализа помеченных данных, которая решает схожие проблемы другими методами. Анализируется не исходный код, а скомпилированные бинарные файлы.

Описанные в данной работе особенности присутствуют в InferSharp в следующем виде:

- поддерживается девиртуализация;
- недостижимый код пропускается и не обрабатывается анализом помеченных данных, при этом используются сложные межпроцедурные алгоритмы его поиска;
- архитектура InferSharp не предполагает добавления пользовательских правил безопасности для анализа помеченных данных.

4.2 Security Code Scan

Security Code Scan [22] – специализированный для платформы .NET инструмент статического анализа кода на C# и VisualBasic.NET. Как и SharpChecker, он использует возможности компилятора Roslyn для анализа исходного кода. Анализатор поддерживает поиск уязвимостей и проблем безопасности, в том числе с помощью анализа помеченных данных. Однако анализ выполняется без использования IFDS, вместо этого производится анализ на основе символьного выполнения, чувствительный к путям, но с ограниченной контекстной чувствительностью. Среди основных особенностей можно отметить отсутствие девиртуализации и учета недостижимого кода. Инструмент позволяет пользователям с помощью XML-конфигурации создавать собственные правила с конфигурируемыми истоками, стоками и санитайзерами, но, в отличие от SharpChecker, не позволяет настраивать пропагаторы или создавать более сложную логику распространения.

4.3 SonarQube

SonarQube [23] – платформа с открытым исходным кодом для непрерывного анализа и измерения качества программного кода для большого набора языков, в том числе и C#. Существует также коммерческая версия анализатора, реализующая более глубокий анализ при поиске ошибок безопасности. Анализ помеченных данных входит в набор возможностей анализатора, с алгоритмами, также основанными на IFDS-подходе.

- Девиртуализация отсутствует, межпроцедурность очень ограничена.
- Проверки на недостижимость при анализе помеченных данных не производятся, недостижимые истоки/стоки/пути анализируются.
- Анализатор предоставляет пользователю возможность модифицировать некоторые из существующих правил под свои нужды, а также создавать собственные. При создании детекторов предоставляются очень широкие возможности, в том числе полный контроль над источниками, стоками, пропагаторами, расширенные паттерны распространения и сложные сценарии.

4.4 Анализатор CP

Анализатор CP – это пакет коммерческого программного обеспечения, состоящий из статического и динамического анализаторов кода, производящий поиск ошибок и уязвимостей в исходных кодах программ, написанных на Си, C++, Java, C# и JavaScript.

- Активно используется девиртуализация.
- Код может проверяться на достижимость, но недостижимый код тоже анализируется.
- Анализатор предоставляет широкие возможности для создания пользовательских детекторов ошибок.

4.5 Результаты тестирования

Из-за различий в реализациях между SharpChecker и другими анализаторами произвести абсолютно точное сравнение между ними не представляется возможным. Так, независимый запуск анализа помеченных данных и анализа методом символьного выполнения, реализованный в SharpChecker, невозможен в других анализаторах, что делает сравнение времени выполнения непоказательным. Кроме того, было решено сравнивать только результаты анализа помеченных данных, однако при этом возникла проблема отсутствия четкого обособления в некоторых детекторах этой категории ошибок. Так, в SonarQube ввиду ограниченной межпроцедурности при поиске уязвимости вида *SQL-инъекция* обнаруживаются не пути от истока к стоку, а только возможные стоки (использование поступившей извне метода переменной в SQL-запросе). Аналогично встречаются такие

«упрощенные» предупреждения и для нескольких других типов уязвимостей. Анализ помеченных данных при этом не обособляется как отдельная категория.

В InferSharp и Security Code Scan удалось явно выделить типы ошибок, связанные с анализом помеченных данных, а для анализатора SonarQube было решено использовать для сравнения типы ошибок, как правило относящиеся к анализу помеченных данных. При этом «упрощенное» предупреждение считалось истинным, если отмеченный сток в действительности может быть достигнут из какого-либо истока.

Упомянутое выше отсутствие в части анализаторов девиртуализации оказало влияние на количество срабатываний, так как интерфейсы и абстрактные классы часто встречаются в проектах на C#.

Анализатор InferSharp лучше работает с более простыми программами, а на проектах с открытым кодом и тестах Juliet его анализ помеченных данных не обнаружил ни одной ошибки несмотря на то, что на небольших синтетических тестах он зачастую показывает более точный результат по сравнению с другими описанными детекторами за счет применения девиртуализации и поиска недостижимых участков кода.

Все анализаторы на приведенных проектах показывают детерминированный результат.

Табл. 1 и 2 демонстрируют, что, хотя точность SharpChecker не абсолютна, она превышает точность большинства других детекторов на обоих представленных проектах. Кроме того, покрытие данного анализатора наиболее широкое из рассматриваемых. Табл. 3 демонстрирует результаты анализа рассматриваемых анализаторов на размеченном наборе синтетических тестов Juliet test suite (версия 1.3), что позволило получить предположительные значения точности и полноты анализа помеченных данных. Можно заметить, что SharpChecker продемонстрировал по совокупности сравнительно высокие показатели точности и полноты, а также широкий набор правил анализа. При этом Security Code Scan более точен, но проигрывает в полноте. Кроме того, для некоторых анализаторов, в частности в случае анализатора CP, малое количество или отсутствие срабатываний для ряда правил связано с более узким набором стоков для них, нежели предполагается тестами.

Табл. 1. Результаты анализа намеренно небезопасного проекта WebGoat [18].

Table 1. Results of analysis of the intentionally insecure WebGoat [18] project.

	Всего	TP	Точность
SharpChecker	131	103	78,6%
InferSharp	0	0	—
Security Code Scan	36	18	50%
SonarQube	42	29	69%
Анализатор CP	121	105	87%

Табл. 2. Результаты анализа проекта OpenSimulator [24], содержащего наибольшее количество ошибок из тех рассматриваемых проектов с открытым кодом, которые удалось запустить без ошибок на всех четырёх анализаторах.

Table 2. The results of the analysis of the OpenSimulator project [24], which contains the highest number of errors among the considered open-source projects that were successfully run without errors on all four third-party analyzers.

	Всего	TP	Точность
SharpChecker	598	418	70%
InferSharp	0	0	—

Security Code Scan	0	0	–
SonarQube	144	92	64%
Анализатор CP	397	112	28%

Табл. 3. Подробные результаты анализа помеченных данных на тестах Juliet.

Table 3. Detailed results of the taint analysis of the Juliet tests.

Тип дефекта	SharpChecker			Security Code Scan			SonarQube			Анализатор CP		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
Code injection	570	540	0	–			–			0	0	570
Command injection	570	0	0	186	0	384	–			208	0	362
Cross site redirect	513	0	0	186	0	327	–			208	2	305
Division by zero	2128	2820	266	–			–			60	72	2234
Format string injection	798	28	0	–			–			0	0	798
Hardcoded password	57	0	57	17	0	97	4	0	110	83	0	31
Information exposure	51	0	51	–			–			–		
Insecure randomness	0	0	34	–			–			–		
Key management	74	0	0	–			0	17	74	44	3	30
LDAP injection	570	0	0	186	0	384	–			364	7	206
Log injection	0	0	570	–			–			0	0	570
Missing encryption	121	526	1398	–			–			168	1894	1391
Path traversal	870	0	270	322	0	818	–			348	8	792
Process control	0	0	17	–			–			–		
XSS	1824	0	228	558	0	1494	–			0	0	2052
Setting manipulation	570	0	0	–			–			–		
SQL injection	1539	54	0	420	0	1119	684	864	855	510	10	1029

Тип дефекта	SharpChecker			Security Code Scan			SonarQube			Анализатор CP		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
Uncontrolled memory allocation	1824	0	245	–			–			–		
Uncontrolled resource consumption	2242	80	0	–			–			–		
Unsafe cookie usage	–			–			17	263	17	17	98	17
Unsafe reflection	513	0	0	–			–			–		
XPath injection	570	18	0	210	308	360	–			204	213	366
Итого	15404	4066	3136	2085	308	4983	705	1144	1056	2214	2307	10753
Полнота	83%			30%			40%			17%		
Точность	79%			87%			38%			49%		

5. Заключение

Разработанный подход позволил достичь высоких относительно других существующих решений полноты и точности, что подтверждено экспериментальными результатами. Это было достигнуто благодаря сочетанию IFDS-анализа в качестве основы с рядом новых подходов, заполняющих пробелы IFDS-абстракции. Кроме того, сделаны шаги для повышения детерминизма, масштабируемости, конфигурируемости.

Перспективы дальнейших исследований видятся в расширении поддерживаемых языков программирования, углублении межпроцедурного анализа и разработке новых адаптивных механизмов настройки параметров анализа. Реализация этих направлений позволит еще больше повысить полноту и точность метода, а также расширить область его практического применения.

Список литературы

[1]. Belyaev M. V. et al. Comparative analysis of two approaches to static taint analysis //Programming and Computer Software. – 2018. – Т. 44. – №. 6. – С. 459-466.

[2]. Кулямин В. В. Обзор методов динамического анализа программного обеспечения //Труды Института системного программирования РАН. – 2023. – Т. 35. – №. 4. – С. 7-44.

[3]. The MITRE Corporation, CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection'), MITRE CWE. Available at: <https://cwe.mitre.org/data/definitions/78>, accessed 10.10.2025.

[4]. Ignatyev V. N. A Holistic Static Analysis for Finding Errors in Source Code //Программная инженерия. – 2025. – Т. 16. – №. 10. – С. 517-531.

[5]. Reps T., Horwitz S., Sagiv M. Precise interprocedural dataflow analysis via graph reachability //Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages. – 1995. – С. 49-61. DOI: 10.1145/199448.199462.

- [6]. Shimchik N. V., Ignatyev V. N., Belevantsev A. A. Improving accuracy and completeness of source code static taint analysis //2021 Ivannikov Ispras Open Conference (ISPRAS). – IEEE, 2021. – С. 61-68.
- [7]. Biktimirov M. G., Ignatyev V. N., Belyaev M. V. Improving the accuracy of library function modeling in the static analyzer //2023 Ivannikov Ispras Open Conference (ISPRAS). – IEEE, 2023. – С. 26-32.
- [8]. The Roslyn .NET compiler provides C# and Visual Basic languages with rich code analysis APIs. Available at: <https://github.com/dotnet/Roslyn>, accessed 10.10.2025.
- [9]. Кошелев В. К., Игнатьев В. Н., Борзилов А. И. Инфраструктура статического анализа программ на языке C //Труды Института системного программирования РАН. – 2016. – Т. 28. – №. 1. – С. 21-40. DOI: 10.15514/ISPRAS-2016-28(1)-2.
- [10]. Koshelev V. K. et al. SharpChecker: Static analysis tool for C# programs //Programming and Computer Software. – 2017. – Т. 43. – №. 4. – С. 268-276.
- [11]. The MITRE Corporation, CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'), MITRE CWE. Available at: <https://cwe.mitre.org/data/definitions/89>, accessed 10.10.2025.
- [12]. The MITRE Corporation, CWE-798: Use of Hard-coded Credentials, MITRE CWE. Available at: <https://cwe.mitre.org/data/definitions/798>, accessed 10.10.2025.
- [13]. Ignatyev V. N. et al. Large language models in source code static analysis //2024 Ivannikov Memorial Workshop (IVMEM). – IEEE, 2024. – С. 28-35.
- [14]. TIOBE Index - TIOBE. Available at: <https://www.tiobe.com/tiobe-index/>, accessed 08.11.2025.
- [15]. Карцев В. С., Игнатьев В. Н. Поддержка Visual Basic. NET в статическом анализаторе SharpChecker //Труды Института системного программирования РАН. – 2024. – Т. 36. – №. 3. – С. 49-62. DOI: 10.15514/ISPRAS-2024-36(3)-4
- [16]. Code for the EVE Isk per Hour program. Available at: <https://github.com/EVEIPH/EVE-IPH>, accessed 27.11.2025.
- [17]. Black P. E. Juliet 1.3 test suite: Changes from 1.2. – Gaithersburg, MD, USA : US Department of Commerce, National Institute of Standards and Technology, 2018.
- [18]. WebGoat.NET is a deliberately insecure application. Available at: <https://github.com/jerryhoff/WebGoat.NET>, accessed 25.11.2025.
- [19]. Puma Prey contains vulnerable .NET applications, which provide a target for running secure coding challenges, CTFs, and testing the Puma Scan analyzers. Available at: <https://github.com/pumasecurity/puma-prey>, accessed 25.11.2025.
- [20]. Infersharp. Available at: <https://github.com/microsoft/infersharp>, accessed 10.10.2025.
- [21]. Calcagno C., Distefano D. Infer: An automatic program verifier for memory safety of C programs //NASA Formal Methods Symposium. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – С. 459-465. DOI: 10.1007/978-3-642-20398-5_33
- [22]. security-code-scan: Vulnerability Patterns Detector for C# and VB.NET. Available at: <https://github.com/security-code-scan/security-code-scan>, accessed 15.10.2025
- [23]. SonarQube. Available at: <https://github.com/SonarSource/sonarqube>, accessed 15.10.2025.
- [24]. OpenSimulator is an open source multi-platform, multi-user 3D application server. Available at: http://opensimulator.org/wiki/Main_Page, accessed 01.11.2025.

Информация об авторах / Information about authors

Михаил Владимирович БЕЛЯЕВ – младший научный сотрудник ИСП РАН. Научные интересы: компиляторные технологии, статический анализ программ, анализ помеченных данных.

Mikhail Vladimirovich BELYAEV is a Junior Researcher at ISP RAS. Research interests: compiler technologies, static code analysis, taint analysis.

Полина Ильинична РАГОЗИНА – старший лаборант ИСП РАН, исследователь в отделе компиляторных технологий. Научные интересы: статический анализ программ, символьное выполнение, анализ помеченных данных.

Polina Ilyinichna RAGOZINA is a Senior Laboratory Assistant at ISP RAS, a researcher at the Department of Compiler Technologies. Scientific interests: static analysis of programs, symbolic execution, taint analysis.

Валерий Николаевич ИГНАТЬЕВ – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент кафедры системного программирования факультета ВМК МГУ. Научные интересы включают методы поиска ошибок в исходных текстах программ на основе статического анализа.

Valery Nikolayevich IGNATYEV – Cand. Sci. (Phys.-Math.) in computer sciences, senior researcher at Ivannikov Institute for System Programming RAS and associate professor at system programming division of CMC faculty of Lomonosov Moscow State University. His research interests include program analysis techniques for error detection in program source code using classical static analysis and machine learning.

DOI: 10.15514/ISPRAS-2026-38(1)-5



Версионирование кода как развитие оптимизации методом Распространения констант

И.А. Зинин, ORCID: 0009-0000-5287-1543 <zinin.ia@phystech.edu>

В.В. Черноног, ORCID: 0009-0007-8407-1082 <chernonog.vv@phystech.edu>

*Московский физико-технический институт,
141701, Московская область, г. Долгопрудный, Институтский переулок, д. 9.*

Аннотация. В работе рассмотрены принципы устройства оптимизации методом Распространения констант, играющей большую роль во многих современных оптимизирующих компиляторах. Исследована техника версионирования кода и предложен алгоритм, улучшающий метод Распространения констант. Реализация алгоритма была осуществлена в компиляторе GCC. Результатами, подтверждающими важность и актуальность данной работы, являются замеры работы вычислительно сложных приложений из пакета задач CPUBench.

Ключевые слова: оптимизации в компиляторе; распространение констант; межпроцедурное распространение констант; версионирование.

Для цитирования: Зинин И.А., Черноног В.В. Версионирование кода как развитие оптимизации методом Распространения констант. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 61–70. DOI: 10.15514/ISPRAS-2026-38(1)-5.

Code Versioning as an Evolution of Constant Propagation Optimization

I.A. Zinin, ORCID: 0009-0000-5287-1543 <zinin.ia@phystech.edu>

V.V. Chernonog, ORCID: 0009-0007-8407-1082 <chernonog.vv@phystech.edu>

*Moscow Institute of Physics and Technology,
9, Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*

Abstract. This paper examines the principles and structure of the Constant Propagation optimization, which plays a significant role in many modern optimizing compilers. The technique of code versioning has been studied, and an algorithm has been proposed to improve Constant Propagation. The algorithm was implemented in the GCC compiler. The results, confirming the importance and relevance of this work, were obtained through performance measurements of computationally heavy applications from the CPUBench benchmark suite.

Keywords: compiler optimizations; constant propagation; interprocedural constant propagation; versioning.

For citation: Zinin I.A., Chernonog V.V. Code versioning as an evolution of Constant Propagation optimization. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 1, 2026, pp. 61-70 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-5.

1. Введение

В современном мире оптимизирующие компиляторы являются неотъемлемой частью разработки программного обеспечения, позволяя разработчикам писать код на высокоуровневом языке и получать эффективный машинный код для любой из множества целевых архитектур. Создание эффективных оптимизаций выступает ключевой задачей для современных оптимизирующих компиляторов, а внедрение новых подходов, повышающих быстродействие программного кода, вызывает значительный научно-практический интерес. В данной работе рассмотрена оптимизация методом *распространения констант*, особенности её работы, целью работы является разработка алгоритма, основанного на версионировании кода, который способен расширить возможности данной оптимизации.

2. Распространение констант

Метод оптимизации *распространением констант* заключается в замене переменных, значения которых могут быть определены на этапе компиляции (статически), на соответствующие константы. Данная оптимизация способна избавить от лишних вычислений и копирований, выполняемых во время исполнения программы, а также открыть больше возможностей для анализа другим оптимизациям.

2.1 Виды распространения констант

Во многих современных оптимизирующих компиляторах, в том числе и в GCC [1], оптимизация *распространением констант* бывает двух видов: *внутрипроцедурная* и *межпроцедурная*. *Внутрипроцедурное распространение констант* [2], которое часто называют просто *распространением констант*, ограничивается графами потока данных и потока управления каждой процедуры, рассматриваемой как отдельной. *Межпроцедурное распространение констант* [3] рассматривает поток данных в рамках взаимодействия всех процедур, используемых в программе, опираясь на граф вызовов.

2.2 Внутрипроцедурное распространение констант

Рассмотрим краткое описание работы *внутрипроцедурного распространения констант*. Данная оптимизация активно работает с понятиями полурешеток и передаточных функций. В качестве оператора сбора вводится оператор слияния $\sqcup$ (join), в качестве верхнего элемента $\top$ берется неопределенное значение (undef), а в качестве нижнего $\perp$ берется переопределённое значение (overdef). Передаточной функцией базового блока берется функция, определяющая изолированное действие этого базового блока на вектор переменных.

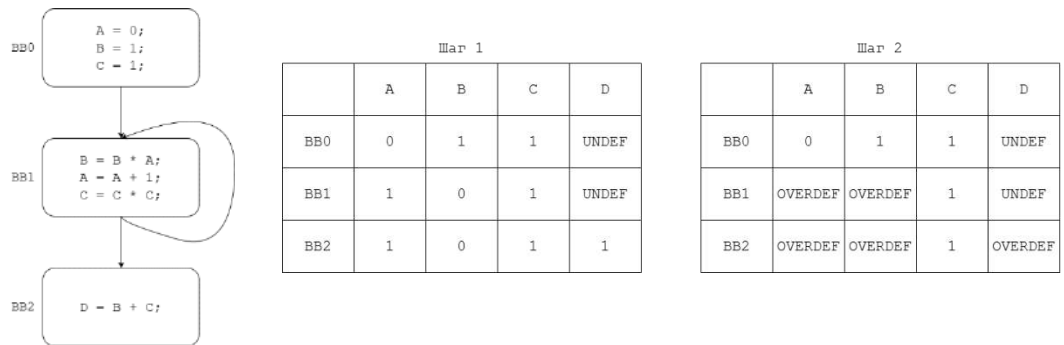


Рис. 1. Пример исполнения алгоритма Распространения Констант.
Fig. 1. Example of the Constant Propagation algorithm execution.

Для графа потока управления, изображенного на рис. 1, передаточные функции базовых блоков выглядят соответствующим образом:

- $F_{BV0}(A, B, C, D) = (0, 1, 1, D)$
- $F_{BV1}(A, B, C, D) = (A+1, B*A, C*C, D)$
- $F_{BV2}(A, B, C, D) = (A, B, C, B+C)$

Работа оптимизации начинается с выставления значения верхнего элемента (undef) всем переменным. Затем производится итеративное исполнение передаточных функций до момента схождения к неподвижной точке, то есть состояния, когда все $F_{BV_i}(X) = X$ для каждой из переменных. Переменные принявшие значение нижнего элемента (overdef) не могут быть заменены на константу, а те, что приняли какое-то одно константное значение, можно заменить на соответствующие константы. Например, на рис. 1 на последнем шаге алгоритм выявил, что для переменной C может быть распространено константное значение 1.

2.3 Межпроцедурное распространение констант

Алгоритм *межпроцедурного распространения констант* аналогичен *внутрипроцедурному*, но имеет следующие отличия: передаточными функциями становятся передаточные функции каждого отдельного процедурного вызова; анализируются только переменные, являющиеся аргументами вызовов; распространение выполняется на графе вызовов программы. Данная оптимизация в случае установления константности одного из аргументов вызова способна произвести клонирование соответствующей процедуры и подставить вместо аргумента константное значение.

Клонирование в случае *межпроцедурного распространения констант* выступает аналогией версионирования. Клоны процедуры с соответствующими возможными константными значениями являются версиями исходной процедуры, адаптированными под конкретный сценарий исполнения. В коде компилируемой программы вместо старых вызовов подставляются вызовы клонов.

3. Проблемы существующих решений

Несмотря на глубину анализа и большое практическое применение, существует ряд проблем, ограничивающий возможности оптимизации распространением констант:

- 1) *Распространение констант* как *внутрипроцедурное*, так и *межпроцедурное*, заменяет переменную на константу, если способно гарантировать единственность данного значения. Но зачастую переменная может принимать некоторый дискретный набор возможных константных значений.
- 2) Существующий анализ области значений переменной (Value Range Analysis) [4], реализованный в том числе и в GCC, способен отслеживать интервал значений переменной. Однако анализ не способен отслеживать дискретный набор значений, а только лишь непрерывные интервалы.
- 3) *Межпроцедурное распространение констант* не работает с графом потока управления в явном виде, потому может упускать некоторые особые случаи, подходящие для трансформации.

3.1 Проблема межпроцедурного распространения констант

Для того, чтобы разобрать проблему *межпроцедурного распространения констант*, необходимо обратиться к примеру, изображенному на рис. 2. В данном примере кода присутствует функция *func_0*, которая содержит внутри себя вызов другой функции *func_1* при выполнении сложного (комплексного) условия. Условие состоит из двух простых

условий проверки равенства значения переменной n какой-либо константе. Эта переменная является полем структуры *tree*, которая выступает аргументом вызова функции *func_1*. Далее код программы из функции *func_0* может привести исполнение к вычислительной функции *compute_func*, которая является “горячей”, то есть исполняется большое число раз. Внутри себя она содержит два вложенных цикла, числом итераций которых и является переменная n , переданная в качестве аргумента. Если во время компиляции можно было бы определить, что значение n является константным, тогда многие методы оптимизации, например, *раскрутка циклов* (Loop Unrolling) [5] или *векторизация циклов* (Loop Vectorization) [6], смогли бы получить больше возможностей для анализа и произвести необходимые трансформации.

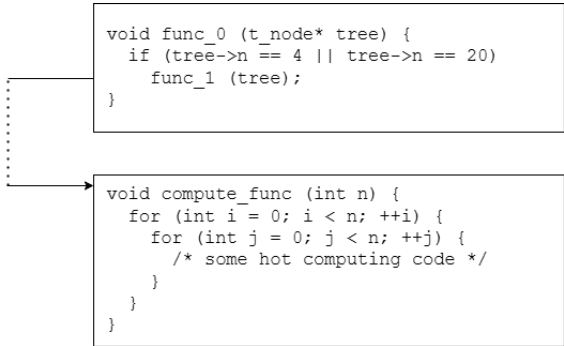


Рис. 2. Пример кода, не обрабатываемый межпроцедурным Распространением Констант.
Fig. 2. Example of a code that is not processed by interprocedural Constant Propagation.

Так как значение n для вызова *func_1* в *func_0* не определено однозначно, существующий алгоритм межпроцедурного *распространения констант* не может распространить ни одно из возможных значений. Потому тело *compute_func* не может быть оптимизировано и его выполнение будет занимать большую часть времени и вычислительных ресурсов.

Согласно семантике языка C, если по ходу проверки простых условий, связанных логическим “ИЛИ” в комплексном условии, выполняется какое-то из них, то управление сразу передается к базовому блоку “then”. А это означает, если в исходном коде *func_0* произвести версионирование, как показано на рис. 3, то поведение программы никак не изменится. Причем условия не обязательно должны быть взаимоисключающими, и сложное условие, соединяющее простые условия логическим “ИЛИ”, можно представить, как последовательность “if-elseif” простых условий.

```
void func_0 (t_node* tree) {
    if (tree->n == 4)
        func_1 (tree);
    else if (tree->n == 20)
        func_1 (tree);
}
```

Рис. 3. Пример возможного версионирования.
Fig. 3. Example of possible versioning.

При данном варианте функции *func_0* для каждого вызова *func_1* будет однозначно определённое константное значение переменной n (4 или 20). Межпроцедурное *распространение констант* для каждого из значений способно создать клоны *func_1*, *compute_func* и других функций, лежащих в графе вызовов на пути от *func_1* до *compute_func*. Таким образом, каждый из клонов *compute_func* будет содержать свое константное значение, которое обеспечит другим методам оптимизации больше свободы для осуществления трансформаций кода.

3.2 Проблема внутрипроцедурного распространения констант

Нередко такие вычислительные функции, как `compute_func`, являются “горячими”, но внутри себя содержат небольшой объем кода. В таких случаях большая часть вычислительных ресурсов во время исполнения программы тратится на осуществление вызова функции и возврата из неё. Эту проблему призвана решить оптимизация методом *встраивания процедур* (Inlining), которая в коде программы подставляет вместо вызовов функций непосредственно их код. Такая оптимизация имеет собственное ограничение: если объем кода увеличивается слишком сильно, то встраивание не осуществляется. Таким образом, если в пользовательской программе есть часто вызываемая функция и она небольшого размера, то её можно встроить в граф потока управления вызывающей функции.

В GCC есть два оптимизационных прохода *встраивания процедур*. *Раннее встраивание процедур* (Early Inlining) производится одним из первых среди оптимизационных проходов, работающих с графом потока управления в рамках одной единицы трансляции. Макроскопический вариант данной оптимизации, который включает в себя сложный межпроцедурный анализ, осуществляет свою работу в отложенном режиме LTO (Link Time Optimization) [7]. LTO объединяет несколько единиц трансляции в одну и с ней, как с единым целым, позволяет остальным оптимизациям производить трансформации.

Если в коде `func_0` осуществится встраивание, как показано на рис. 4, тогда произвести версионирование будет невыгодно из-за значительного увеличения кода. Кроме того, в общем случае, версионирование лишь из-за возникновения подобного сложного условия чаще всего не будет улучшать производительность программы. Преимущество подхода, показанного на рис. 3, заключается в том, что оптимизации размера кода производятся вместе с более релевантными стадиями, такими как *встраивание процедур* или *межпроцедурное распространение констант*. В свою очередь, эти оптимизации, основываясь на своих эвристиках, будут осуществлять необходимое встраивание или клонирование. Соответственно, если осуществить версионирование, как на рис. 3, потом предоставить работу *встраиванию процедур*, а затем выполнить *внутрипроцедурное распространение констант*, то подобное версионирование способно качественно помочь в распространении соответствующих констант.

```
void func_0 (t_node* tree) {
    if (tree->n == 4 || tree->n == 20) {
        // some other code

        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                /* some hot computing code */
            }
        }

        // some other code
    }
}
```

Рис. 4. Пример процедурного встраивания.

Fig. 4. Example of procedural inlining.

4. Реализация

Так как алгоритм, который будет представлен дальше, не зависит от конкретной платформы, он будет работать с высокоуровневым промежуточным представлением GIMPLE [8]. GIMPLE – это независимое от языка и целевой архитектуры трёхадресное промежуточное представление в компиляторе GCC, которое упрощает оптимизацию, представляя вычисления в виде последовательности базовых операций и превращая управляющие конструкции в условные переходы.

В конвейере оптимизационных проходов компилятора представленный проход вызывается до запуска основных межпроцедурных. В первую очередь, это обусловлено тем, что данное версионирование увеличивает число обрабатываемых участков кода *межпроцедурным распространением констант*, которое, в свою очередь, в режиме LTO способно дать больше возможностей для анализа *раскрутке циклов* (Loop Unrolling), *циклической векторизации* (Loop Vectorization) и другим макро-оптимизациям. Во-вторых, оптимизация *встраиванием процедур* (Inlining) и её ранняя версия (Early Inlining), благодаря подстановке кода функции вместо соответствующих вызовов, способны сильно ограничить число подходящих для версионирования шаблонов кода.

Ознакомиться с исходным кодом оптимизационного прохода можно в репозитории openEuler/gcc [9].

4.1 Подходящие шаблоны кода

Ниже определены подходящие шаблоны кода, к которым алгоритм будет применять версионирование:

- 1) Алгоритм находит сложные условия, объединяющие простые условия логическим “ИЛИ”.
- 2) Одно из простых условий является проверкой на равенство значения переменной константе.
- 3) Эта переменная является аргументом вызова функции в базовом блоке “then”, причем явным, или полем какой-либо структуры, являющейся аргументом функции. В виду возможности неограниченной вложенности структур (nesting), алгоритм учитывает до двух таких уровней (причина выбора данного числа будет упомянута дальше).

4.2 Первый шаг оптимизации

В компиляторе GCC сложные условия могут быть представлены либо явной проверкой в одном базовом блоке, либо двумя базовыми блоками (рис. 5), образуя “if-elseif” конструкцию (рис. 6).

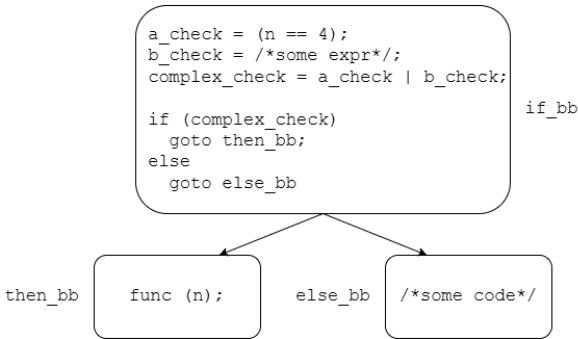


Рис. 5. Пример сложного условия в одном базовом блоке.
Fig. 5. Example of complex condition in one basic block.

Представление проверки сложного условия в явном виде в одном базовом блоке является полезным для осуществления мелкозернистых арифметических операций над *complex_cond*. Так как после предыдущих компиляторных проходов могут встретиться оба варианта, необходимо реализовать первый шаг оптимизации, заключающийся в нахождении всех сложных условий, представленных одним базовым блоком и удовлетворяющих условиям

подходящего шаблона, и трансформации их к виду двух базовых блоков, образующих цепочку “if-elseif”.

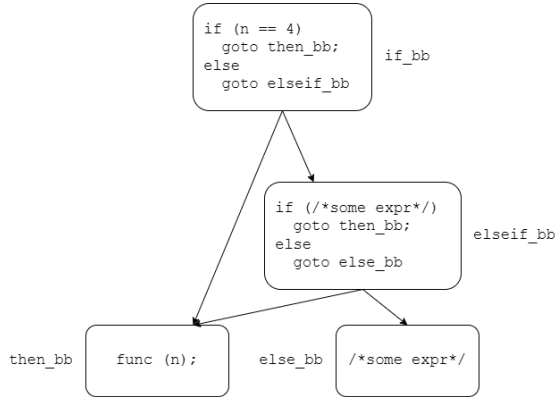


Рис. 6. Пример сложного условия в виде конструкции из двух базовых блоков.

Fig. 6. Example of complex condition in form of two basic blocks.

Для этого алгоритм сначала находит все базовые блоки, содержащие сложное условие, удовлетворяющее Форме 1, среди них отбирает те, у которых одно из простых условий удовлетворяет Форме 2 (рис. 7). Затем он проверяет, что переменная, фигурирующая в проверке на равенство константе, либо является непосредственно аргументом в процедурном вызове в базовом блоке “then”, либо полем структуры, являющейся аргументом. Отметим, что алгоритм учитывает возможность вложенности переменной в структуру как по указателю ($s \rightarrow n$), так и явно ($s.n$). С помощью компиляции тестовых приложений экспериментально было установлено, что анализ до двух уровней вложенности структур является оптимальным ($outer_s \rightarrow inner_s \rightarrow n$). Одного уровня вложенности было недостаточно, а анализ до трёх уровней не увеличивал множество обрабатываемых случаев.

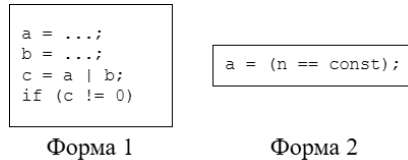


Рис. 7. Шаблонные формы, необходимые для первого шага оптимизации.

Fig. 7. Template forms required for the first optimization step.

Далее алгоритм выполняет трансформацию: в коде базового блока “if” остается лишь первое из простых условий, затем создается базовый блок “elseif”, в который помещается код второго простого условия. Стоит отметить, что алгоритмический подход первого шага способен обрабатывать цепочки “if-elseif” любой длины, так как в GCC не встречаются базовые блоки со сложными условиями, состоящими более чем из двух простых.

4.3 Второй шаг оптимизации

Второй шаг призван осуществить версионирование для подходящих шаблонов кода. Благодаря первому шагу все конструкции “if-elseif”, удовлетворяющие необходимым условиям, теперь представлены в виде двух отдельных базовых блоков.

Алгоритм второго шага в порядке RPO (Reverse Postorder) находит в графе потока управления базовые блоки с условиями проверки на равенство какой-либо переменной константе и дополнительно проверяет, что соответствующий базовый блок “then” содержит необходимый вызов так же, как это проверялось на первом шаге. Порядок RPO гарантирует,

что в любой паре базовых блоков “if-elseif” блок “if” посетится первым, так как является предком блока “elseif”.

Трансформации, выполняемые на втором шаге (рис. 8):

- 1) Происходит расщепление (split) ребра, соединяющего блок “then” с его потомком. Вставляется специальный блок “merge”. Данный блок отвечает за точку схождения путей, проходящих через блоки “then” и “then_1” (описан в следующем пункте), позволяя компилятору осуществить восстановление SSA [10] формы с помощью добавления в блок “merge” необходимых ϕ -функций.
- 2) Базовый блок “then”, содержащий необходимый вызов, копируется, создавая блок “then_1”. При создании блока “then_1” ребро, соединявшее блок “if” с блоком “then”, перенаправляется так, что теперь его потомком становится “then_1”.

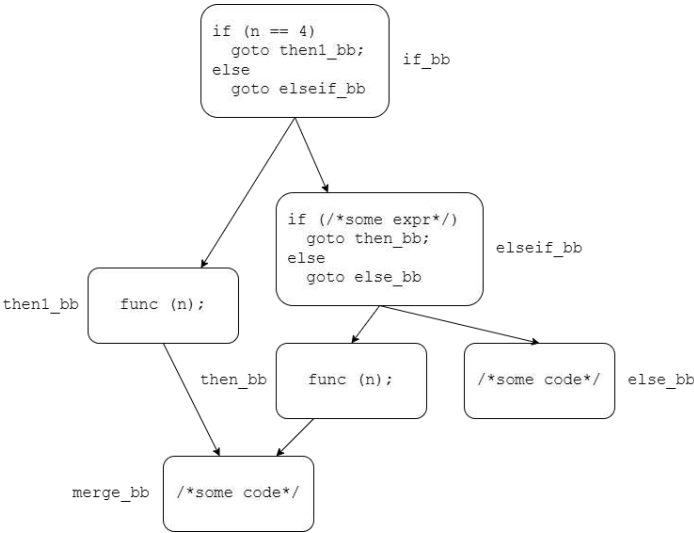


Рис. 8. Пример работы алгоритма второго шага.
Fig. 8. Example of second step algorithm work.

Поскольку первый шаг способен формировать цепочки “if-elseif” любой длины, тогда, соответственно, второй шаг способен осуществлять версионирование для любого числа подходящих проверок для каждого из вызовов, то есть могут быть обработаны примеры кода подобные изображенному на рис. 9.

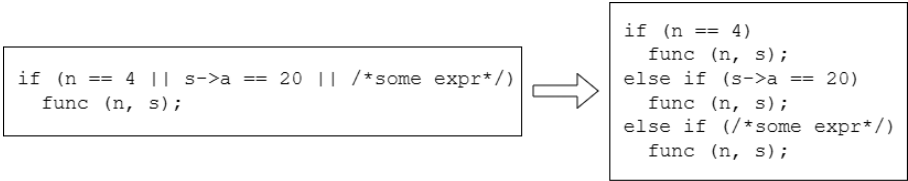


Рис. 9. Пример версионирования для сложного случая.
Fig. 9. Example of versioning for complex case.

5. Результаты работы оптимизации

В качестве результатов и демонстрации ценности проделанной работы приведена статистика работы реальных тестовых приложений из пакета CPUBench [11]. Замеры проводились на серверном процессоре с архитектурой ARM64, исходным набором опций был взят ‘-O3 -fno’. На диаграмме (рис. 10) показано отношение результатов запуска приложений с исходным

набором опций, дополненным нашим проходом, к результатам запуска приложений с исходным набором. Методология CPUbench устанавливает, что в качестве результата каждой отдельного запуска выступает отношение времени выполнения данного запуска на эталонной машине к времени выполнения этого же запуска на тестируемой системе. Запуск каждого теста проводился 100 раз, при этом среднеквадратичное отклонения времени запуска не превышало 0,5%.

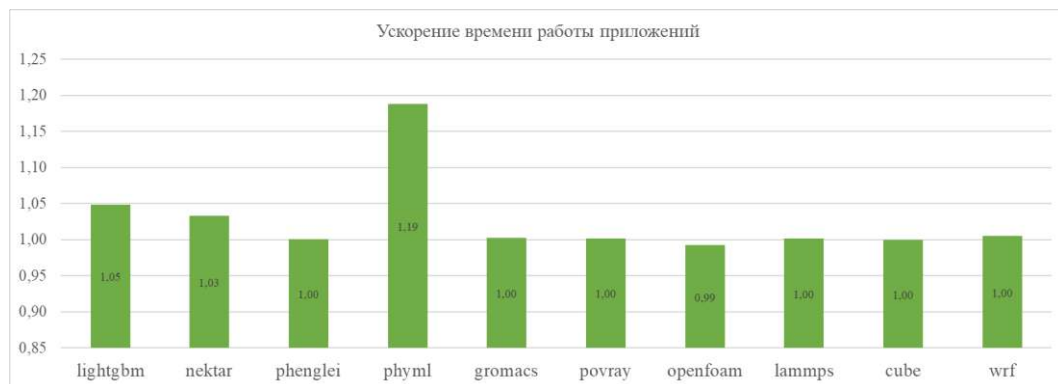


Рис. 10. Результаты по улучшению производительности тестовых приложений.
Fig. 10. Results of improving the performance of test applications.

Максимальное ускорение в 19% было достигнуто на бенчмарке *phym1*. В этом бенчмарке встретилась конструкция, где “горячая” вычислительная функция содержала внутри себя два вложенных цикла с числом итераций, поступающим в функцию в качестве аргумента. Выше по графу вызовов располагалась функция, содержащая в себе проверку соответствующей переменной на равенство 4 или 20. После изучения профиля программы оказалось, что значение этой переменной чаще всего было равно 4. Версионирование и последующее *распространение* значения 4 позволило *циклической векторизации* превратить вычисления в векторные, а *раскрутке циклов* трансформировать этот цикл в полностью линейный участок кода. Также имеют место ускорения на бенчмарке *lightgbm* примерно на 5% и на *nektar* примерно на 3%. Отметим, что на бенчмарке *openfoam* была обнаружена деградация производительности порядка 1%, которую не удалось сопоставить с каким-либо участком программы, исходя из анализа профиля. Предположительно, деградация возникла из-за изменения разложения кода в памяти и целевых адресов инструкций перехода.

Кроме того, было установлено, что при использовании представленного оптимизационного прохода, осуществляющего версионирование, общее число применений трансформации Распространения Констант в целом на пакете тестов увеличилось на 115 единиц.

6. Заключение

В рамках данной работы была рассмотрена техника версионирования, способная развить оптимизацию *распространением констант*. Были рассмотрены существующие решения и был предложен алгоритм, анализирующий множество возможных примеров кода, не обрабатывавшихся ранее. Алгоритм был реализован и внедрен в качестве оптимизационного прохода в компиляторе GCC.

Было показано, что оптимизация способна давать значительное ускорение работы тестовых приложений, а если вносить деградацию, то незначительную. Также было показано, что предложенный алгоритм действительно предоставил больше возможностей существующей оптимизации *распространением констант*.

Список литературы / References

- [1]. GCC official website. Available at: <https://gcc.gnu.org/>, accessed 01.11.2025.
- [2]. Wegman, Mark N., Zadeck F. K. Constant propagation with conditional branches. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1991, vol. 13, no. 2, pp. 181-210.
- [3]. Callahan D., Cooper K. D., Kennedy K., Torczon L. Interprocedural constant propagation. *ACM SIGPLAN Notices*, 1986, vol. 21, no. 7, pp. 152-161.
- [4]. Harrison W. H. Compiler analysis of the value ranges for variables. *IEEE Transactions on software engineering*, 1977, no. 3, pp. 243-250.
- [5]. Huang J. C., Leng T. Generalized loop-unrolling: a method for program speedup. *Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. ASSET'99 (Cat. No. PR00122)*, IEEE, 1999, pp. 244-248.
- [6]. Liang X., Humos A. A., Pei T. Vectorization and parallelization of loops in C/C++ code, *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS). The Steering Committee of The World Congress in Computer Science, Computer*, 2017, pp. 203-206.
- [7]. Glek T., Hubicka J. Optimizing real world applications with GCC link time optimization. *arXiv preprint*. Available at: <https://arxiv.org/abs/1010.2196>, 2010, accessed 03.02.2026.
- [8]. Khedker U. GCC Translation Sequence and Gimple IR. *GCC Resource Center, Department of Computer Science and Engineering, Indian Institute of Technology, Bombay*, 2010.
- [9]. Исходный код в openEuler/gcc. Available at: <https://gitee.com/openeuler/gcc/pulls/256/commits>, accessed 03.02.2026.
- [10]. Cytron R., Ferrante J., Rosen B. K. et al. Efficiently computing static single assignment form and the control dependence graph. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 1991, vol. 13, no. 4, pp. 451-490.
- [11]. Lu H., Ren X., Zhong W. et al. CPUBench: An open general computing CPU performance benchmark tool. *Microelectronics & Computer*, 2023, vol. 40, no. 5, pp. 75-83.

Информация об авторах / Information about authors

Иван Александрович ЗИНИН – студент МФТИ. Сфера научных интересов: компиляторные технологии, оптимизации, гетерогенные вычислительные системы.

Ivan Aleksandrovich ZININ – MIPT student. Research interests: compiler technologies, optimizations, heterogeneous computing systems.

Вячеслав Викторович ЧЕРНОНОГ – кандидат технических наук, ассистент кафедры перспективных вычислительных технологий МФТИ. Сфера научных интересов: компиляторные технологии, оптимизирующие компиляторы, анализ производительности высоконагруженных систем.

Viacheslav Viktorovich CHERNONOG – Cand. Sci. (Tech.), Assistant at the Department of Advanced Computing Technologies at MIPT. Research interests: compiler technologies, optimizing compilers, performance analysis of high-load systems.

DOI: 10.15514/ISPRAS-2026-38(1)-6



Type-based Escape Analysis with Existential Lifetimes

¹A.S. Stoyan, ORCID: 0009-0004-6935-1447 <a.stoyan@hse.ru>

¹HSE University,

16, Soyuza Pechatnikov st., St Petersburg, 190121, Russia.

Abstract. Escape analysis is a fundamental technique in programming languages used to ensure the safety of various compiler optimizations and language features. To achieve high precision and efficiency, particularly in interprocedural scenarios, escape analysis can be integrated into the type system. However, a significant drawback of this approach is that it often necessitates complex and verbose type annotations from the programmers.

In this paper, we introduce a novel type-based escape analysis that simplifies lifetime tracking for data types through the use of existential lifetimes. Additionally, our escape-analysis incorporates a minimum operation on lifetimes, which enables user-provided type annotations to more accurately capture a function's data flow, thereby making them simpler and more intuitive to write. Finally, we present a prototype implementation of the corresponding type-checking algorithm.

Keywords: type systems; escape analysis; subtyping.

For citation: Stoyan A.S. Type-based Escape Analysis with Existential Lifetimes. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 1, 2026, pp. 71-78. DOI: 10.15514/ISPRAS-2026-38(1)-6.

Escape-анализ с экзистенциальными метками времени жизни

¹А.С. Стоян, ORCID: 0009-0004-6935-1447 <a.stoyan@hse.ru>

¹Национальный исследовательский университет «Высшая школа экономики»,
Россия, 190121, г. Санкт-Петербург, ул. Союза Печатников, д. 16.

Аннотация. Escape-анализ широко используется для обеспечения безопасности различных компиляторных оптимизаций и языковых возможностей. Для достижения лучшей точности и эффективности, особенно в интерпроцедурных сценариях, escape-анализ может быть интегрирован с системой типов. Однако, этот подход зачастую требует сложных и объёмных типовых аннотаций со стороны программистов.

В этой статье мы предлагаем основанный на системе типов метод escape-анализа, упрощающий контроль времен жизни типов данных путём использования экзистенциальных меток времени жизни. К тому же, наш метод включает в себя операцию минимума на метках времени жизни, что позволяет пользовательским типовым аннотациям точно отражать поток данных функций, упрощая написание таких аннотаций. И наконец, мы представляем прототип реализации соответствующего алгоритма проверки типов.

Ключевые слова: системы типов; escape-анализ; подтипизация.

Для цитирования: Стоян А.С. Escape-анализ с экзистенциальными метками времени жизни. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 71–78 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(1)-6.

1. Introduction

Escape analysis is a static analysis technique used to determine whether the lifetime of an object exceeds its defining lexical scope. This technique is widely used by optimizing compilers, *for example*, to facilitate stack allocation of objects that would otherwise be allocated on the heap, thereby reducing garbage collection overhead and enhancing overall program performance [1]. Beyond optimization, escape analysis is critical for verifying the safety of various language features. For example, ensuring that a first-class function does not escape its intended scope allows for the safe execution of non-local returns [2]. Similarly, ownership-based resource management systems frequently rely on borrowing mechanisms, which permit temporary resource access without ownership transfer within confined scopes [3-4]. The primary motivation for this work is the application of escape analysis to enforce safety in contextual polymorphism within capability-based effect systems [5-6].

Integrating escape analysis with the type system offers several significant advantages over more conventional approaches. First, escape information is made explicit in function signatures, which allows interprocedural analysis to be performed as an intraprocedural process that consults the signatures of callees. Second, such an analysis does not generate long-distance constraints, which contributes to more precise and localized error reporting.

Despite these advantages, relatively few programming languages have adopted escape analysis integrated directly into the type system, primarily due to the inherent rigidity or complexity of the resulting systems. For instance, the lifetime system in the Rust programming language requires support for variance inference, lifetime polymorphism for compound data types, and explicit lifetime constraints [3]. Collectively, these requirements result in a model that, while flexible, presents significant challenges for both compiler implementers and end users.

In this work, we present a type-based escape analysis that intentionally trades off some expressivity to simplify user-provided type annotations. Specifically, we introduce existential lifetimes to automate the handling of lifetimes within data types. Additionally, our approach incorporates minimum operation on lifetimes, enabling type signatures to transparently reflect a function's data flow.

The remainder of this paper is organized as follows. Section 2 provides an informal introduction to our escape analysis approach, illustrated through practical examples. In Section 3, we present a formalization of our system using a rigorous formal calculus. Section 4 discusses details of our prototype implementation. Finally, Section 5 compares our design with existing approaches.

2. Solution overview

This section provides an informal overview of the type-based escape analysis technique introduced in this paper.

We define values that are constrained to a particular scope as *tracked*, and their corresponding types are designated as *tracked types*. To distinguish between tracked and non-tracked types, each type is annotated with a label, referred to as a *lifetime*. Syntactically, a type T with lifetime lab is denoted as $T'lab$. In the case of function types, the annotation specifies the lifetime of the closure, as in $(A)'lab \rightarrow B$.

The *free* lifetime label denotes non-tracked types. In contrast, the *local* label appears in tracked types, which should be prevented from escaping their defining lexical scope. Specifically, values typed with *local* cannot be returned from functions. For example, this allows the type signature of a higher-order *map* function to explicitly guarantee that an argument function remains confined to the function's scope:¹²

¹ We adopt the Haskell naming convention, where type names beginning with a lowercase letter denote type variables, which are implicitly universally quantified.

² When we omit the explicit lifetime annotation, we assume the *free* lifetime.

```
fun map(xs: List<a>, f: (a)'local -> b): List<b>
```

To permit a function to propagate arguments through its return value, we employ polymorphism over lifetime labels. This approach effectively encodes the function's data flow within the type signature. For instance, the `lazyMap` function does not eagerly compute a final result; rather, it yields a collection that encapsulates the computation logic. This behavior is modeled by assigning the same lifetime variable `l` to both the argument closure `f` and the result type. Consequently, the type system establishes that any tracked values captured by `f` may persist within the returned value:

```
fun lazyMap(xs: LazyList<a>, f: (a)'l -> b): LazyList<b>'l
```

```
fun printAll(file: File'local, xs: LazyList<Int>) {
  let ys: LazyList<Int>'local = lazyMap(xs, (x) => file.write(x); x)
  ys.collect() // force execution
}
```

To express the capture of two or more lifetime-polymorphic values, we employ a syntax with `plus` that denotes the minimum of their lifetimes:

```
fun compose(f: (b)'lf -> c, g: (a)'lg -> b): (a)'lf+lg -> c =
  (x) => f(g(x))
```

For the sake of simplicity, lifetime polymorphism is not supported for compound data types. Instead, the lifetime of a compound data type is derived as the minimum of the lifetimes of its constituent components; this same principle applies to closures. Conversely, during destructuring, the lifetimes of extracted components are approximated by the lifetime of the containing compound data type:

```
fun makeRepository(file: File'lf, conn: Connection'lc): Repository'lf+lc =
  Repository(file, conn)
```

```
fun registerUser(repo: Repository'local, user: User) {
  let conn: Connection'local = repo.conn
  conn.send(user)
}
```

We define a subtyping relationship such that non-tracked values may be used where tracked values are expected: `free <: local`. For simplicity, our system treats all type constructors as covariant with respect to their lifetimes.

Following [6], we permit tracked types to instantiate polymorphic type parameters, thereby making them first-class citizens in the language. Bounded polymorphism is used to control when tracked types are appropriate for such instantiation. For example, the `map` function operates on values of type `a` and returns values of type `b` to the caller's scope. Consequently, these type parameters can be safely instantiated with tracked types:³

```
fun map(xs: List<a>, f: (a)'local -> b): List<b>
  where a <: Any'local, b <: Any'local
```

When a variable is captured, an unknown lifetime associated with a type argument is approximated by the lifetime of its bound. Consider the most permissive type signature for the `lazyMap` function, which demonstrates this principle:

```
fun lazyMap(xs: LazyList<a>, f: (a)'lf -> b): LazyList<b>'lf+la
  where a <: Any'la, b <: Any'lb
```

3. Formalization

We formalize our approach as a call-by-value calculus, denoted as Core_Δ , which is essentially a polymorphic λ -calculus extended with data constructors, subtyping and lifetime labels.

³ We assume that `Any` type serves as the top element in the type lattice.

3.1 Syntax and semantics of $Core_{\Delta}$

The term-level syntax of $Core_{\Delta}$ is presented in Fig. 1. In this notation, the overline denotes a sequence of corresponding entities. Furthermore, type application and type abstraction have two groups of arguments: lifetimes and types.

Variables	x, y, z, f, g
Data constructors	K
Values	$v ::= x \mid K \mid \Lambda \bar{l} (\alpha <: \tau). v \mid K \bar{\Delta} \bar{\tau} \bar{v} \mid \lambda(x : \tau). t$
Terms	$t, u ::= v \mid t \bar{\Delta} \bar{\tau} \mid t \bar{t} \mid match\ t \{ \bar{K} \bar{x} \rightarrow \bar{t} \}$
Evaluation context	$E ::= \square \mid E \bar{\Delta} \bar{\tau} \mid E \bar{t} \mid v (\bar{v}, E, \bar{t})$ $\mid K \bar{\Delta} \bar{\tau} (\bar{v}, E, \bar{t}) \mid match\ E \{ \bar{K} \bar{x} \rightarrow \bar{t} \}$

Fig. 1. Syntax of $Core_{\Delta}$.

The type-level syntax of $Core_{\Delta}$ is given in Fig. 2. We use the $+$ operator on lifetimes in prefix position when taking the minimum of a sequence of lifetimes (e.g. $+\bar{\Delta}$)⁴ and in notation for specific lifetimes (e.g. $\Delta_1 + \Delta_2$). All monotypes are annotated with lifetimes, while type schemas quantify over lifetime parameters and bounded type parameters.

Type variables	α, β
Type constructors	T
Lifetime variables	l
Lifetimes	$\Delta ::= l \mid local \mid free \mid +\bar{\Delta}$
Monotypes	$\tau, \sigma, \eta ::= \alpha \mid T \bar{\Delta} \bar{\tau} \mid \bar{\tau} \bar{\Delta} \rightarrow \sigma$
Type schemas	$s ::= \forall \bar{l} (\alpha <: \tau). \sigma$
Typing contexts	$\Gamma ::= \emptyset \mid x : s, \Gamma \mid l <: \Delta, \Gamma \mid \alpha <: \tau, \Gamma$ $\mid K : \forall \bar{l} \bar{\alpha}. \bar{\tau} \rightarrow T \bar{\Delta} \bar{\alpha}, \Gamma$

Fig. 2. Type syntax of $Core_{\Delta}$.

The reduction semantics of $Core_{\Delta}$ is presented in Fig. 3.

(tapp)	$(\Lambda \bar{l} (\alpha <: \tau). t) \bar{\Delta} \bar{\sigma} \longrightarrow [\bar{l} \rightarrow \bar{\Delta}, \bar{\alpha} \rightarrow \bar{\sigma}] t$
(app)	$(\lambda(x : \tau). t) \bar{v} \longrightarrow [\bar{x} \rightarrow \bar{v}] t$
(match)	$match\ K \bar{\Delta} \bar{\tau} \bar{v} \{ \bar{K}_i \bar{x}_i \rightarrow \bar{u}_i \} \longrightarrow [\bar{x}_k \rightarrow \bar{v}] u_k, \text{ where } K = K_k$
	$\frac{t \longrightarrow t'}{E[t] \longrightarrow E[t']} \text{ (step)}$

Fig. 3. Operational semantics of $Core_{\Delta}$.

3.2 Subtyping relation

The subtyping relation on lifetimes, illustrated in Fig. 4, constitutes a bounded lattice with *local* as the top element and *free* as the bottom. Consequently, the minimum of a set of lifetimes corresponds to the least upper bound within this lattice, making it analogous to the union type operation⁵. When $\Delta' < : \Delta$, we say that Δ' outlives Δ .

⁴ Effectively, the empty list of lifetimes in the minimum denotes the *free* lifetime. We discuss normalized presentation later in Section 4, but in the paper, we use the direct definition for clarity.

⁵ Note that so-called “minimum” of lifetimes correspond to “maximum” in terms of lattice theory.

$$\boxed{\Gamma \vdash \Delta' <: \Delta}$$

$$\frac{}{\Gamma \vdash \text{free} <: \Delta} \text{SubFree} \qquad \frac{}{\Gamma \vdash \Delta <: \text{local}} \text{SubLocal}$$

$$\frac{l <: \Delta \in \Gamma}{\Gamma \vdash l <: \Delta} \text{SubCtx}_{\Delta} \qquad \frac{\forall \Delta' \in \overline{\Delta'}. \exists \Delta \in \overline{\Delta}. \Gamma \vdash \Delta' <: \Delta}{\Gamma \vdash +\overline{\Delta'} <: +\overline{\Delta}} \text{Sub+}$$

$$\boxed{\Gamma \vdash \sigma <: \tau}$$

$$\frac{\alpha <: \tau \in \Gamma}{\Gamma \vdash \alpha <: \tau} \text{SubCtx} \qquad \frac{\Gamma \vdash \Delta' <: \Delta}{\Gamma \vdash T \Delta' \overline{\tau} <: T \Delta \overline{\tau}} \text{SubData}$$

$$\frac{\Gamma \vdash +lt_{\Gamma}(\tau) <: \Delta}{\Gamma \vdash \tau <: \text{Any } \Delta} \text{SubAny} \qquad \frac{\overline{\Gamma \vdash \tau <: \tau'} \quad \Gamma \vdash \Delta' <: \Delta \quad \sigma' <: \sigma}{\Gamma \vdash \overline{\tau'} \Delta' \rightarrow \sigma' <: \overline{\tau} \Delta \rightarrow \sigma} \text{SubFun}$$

Fig. 4. Subtyping rules of Core_{Δ} .

We define the operation $lt_{\Gamma}(\cdot)$ to aggregate all lifetime restrictions inherent in a type. This operation is formally defined as follows:

$$\begin{aligned}
 lt_{\Gamma}(\alpha) &= lt_{\Gamma}(\tau) \text{ if } \alpha <: \tau \in \Gamma, \emptyset \text{ otherwise} \\
 lt_{\Gamma}(T \Delta \overline{\tau}) &= \overline{lt_{\Gamma}(\tau)} \cup \{\Delta\} \\
 lt_{\Gamma}(\overline{\tau} \Delta \rightarrow \overline{\sigma}) &= \{\Delta\} \\
 lt_{\Gamma}(\forall \overline{l} (\alpha <: \tau). \sigma) &= lt_{(\overline{\alpha <: \text{Any free}}, \Gamma)}(\sigma) \text{ remove } \overline{l}
 \end{aligned}$$

The lifetime of a type variable is approximated by the lifetime of its bounding type, if such a bound exists. In some cases, we intentionally employ an empty context to exclude the influence of bounds. The last rule excludes the influence of quantified parameters from the result by changing the bounds for $\overline{\alpha}$ and removing all occurrences of $\overline{l}$.

The *SubAny* rule establishes that the type *Any* Δ is a supertype of τ if and only if all constituent components of τ outlive Δ .

3.3 Typing basics of Core_{Δ}

The *Lam* rule (Fig. 5) is crucial for our escape analysis as it prevents leaking of tracked values by prohibiting types with *local* lifetime present in the return type. Furthermore, this rule establishes that the lifetime of a function is determined by the lifetimes of all values it captures.

3.4 Typing data of Core_{Δ}

Analogous to functions, the lifetime of a data constructor is approximated by the minimum of the lifetimes of all its constituent data types (Fig. 6). Consequently, the lifetime parameters of a data constructor do not appear explicitly in the resulting type, which makes them existential in nature.

Working with existential lifetimes requires special care in the *Match* rule. For each data constructor K_i and its existential lifetimes $\overline{l}_i$ we generate fresh lifetimes $\overline{l}'_i$ and approximate their bound using the lifetime of the enclosing data structure.

In each pattern-matching branch, it is essential to eliminate these fresh lifetimes from the resulting type, as they are not bound in the outer scope. The function $elim_{\Delta, \overline{l}}^p(\cdot)$ eliminates lifetimes $\overline{l}$ (Fig. 7), considering the lower bound Δ and the variance position p , where p ranges over $+$, $-$ and *inv* (with p^{-1} defined as $+$ if $p = -$, $-$ if $p = +$, and *inv* if $p = \text{inv}$).

$$\boxed{\Gamma \vdash t : s}$$

$$\frac{x : s \in \Gamma}{\Gamma \vdash x : s} \text{Var}$$

$$\frac{\overline{\alpha} <: \sigma, \Gamma \vdash t : \tau}{\Gamma \vdash \Lambda \bar{l} (\overline{\alpha} <: \sigma). t : \forall \bar{l} (\overline{\alpha} <: \sigma). \tau} \text{TLam}$$

$$\frac{x : \forall \bar{l} (\overline{\alpha} <: \tau). \sigma \in \Gamma \quad \Gamma \vdash \tau' <: [\bar{l} \rightarrow \Delta] \tau}{\Gamma \vdash x \Delta \tau' : [\overline{\alpha} \rightarrow \tau'] [\bar{l} \rightarrow \Delta] \sigma} \text{TApp}$$

$$\frac{\overline{x} : \tau, \Gamma \vdash t : \sigma \quad \Gamma \vdash_{fv(t) \setminus \overline{x}} \bar{s} \quad local \notin lt_{\emptyset}(\sigma)}{\Gamma \vdash \lambda(\overline{x} : \tau). t : \bar{\tau} \left(+lt_{\Gamma}(\bar{s}) \right) \rightarrow \sigma} \text{Lam}$$

$$\frac{\Gamma \vdash t : \bar{\tau} \Delta \rightarrow \sigma \quad \Gamma \vdash u : \tau' \quad \Gamma \vdash \tau' <: \tau}{\Gamma \vdash t \bar{u} : \sigma} \text{App}$$

$$\boxed{\Gamma \vdash_{\overline{x}} \bar{\tau}} \quad \frac{\overline{x} : \bar{s} \subset \Gamma}{\Gamma \vdash_{\overline{x}} \bar{s}} \text{TypesOf}$$

 Fig. 5. Typing basic constructs of $Core_{\Delta}$.

$$\boxed{\Gamma \vdash t : s}$$

$$\frac{K : \forall \bar{l} \overline{\alpha}. \bar{\tau} \rightarrow T \left(+lt_{\emptyset}(\bar{\tau}) \right) \quad \overline{\alpha} \in \Gamma}{\Gamma \vdash K : \forall \bar{l} \overline{\alpha}. \bar{\tau} \rightarrow T \left(+lt_{\emptyset}(\bar{\tau}) \right) \quad \overline{\alpha}} \text{Ctor}$$

$$\frac{\Gamma \vdash t : T \Delta \bar{\tau} \quad \Gamma \vdash K_i : \forall \bar{l}_i \overline{\alpha}. \bar{\sigma}_i \rightarrow T \left(+lt_{\emptyset}(\bar{\sigma}_i) \right) \quad \overline{\alpha} \quad \overline{l'_i} \text{ fresh} \quad \overline{l'_i} <: \Delta, x_i : [\overline{\alpha} \rightarrow \bar{\tau}] [\bar{l}_i \rightarrow \overline{l'_i}] \sigma_i, \Gamma \vdash u_i : \eta_i}{\Gamma \vdash match \ t \ \{ \overline{K_i} \ \overline{x_i} \rightarrow \overline{u_i} \} : lub \left(elim_{\Delta, \overline{l'_i}}^+ (\eta_i) \right)} \text{Match}$$

 Fig. 6. Typing data constructs of $Core_{\Delta}$.

$$\begin{aligned}
 elim_{\Delta, \bar{l}}^p(\alpha) &= \alpha \\
 elim_{\Delta, \bar{l}}^p(T \ \Delta' \ \bar{\tau}) &= T \left(elim_{\Delta, \bar{l}}^p(\Delta') \right) \left(elim_{\Delta, \bar{l}}^{inv}(\bar{\tau}) \right) \\
 elim_{\Delta, \bar{l}}^p(\bar{\tau} \ \Delta' \rightarrow \sigma) &= \left(elim_{\Delta, \bar{l}}^{p-1}(\bar{\tau}) \right) \left(elim_{\Delta, \bar{l}}^p(\Delta') \right) \rightarrow elim_{\Delta, \bar{l}}^p(\sigma) \\
 elim_{\Delta, \bar{l}}^p(l') &= \begin{cases} l' & \text{if } l' \notin \bar{l} \\ \Delta & \text{if } l' \in \bar{l} \wedge p = + \\ free & \text{if } l' \in \bar{l} \wedge p = - \\ error & \text{if } l' \in \bar{l} \wedge p = inv \end{cases} \\
 elim_{\Delta, \bar{l}}^p(+\overline{\Delta'}) &= + \left(elim_{\Delta, \bar{l}}^p(\Delta') \right) \\
 elim_{\Delta, \bar{l}}^p(\Delta') &= \Delta'
 \end{aligned}$$

 Fig. 7. Elimination function $elim$.

When an existential lifetime appears in an invariant position, it cannot be safely approximated by either its lower or upper bound; consequently, an error is reported.

The resulting type of the entire pattern-matching expression is computed as the least upper bound (*lub*) of the branch types, from which the existential lifetimes have been eliminated.

4. Implementation

We provide a prototype implementation of the type-checker, developed in Haskell [7]. The core logic of the type-checker is concise, spanning approximately two hundred lines of code. The implementation directly follows the formal rules described in Section 3, with the exception that lifetimes are represented in a normalized form:

```
data Lt = LtLocal | LtMin (Set LtName)
```

In this representation, the empty set denotes the *free* lifetime.

5. Related work

The Rust language supports type-based escape analysis to ensure memory safety when using references [3]. This is achieved through a combination of lifetime polymorphism, variance inference, and lifetime constraints, which together provide significant representational power but also increase the overall complexity of the type system. However, the minimum of lifetimes can be achieved only indirectly through additional lifetime constraints or by making all corresponding lifetimes equal. In contrast, our approach provides a way to express minimum directly.

In modal OCaml, the *local* modality serves to prevent borrowed references from escaping their defining region, ensuring safe memory use and enabling optimizations such as stack allocation [4]. Modalities here are not first-class type constructors, but rather properties of bindings, so types with modalities cannot, for example, instantiate type variables, which limits the expressiveness further.

Contemporary research in effect systems makes extensive use of type-based escape analysis to formally guarantee safety properties. Xie et al. [8] adopt higher-rank types to enforce the non-leakage of lexical effect handlers. Capability-based effect systems employ type-based escape analysis to establish safety of contextual polymorphism [9, 5, 10]. Recent work in the Scala language utilizes dependent types to check escaping [6].

6. Conclusion

In this work, we proposed a type-based escape analysis technique that strategically trades a degree of flexibility for the simplicity of user-defined annotations. Specifically, our approach incorporates the minimum operation on lifetimes and streamlines the handling of compound data types by approximating their lifetimes as the minimum of the lifetimes of their constituent components. Furthermore, we provided a formal specification of the system alongside a prototype implementation of the type-checker.

In future work, we intend to develop a mechanized proof of soundness and evaluate the practical implications of the trade-offs inherent in our system through application to larger-scale codebases.

References

- [1]. Blanchet B. Escape analysis for object-oriented languages: application to Java. ACM SIGPLAN Notices, vol. 34, no. 10, 1999, pp. 20–34. DOI: 10.1145/320384.320387.
- [2]. Akhin M., Belyaev M. Kotlin language specification. Kotlin Language Specification, 2021.
- [3]. Matsakis N.D., Klock F.S. The Rust language. Proceedings of the 2014 ACM SIGAda annual conference on High integrity language technology, 2014, pp. 103–104. DOI: 10.1145/2663171.2663188.
- [4]. Lorenzen A., White L., Dolan S., Eisenberg R.A., Lindley S. Oxidizing OCaml with modal memory management. Proceedings of the ACM on Programming Languages, vol. 8, issue ICFP, 2024, pp. 485–5104. DOI: 10.1145/3674642.

- [5]. Brachthäuser J.I., Schuster P., Ostermann K. Effects as capabilities: effect handlers and lightweight effect polymorphism. *Proceedings of the ACM on Programming Languages*, vol. 4, issue OOPSLA, 2020, pp. 1–30. DOI: 10.1145/3428194.
- [6]. Boruch-Gruszecki A., Odersky M., Lee E., Lhoták O., Brachthäuser J. Capturing types. *ACM Transactions on Programming Languages and Systems*, vol. 45, no. 4, 2023, pp. 1–52. DOI: 10.1145/3622839.
- [7]. Stoyan A. Supplementary code for this work. Available at: <https://github.com/penguin-boxx/existential-escape-analysis-paper>, accessed 05.02.2026.
- [8]. Xie N., Cong Y., Ikemori K., Leijen D. First-class names for effect handlers. *Proceedings of the ACM on Programming Languages*, vol. 6, issue OOPSLA2, 2022, pp. 30–59. DOI: 10.1145/3563309.
- [9]. Osvald L., Essertel G., Wu X., Alayón L.I.G., Rompf T. Gentrification gone too far? Affordable 2nd-class values for fun and (co-) effect. *ACM SIGPLAN Notices*, vol. 51, no. 10, 2016, pp. 234–251. DOI: 10.1145/2983990.2984009.
- [10]. Brachthäuser J.I., Schuster P., Lee E., Boruch-Gruszecki A. Effects, capabilities, and boxes: from scope-based reasoning to type-based reasoning and back. *Proceedings of the ACM on Programming Languages*, vol. 6, issue OOPSLA1, 2022, pp. 1–30. DOI: 10.1145/3527320.

Информация об авторах / Information about authors

Андрей Сергеевич СТОЯН – аспирант. Сфера научных интересов: теория типов, дизайн языков программирования.

Andrey Sergeevich STOYAN – postgraduate student. Research interests: type theory, design of programming languages.

DOI: 10.15514/ISPRAS-2026-38(1)-7



Lightweight File System Crash-consistency Checking with Differential Fuzzing

V.M. Kovalevsky, ORCID: 0009-0002-4501-9672 <slava0135@vk.com>

V.V. Kechin, ORCID: 0009-0006-0845-2740 <valeraghst@gmail.com>

A.S. Yanin, ORCID: 0009-0002-2627-8281 <slowmime@gmail.com>

V.M. Itsykson, ORCID: 0000-0003-0276-4517 <itsykson@yandex.ru>

ITMO University,

Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia.

Abstract. File systems are critical operating system components, impacting the performance, reliability, and security of the entire computing system. One of the key properties of modern file systems is their ability to ensure data consistency. If a file system fails to restore persisted data after a crash, this can lead to corruption of the application state, loss of integrity, or even complete data loss. This paper proposes a lightweight approach for detecting data inconsistency errors caused by unexpected crashes. The approach is based on modeling such crashes and integrating this behavior into a file system error detection system DIFFuzzer based on differential fuzzing. Experimental testing of the updated tool on several well-known file systems demonstrated the effectiveness of the approach. Several crash-consistency errors were discovered during the experiments, confirmed by the file system authors.

Keywords: file systems; differential fuzzing; file system testing; crash consistency.

For citation: Kovalevsky V.M., Kechin V.V., Yanin A.S., Itsykson V.M. Lightweight file system crash-consistency checking with differential fuzzing. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 1, 2026, pp. 79-92. DOI: 10.15514/ISPRAS-2026-38(1)-7.

Легковесная проверка согласованности файловой системы после сбоев с помощью дифференциального фаззинга

В.М. Ковалевский, ORCID: 0009-0002-4501-9672 <slava0135@vk.com>

В.В. Кечин, ORCID: 0009-0006-0845-2740 <valeraghst@gmail.com>

А.С. Янин, ORCID: 0009-0002-2627-8281 <slowmime@gmail.com>

В.М. Ицыксон, ORCID: 0000-0003-0276-4517 <itsykson@yandex.ru>

Университет ИТМО,

Россия, 197101, Санкт-Петербург, Кронверкский пр., д. 49, лит. А.

Аннотация. Файловые системы являются важнейшими компонентами операционной системы, влияющими на производительность, надежность и безопасность всей вычислительной системы. Одним из ключевых свойств современных файловых систем является их способность обеспечивать согласованность данных. Если файловой системе не удастся восстановить сохраненные данные после сбоя, это может привести к повреждению состояния приложения, потере целостности или даже полной потере данных. В этой статье предлагается упрощенный подход к обнаружению ошибок несогласованности данных, вызванных неожиданными сбоями. Подход основан на моделировании таких сбоев и интеграции этого поведения в систему обнаружения ошибок файловой системы DIFFuzzer, основанную на дифференциальном фаззинге. Экспериментальное тестирование обновленного инструмента на нескольких известных файловых системах продемонстрировало эффективность подхода. В ходе экспериментов было обнаружено несколько ошибок согласованности данных после сбоев, подтвержденных авторами файловой системы.

Ключевые слова: файловые системы; дифференциальный фаззинг; тестирование файловых систем; согласованность после сбоев.

Для цитирования: Ковалевский В.М., Кечин В.В., Янин А.С., Ицыксон В.М. Легковесная проверка согласованности файловой системы после сбоев с помощью дифференциального фаззинга. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 79–92 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(1)-7.

1. Introduction

In general, file systems are used by applications as a reliable data storage that is expected to persist application state during the events of hardware or software crashes. Therefore, if a file system is unable to recover persisted data after the crash, then it will result in corrupted application state and catastrophic data loss.

However, modern file systems often have to compromise reliability to get optimal performance. First, file systems can store dirty data in volatile memory. Second, file systems can execute file operations out-of-order. For these reasons, file systems can often exhibit unexpected crash behaviors. As an example, consider the “ext4 data loss” incident of 2009 [1], where a common application pattern resulted in unexpected application states when the system crashes during execution because of these optimizations.

Consequently, application developers should take measures to ensure the application state consistency in the event of crashes. For instance, they should use synchronization operations defined in the POSIX standard [2] to guarantee data persistence, such as *fsync* and *sync*.

A file system is crash-consistent if it always recovers to a correct state and can guarantee data persistence after a fail-stop crash. Unfortunately, file systems are complex, and the resulting complexity often leads to bugs in the synchronization logic that can be hard to detect even under heavy testing.

As a result, a lot of work has been put into developing tools for finding bugs in file systems using both static (e.g., model checking) and dynamic (e.g., fuzzing) software analysis methods. Despite all the effort, the problem of checking file system crash-consistency is not entirely solved as these tools are still not widely adopted by the file system developers. On the one hand, most of the prior

work did not focus on finding crash-consistency bugs in the first place. On the other hand, the solutions that can find crash-consistency bugs are too complex and too difficult to use or to maintain. We propose a new lightweight approach for finding crash-consistency bugs based on differential fuzzing. Fuzzing is an automatic bug and vulnerability discovery technique that continuously generates inputs and reports those that crash the program [3]. It is typically limited to uncovering memory or logic errors because only these kinds of errors result in program termination. Differential fuzzing is a more powerful method that is capable of finding semantic bugs, such as specification violations, where the program fails to adhere to agreed specifications (e.g., POSIX). It works by providing the same input to different implementations of the same program and observing differences in their execution. In our approach, rather than using different file system implementations, we compare file system behavior under normal conditions and behavior of the same file system after a simulated crash. Instead of developing a formal crash-consistency model or enumerating all valid crash states, the behavior is checked by just comparing files that were explicitly persisted using synchronization operations.

We implemented the proposed approach by extending our tool DIFFuzzer from our previous work [4]. In a short experiment, the extended toolkit found several crash-consistency bugs in popular Linux file systems.

The rest of the paper is organized as follows. Section 2 gives background on file system reliability and crash-consistency. Section 3 relates to previous work. Section 4 presents the proposed approach. Section 5 explains the details of architecture and implementation of the tool. Section 6 shows results of evaluation. Section 7 concludes.

2. Background

In prior research, the concept of crash-consistency was not precisely defined and was simply referred to as a property of file systems being able to recover to a correct state after a crash [5]. We want to give a more strict definition of crash-consistency utilizing already established terminology used in the context of software quality assurance.

Consider the series of standards ISO/IEC 25000, also known as SQuaRE (System and Software Quality Requirements and Evaluation). Specifically, the standard ISO/IEC 25010:2023 [6] defines the characteristic of reliability as a “capability of a product to perform specified functions under specified conditions for a specified period of time without interruptions and failures”. Additionally, this characteristic is composed of several sub-characteristics: faultlessness, availability, fault tolerance and recoverability. Importantly, recoverability is defined as a “capability of a product in the event of an interruption or a failure to recover the data directly affected and re-establish the desired state of the system” and closely matches the previously given definition of crash-consistency.

Along with this, the standard defines the characteristic of functional suitability as a “capability of a product to provide functions that meet stated and implied needs of intended users when it is used under specified conditions”. Consequently, crash-consistency properties can also be considered to be a part of functional suitability characteristics, as file systems should adhere to the POSIX standard.

In particular, the POSIX.1-2024 standard [2] gives the following rationale for synchronization operations:

- The *fsync()* function is intended to force a physical write of data from the buffer cache, and to assure that after a system crash or other failure that all data up to the time of the *fsync()* call is recorded on the disk. An application that modifies a directory can invoke *fsync()* on the directory to ensure that the directory's entries and file attributes are synchronized.
- The *sync()* function shall cause all information in memory that updates file systems to be scheduled for writing out to all file systems.

The standard also defines additional flags for *open()* operation:

- **O_DSYNC.** Write I/O operations on the file descriptor shall complete as defined by synchronized I/O data integrity completion.
- **O_RSYNC.** Read I/O operations on the file descriptor shall complete at the same level of integrity as specified by the *O_DSYNC* and *O_SYNC* flags.
- **O_SYNC.** Write I/O operations on the file descriptor shall complete as defined by synchronized I/O file integrity completion.

Here, synchronized I/O *file* integrity completion means that write operations will flush data and all associated metadata to the disk. In contrast, synchronized I/O *data* integrity completion means that write operations will flush data to the underlying hardware, but will only flush metadata updates that are required to allow a subsequent read operation to complete successfully [7].

Overall, we believe that the concept of file system crash-consistency cannot be directly interchanged with the concept of file system recoverability, because of these subtle details. Nevertheless, based on this knowledge, we can define file system crash-consistency as a capability of a file system in the event of a fail-stop crash to recover the data and re-establish the desired state of the system.

Accordingly, the POSIX standard only determines a bare minimum recoverability capability, but file systems can choose to provide stronger data persistence guarantees. For example, a file system can implement a sequential crash-consistency model: after a crash, the system appears as if it committed the system calls in order and all data was recorded on disk [8].

In this paper, we only focus on checking file system crash-consistency guarantees required by the POSIX standard.

3. Related work

3.1 Kernel fuzzers

Syzkaller [9] is a coverage-guided kernel fuzzer used extensively for Linux kernel testing. Over the years, Syzkaller has uncovered hundreds of Linux kernel bugs and proved itself to be a useful and effective tool. Although the system is effective at detecting memory and logic bugs, it lacks the capability to identify crash-consistency bugs. It is also worth mentioning kAFL [10], despite it having the same fundamental problem.

3.2 Persistent memory crash-consistency

Persistent memory (PM) is a relatively new storage-class memory technology enabling persistent, byte-addressable storage with memory-like latency through direct access via the memory bus [11-12]. The effort has resulted in a suite of user-space applications, libraries, and tools for PM, as well as multiple file systems that capitalize on its unique properties [13]. However, crash-consistency testing methodologies are not directly transferable between traditional file systems and those designed for persistent memory. Consequently, a significant and distinct body of research, along with specialized tools, has been developed to identify bugs in PM software. Notable examples include Chipmunk [13], Vinter [14], and, more recently, Silhouette [15]. Again, due to low-level implementation details it is not possible to apply the same methods for testing general purpose file systems, which are the focus of this work.

3.3 File system crash-consistency

Several studies have previously focused on the formal specification and verification of file systems. For instance, SifylFS [16] specifies the range of allowed behaviors of a file system using a mathematically rigorous model of file system behavior based on the POSIX standard. The model did not, however, address file system behavior across system crashes, which constitutes a significant

limitation. A further illustration is the work of Bornholt et al. [8], who introduced the first formalization of file system crash-consistency models. Nevertheless, their research was not directed toward identifying bugs within file systems. In summary, the inherent complexity of file systems renders the development of formal specifications to be challenging, and thus the latest research is more focused on dynamic analysis methods.

CrashMonkey [5] tests the file system in a black-box manner by simulating power-loss crashes while the workload is being executed and checking if the file system recovers to a correct state. However, this tool had several significant limitations. First, the methodology lacks both input randomization and the use of sophisticated strategies for selecting test workloads. Second, the chosen methodology for crash simulation, which is based on the implementation of a custom kernel module, presents significant maintenance challenges. This is a direct consequence of the kernel API's inherent instability and lack of long-term stability guarantees. To address the first issue, Dogfood [17] introduced a novel workload generation methodology based on layered models and the refinement of abstract workloads. Nevertheless, the execution of tests continues to be delegated to CrashMonkey, and the second issue remains unresolved.

Hydra [18] introduces the first grey-box testing framework for assessing file system crash-consistency. It takes an initial file system image and a syscall trace, emulates the operations to produce a symbolic representation of all valid post-crash states based on the file system's crash-consistency guarantees, and then confirms whether a recovered image is among these valid states. However, the task of designing effective testing oracles to uncover crash-consistency bugs after execution is highly demanding and is potentially error-prone. Another issue is that, despite being the most feature-complete file system fuzzer currently available, Hydra's practical application is hampered by maintainability challenges and lack of documentation. Its reliance on obsolete infrastructure, such as Python 2 and Ubuntu 18.04, necessitates substantial modernization efforts.

SnapCC [19] is a more recent effort focused on testing file system crash-consistency. Its methodology employs a suite of specialized facilities: a specification-based generator and mutator for invocation sequences; an Automatic Valid State Finder to establish test oracles; a Systematic On-Disk State Explorer, which enumerates all possible on-disk states resulting from a given sequence; and a Consistency Verifier that identifies file system inconsistency defects. The results shown in the paper demonstrated that SnapCC is effective at identifying crash-consistency bugs in real-world environments, performing at a state-of-the-art level. At the same time, we believe that a more lightweight alternative exists, enabling a solution to the same problem without the implementation of sophisticated test oracles or verifiers and without exhaustive state enumeration.

4. Approach

4.1 Workload generation

File system testing involves executing a series of file operations, such as creating or deleting files. To ensure the test is meaningful, the order of these operations must be semantically correct. An invalid sequence, e.g., attempting to open a file that has already been deleted, will generate trivial errors and is generally less useful for testing. The creation of valid and diverse workloads that incorporate different interactions and dependencies between operations is particularly challenging. Black-box fuzzing generates inputs without any knowledge of the program [3]. There are two primary approaches to black-box fuzzing:

- **Mutational fuzzing.** Begins with initial seeds and generates new test cases by introducing random mutations to them.
- **Generational fuzzing.** Constructs inputs from scratch without relying on seeds. It can be guided by a specification of the input format to ensure they are syntactically valid.

Grey-box fuzzing effectively guides test case generation by using program instrumentation for feedback [3]. Inputs that discover new code paths are retained as new seeds to improve coverage. New test cases are generated by introducing random mutations to seed corpus.

We base our black-box and grey-box methodologies on an imperative file system model, established in our prior work, which serves as a relatively simple approximation of real file system behavior. The abstract model generates file operation sequences of arbitrary length, enabling subsequent replay or the creation of test cases for execution on actual file systems. The model also permits the application of random mutations to the workloads through several techniques: appending new operations, removing existing ones, or merging two workloads into one.

4.2 Crash simulation

Although triggering a real system crash is the most straightforward approach to assess crash-consistency, the mandatory reboot severely compromises testing efficiency. Therefore, this approach is not used in either production or academic environments.

For instance, Linux kernel developers utilize kernels built with the *dm-flakey* feature, which can exhibit unreliable behavior periodically, in order to test file system crash-consistency. However, its application is poorly documented and requires further investigation.

Previous studies employed an alternative methodology using a specialized proxy block device to simulate system crashes. For instance, Bornholt et al. [8] utilized Network Block Device (NBD) and CrashMonkey used a custom Block Ram Device (BRD) module. To simulate a crash, all recorded IO device requests from the start of the workload are replayed until the crash point.

Another approach to emulating a system crash involves the use of virtual machine (VM) snapshots. For example, SnapCC takes snapshots before and after each block device write operation. It then uses a clean snapshot of the operating system and attaches each on-disk state.

In theory, using a proxy block device should result in better performance compared to all other methods, as it eliminates the necessity for extensive data copying. However, this hypothesis remains unexplored in existing research and in this paper.

In summary, the snapshot approach was selected for its straightforward implementation. In contrast to methods used in SnapCC, only the final file system state is captured and analyzed, which we will demonstrate is sufficient to find unexpected system behaviors. More specifically, a clean snapshot of the operating system is loaded once the entire workload and the test process have completed, while the file system hosted on a separate disk remains untouched.

4.3 Unexpected behavior detection

As previously established, the inherent complexity of file systems poses a significant challenge to implement formal specifications, despite their demonstrated utility in detecting unexpected behavior.

Instead, many tools implement various checkers for finding crash-consistency bugs post-execution. At the same time, the development of effective testing oracles is both demanding and susceptible to error. Moreover, the crash-consistency guarantees vary by file system, and the checker must account for these differences.

To reduce effort and mitigate the risk of error, we suggest employing a minimal viable approach for detecting crash-consistency bugs. In short, during the first test execution, the test runtime maintains a record of all inodes that have been explicitly synchronized using the file system API. Subsequently, it updates the expected synchronized file system state after every operation based on changes in full file system state. The second test execution involves a comparison between the post-crash state and the expected synchronized state, limited exclusively to the files designated as synchronized.

5. Implementation

As previously established, this research builds upon our prior work and implementation. The architecture currently includes four components, developed entirely in Rust:

- **AbstractFS.** The model for workload generation.
- **Dash.** Differential abstract state hasher, used to compare file system states.
- **FSTest.** Runtime for executing tests.
- **DIFFuzzer.** Fuzzer itself.

5.1 Overview

The fuzzing session begins by taking a system snapshot once the operating system has finished loading. This snapshot is subsequently used to restore the system to a known clean state. Following this, the file system is subjected to iterative testing using the generated workloads, as described in detail below.

Fig. 1 illustrates the overall workflow. The workload is generated (1) using either black-box or grey-box approach. The program is sent to VM (2). The test is executed for the first time, and the expected synchronization state is established (3). The clean snapshot of the entire VM is loaded and the test is executed a second time (4), incorporating a simulated crash at the end of the process (5). The clean snapshot of the VM is loaded, excluding the disk with the file system, after which the damaged file system is remounted (6). The actual file system state is retrieved and is compared against the expected state (7). Reduction and replication are not integral to the fuzzing operation itself and can be executed as a post-processing step.

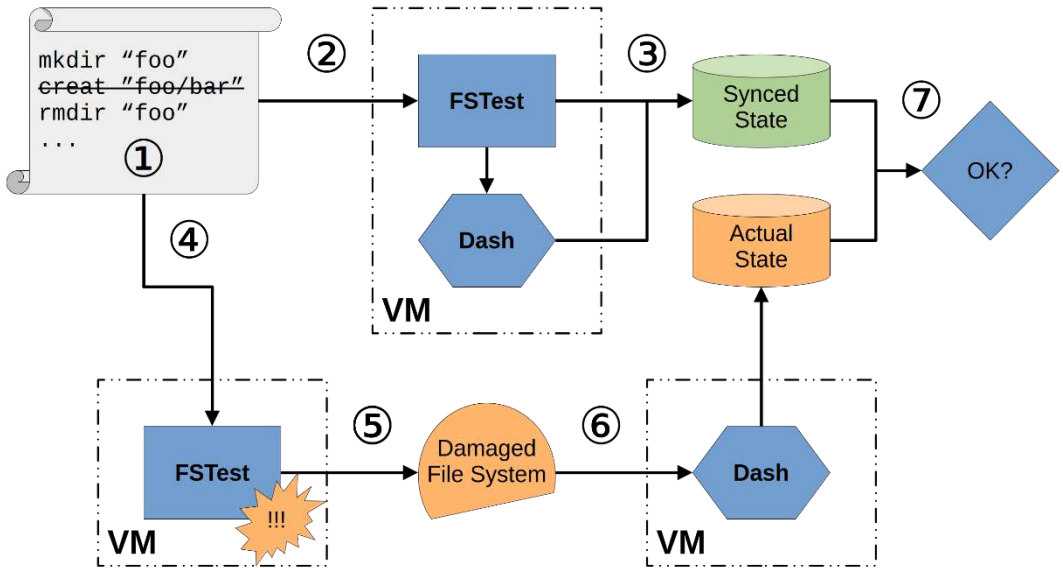


Fig. 1. Overview of fuzzing workflow.

5.2 Workload generation

The abstract model state is defined by:

- Map for each file type (regular file, directory, symlink) associating paths with file metadata objects.
- Maps for each file type associating open file descriptor with file metadata objects.

- Arrays of file metadata objects for each file type. The metadata objects include extended attributes (for all file types) and target path (only for symlinks).

The model performs file operations by utilizing paths directly, eliminating the need for a tree-based abstraction. To illustrate it, consider *rename(old, new)* operation:

- For any path in any map beginning with the prefix *old*, that prefix is replaced with *new*.
- The respective map is updated with the renamed path, which shares the existing metadata object.
- The original path remains in the map unchanged.

The original path is retained in the map to account for potential failures, as the model does not perfectly emulate a real file system. Specifically, the actual file system state after *rename()* operation may include the original path, the renamed path, or neither.

Furthermore, the file naming convention utilizes a finite set of identifiers, which introduces a probability of name collision. This approach is intentional, as it facilitates the testing of error-handling mechanisms without adding complexity to the model. For instance, this allows the model to occasionally attempt to create a file that already exists, testing this specific error case without requiring dedicated programming for it.

Overall, this approximation maintains an acceptable error rate for practical applications, while also providing means to test file system error handling.

The workload is generated by selecting each subsequent operation based on two criteria: satisfaction of state-based preconditions and a configurable weighted probability. As an illustration, a precondition requires that the *close()* operation can only be invoked provided there is a minimum of one open descriptor.

The model supports over 30 file system operations:

- **Creating and removing files.** *mkdir, mkdirat, creat, rmdir, unlink, unlinkat, hardlink, hardlinkat, symlink, symlinkat, rename, renameat*.
- **Reading and writing files.** *open, close, write, pwrite, read, pread, truncate, ftruncate, lseek*.
- **File synchronization.** *fsync, sync*.
- **File permissions and access.** *chmod, fchmod, fchmodat, access, faccessat*.
- **Extended attributes (Linux).** *setxattr, getxattr, removexattr, listxattr*.

5.3 Workload execution

The FSTest runtime executes workloads, which are generated in JSON format by the model.

Initially, the file system is created and mounted on a separate, unallocated disk. This disk is identified by a parameter provided when initializing the QEMU virtual machine. The use of a separate virtual device is a deliberate design choice to enable easy crash simulation at a later stage.

During the first test execution the runtime reconstructs synchronized file system state, which consists of:

- Map “*entries*” associating directory inode with a set of directory entries (inode and name).
- Set “*files*” of regular file inodes which are fully synchronized (metadata and content).

Each operation can update the state as follows:

- *fsync()* called on a regular file adds the associated inode to *files* set.
- *fsync()* called on a directory adds all actual directory entries to *entries* map.
- *write()* or *pwrite()* called on a regular file opened with flag *O_SYNC* has the same effects as *fsync()*.

- *open()* called on a regular file with flag *O_TRUNCATE* removes the associated inode from files set.
- *write()*, *pwrite()*, *ftruncate()*, *truncate()*, *chmod()*, *fchmod()* or *fchmodat()* called on a regular file removes the associated inode from the files set..

Information regarding inodes is retrieved using standard file system APIs:

- *stat()* for operations using file paths.
- *fstat()* for operations using file descriptors.
- *fstatat()* for operations using both file paths and file descriptors.

After operation is completed, full file system state is retrieved and all entries or paths that do not longer exist are removed from respective maps and sets. Fig. 2 demonstrates the reconstruction of synchronized state.

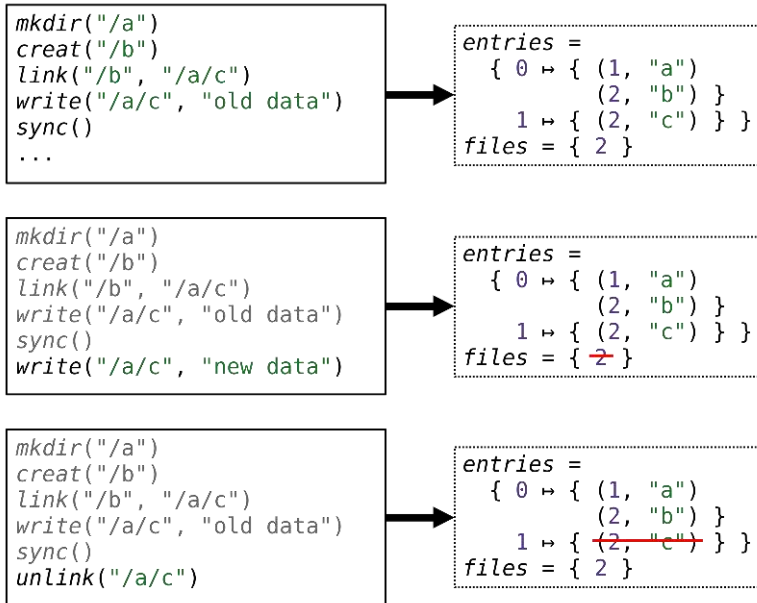


Fig. 2. Reconstruction of synchronized states using simplified operation semantics.

At the end of test execution, the runtime refines the synchronized state, as also demonstrated in Fig. 3:

- Array "*paths*" of synchronized paths derived from the *entries* map. If any component of the path (directory) is not synchronized, the path is ignored.
- Array "*files*" of fully synchronized regular files paths derived from the *entries* map and the *files* set. If any component of the path (directory) is not synchronized, the path is ignored.

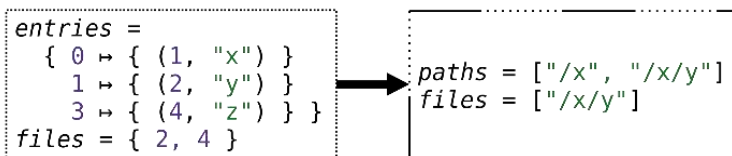


Fig. 3. Refined synchronized state.

Additionally, for the purpose of later comparison, the Dash utility establishes the complete state of the file system. This state, stored separately, includes metadata and a content-derived hash of each file.

The current implementation has two primary limitations. First, partial synchronization updates and mechanisms (e.g., *fdatasync()*) are unsupported. Second, only metadata for regular files is checked. Implementing all this functionality, however, is not expected to be difficult.

The second test execution is conducted without additional procedures. Upon completion, the fuzzer simulates a system crash. Two implementations are available for this process:

- The “*real*” crash, done by executing QEMU *system_reset* command via Human Monitor Protocol (HMP).
- The “*light*” crash, done by loading a snapshot of the OS using the *snapshot-load* command via QEMU Monitor Protocol (QMP). This method preserves the virtual disk state, leaving the file system unaltered.

5.4 Unexpected behavior detection

After the system crashes, the file system is remounted. The inability to remount the file system is considered a bug. Currently, the *fsck* utility is not used for file system repair, given that most file systems can recover without it. Implementation of *fsck* for repair procedures will be available later as the last resort measure. Finally, the recovered state is collected by the Dash utility and subsequently compared against the synchronized state refined earlier:

- All paths from the *paths* array should exist.
- All files from the *files* array should be equal, including metadata and content.

A failure to meet these conditions is reported as a bug.

6. Evaluation

The evaluation was conducted on a host system utilizing an 8-core Intel Core i7-10700 2.90GHz processor with 32 GB of RAM. The host operated an Ubuntu 25.10 server cloud image within a virtual machine, which was allocated 2 virtual cores and 4 GB of RAM. The study utilized two versions of the Linux kernel for testing: the stable release 6.14.11 and the development release candidate 6.18-rc2. The file systems selected for testing were Ext4, Btrfs, XFS, BcacheFS, NILFS2 and the FUSE implementation of LittleFS. Testing was prioritized for the Btrfs file system due to its extensive history of crash-consistency bugs.

Test execution and snapshot manipulations consume the bulk of processor time. Extended testing indicates that a single VM achieves a throughput of roughly 0.50 tests per second with a non-instrumented kernel, which is reduced by half to 0.25 tests per second when a kernel is instrumented. The observed performance marks a considerable decline relative to the tool’s standard differential fuzzing implementation, where throughput reaches about 4.00 tests per second.

The reported performance metrics are based on the use of the snapshot (“*light*”) method for crash emulation. It should be noted that when using a methodology that induces actual system crashes, the resulting throughput is diminished by a factor of ten, making it very impractical. Additionally, these numbers can vary across file systems.

The execution of more than one million tests led to the discovery of four new confirmed bugs within the Btrfs file system [20-23], present in both Linux kernels 6.14.11 and 6.18-rc2. Furthermore, the testing tool successfully reproduced one previously identified bug specific to Btrfs [24], as well as another previously identified bug common to both the Btrfs and Ext4 file systems [25]. All discovered bugs were found to be highly deterministic and easily reproducible.

The first bug (Listing 1) manifests when the following sequence of operations is executed [20]:

1. Create symlink "*slink-old*" in directory "*dir*".
2. *fsync()* directory "*dir*" to persist the symlink.
3. *rename()* the symlink from "*slink-old*" to "*slink-new*".
4. *fsync()* directory "*dir*" to persist the new symlink name.

The defect is located in the fourth step, which fails to execute correctly. Consequently, following a system crash, the symbolic link reverts to its original name, "*slink-old*", rather than retaining the new name, "*slink-new*".

```
int main() {
    int root_fd, dir_fd1, dir_fd2;
    mkdir("dir", S_IRWXO);
    root_fd = open(".", O_RDONLY | O_DIRECTORY);
    fsync(root_fd); // persist `dir`
    symlink("foobar", "dir/slink-old");
    dir_fd1 = open("dir", O_RDONLY | O_DIRECTORY);
    fsync(dir_fd1); // persist `slink-old`
    rename("dir/slink-old", "dir/slink-new");
    dir_fd2 = open("dir", O_RDONLY | O_DIRECTORY);
    fsync(dir_fd2); // persist `slink-new`
}
// after crash, symlink is named `slink-old` instead of `slink-new`
```

Listing 1. Symlink entry is not persisted after rename.

In the original report, the second, more complex bug (Listing 2), results in the directory not being persisted after an implicit *fsync()* call after *write()* to another file, leaving it vulnerable to loss in the event of a system crash [21]. A detailed analysis is omitted for brevity.

```
int main() {
    int file_fd, root_fd;
    creat("file1", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    sync(); // persist `file1`
    mkdir("dir", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    file_fd = open("file1", O_WRONLY | O_SYNC);
    write(file_fd, "Test data!", 10);
    link("file1", "dir/file2");
    root_fd = open(".", O_RDONLY | O_DIRECTORY);
    fsync(root_fd); // persist `dir`
}
// after crash, directory `dir` does not exist.
```

Listing 2. Directory is not persisted after writing to the file within directory.

The third identified bug [22] leads to corruption of the file system after renaming a directory and creating a new file with the same name if system crashes. The report includes a detailed analysis along with the relevant test case.

The fourth and final bug [23] manifests as a failure to persist the directory after performing *write()* and *fsync()* on another file and then renaming that file. The report includes a detailed analysis along with the relevant test case.

Following confirmation of these four issues, the Btrfs development team produced patches and regression tests to address them.

As was established earlier, the tool was able to reproduce two previously identified bugs.

The first bug related to Btrfs, previously documented and resolved, occurred when a power loss and subsequent remount prevented the file system from recovering the log tree, which consequently led

to a system boot failure. This issue, which appeared to impact a significant number of Linux users [26], highlights the critical need for robust crash-consistency testing.

The second previously reported low-level implementation bug [25] related to the `O_SYNC` flag integrity guarantees was found to be independent of the file system, as it manifested on both Ext4 and Btrfs. However, our test results indicated that the XFS file system remained unaffected for undetermined reasons.

Employing either black-box or grey-box implementations is sufficient to uncover these bugs. Although the grey-box approach offers greater potential, its associated instrumentation costs are not currently justified.

In summary, the primary objective of this study has been fulfilled. The presented approach offers an effective solution for detecting crash-consistency bugs, while also affording a relatively uncomplicated implementation process. However, to advance the utility of the tool, future work is necessary to refine its performance and increase its effectiveness.

7. Conclusion

In this paper, a lightweight approach to detecting file system crash-consistency errors was proposed. It involves generating a workload based on a built-in file system model that approximates the behavior of a real file system. The generated workload is used twice: once to capture the reference behavior of the file system, and once to analyze the behavior during a crash, which is modeled using virtual machine crashes and snapshots. Both behaviors are compared to detect crash-consistency errors.

The approach was implemented as an extension to the tool DIFFuzzer and tested on older versions of the Ext4 and Btrfs file systems, known to contain inconsistency errors, and on the current version of the Btrfs file system. Experiments showed that the tool, first, detected previously known errors, and second, detected four errors in the current version of Btrfs. These bugs were reported to the Btrfs bug tracking system; all bugs were acknowledged by the authors and subsequently fixed.

Future plans include expanding the approach to generate a wider range of workloads and employing more complex methods for simulating file system failures.

References

- [1]. Ext4 data loss. Ubuntu repository. [Online]. Available at: <https://bugs.launchpad.net/ubuntu/+source/linux/+bug/317781>, accessed: 2025-10-31.
- [2]. IEEE/Open Group standard for information technology—portable operating system interface (POSIX™) base specifications, issue 8. IEEE/Open Group Std 1003.1-2024 (Revision of IEEE Std 1003.1-2017), 2024, pp. 1-4107.
- [3]. Böhme M., Cadar C., Roychoudhury A. Fuzzing: Challenges and reflections. *IEEE Software*, 2020, vol. 38, issue 3, pp. 79-86. [Online]. DOI: 10.1109/MS.2020.3016773.
- [4]. Kovalevsky V.M., Kechin V.V., Itsykson V.M. DIFFuzzer: Detecting File System Errors with Differential Grey-box Fuzzing. *Trudy ISP RAN/Proc. ISP RAS*, 2025, vol. 37, issue 4-2, pp. 31-46. [Online]. DOI: 10.15514/ISPRAS-2025-37(4)-17.
- [5]. Mohan J., Martinez A., Ponnappalli S., Raju P., Chidambaram V. CrashMonkey and ACE: Systematically testing file-system crash consistency. *ACM Trans. Storage*, 2019, vol. 15, issue 2. [Online]. DOI: 10.1145/3320275.
- [6]. ISO/IEC 25010:2023, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model. ISO/IEC JTC 1/SC 7, 2023, pp. 1-22. [Online]. Available at: <https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en>, accessed: 03.02.2026.
- [7]. open(2) – Linux manual page. The Linux Foundation. [Online]. Available at: <https://man7.org/linux/man-pages/man2/open.2.html>, accessed: 31.10.2025.
- [8]. Bornholt J. et al. Specifying and checking file system crash-consistency models. *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016, pp. 83-98. [Online]. DOI: 10.1145/2872362.2872406.

- [9]. (2015) Syzkaller: coverage-guided kernel fuzzer. Google. [Online]. Available at: <https://github.com/google/syzkaller>, accessed: 31.10.2025.
- [10]. Schumilo S., Aschermann C., Gawlik R., Schinzel S., and Holz T. kAFL: Hardware-Assisted feedback fuzzing for OS kernels. 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 167-182. [Online]. Available at: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/schumilo>, accessed: 03.02.2026.
- [11]. Izraelevitz J. et al. Basic performance measurements of the intel optane DC persistent memory module. arXiv preprint arXiv:1903.05714, 2019. [Online]. DOI: 10.48550/arXiv.1903.05714.
- [12]. Yang J. et al. An empirical guide to the behavior and use of scalable persistent memory. 18th USENIX Conference on File and Storage Technologies (FAST 20), 2020, pp. 169-182. [Online]. Available at: <https://www.usenix.org/conference/fast20/presentation/yang>, accessed: 03.02.2026.
- [13]. LeBlanc H. et al. Chipmunk: Investigating crash-consistency in persistent-memory file systems. Proceedings of the Eighteenth European Conference on Computer Systems, 2023, pp. 718-733. [Online]. DOI: 10.1145/3552326.3567498.
- [14]. Kalbfleisch S., Werling L., Bellosa F. Vinter: Automatic Non-Volatile Memory Crash Consistency Testing for Full Systems. 2022 USENIX Annual Technical Conference (USENIX ATC 22), 2022, pp. 933-950. [Online]. Available at: <https://www.usenix.org/conference/atc22/presentation/werling>, accessed: 03.02.2026.
- [15]. Jiao B., Goel A., Wang A.I.A. Silhouette: Leveraging Consistency Mechanisms to Detect Bugs in Persistent Memory-Based File Systems. 23rd USENIX Conference on File and Storage Technologies (FAST 25), 2025, pp. 407-423. [Online]. Available at: <https://www.usenix.org/conference/fast25/presentation/jiao>, accessed: 03.02.2026.
- [16]. Ridge T. et al. SibylFS: formal specification and oracle-based testing for POSIX and real-world file systems. Proceedings of the 25th Symposium on Operating Systems Principles, 2015, pp. 38-53. [Online]. DOI: 10.1145/2815400.2815411.
- [17]. Chen D. et al. Testing file system implementations on layered models. Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, 2020, pp. 1483-1495. [Online]. DOI: 10.1145/3377811.3380350.
- [18]. Kim S. et al. Finding semantic bugs in file systems with an extensible fuzzing framework. Proceedings of the 27th ACM Symposium on Operating Systems Principles, 2019, pp. 147-161. [Online]. DOI: 10.1145/3341301.3359662.
- [19]. Liu J. et al. SnapCC: Effective file system consistency testing using systematic state exploration. ACM Transactions on Software Engineering and Methodology, 2025, vol. 34, issue 8. [Online]. DOI: 10.1145/3718738.
- [20]. Symlink entry is not persisted after rename if system crashes, accessed: 2025-10-31. [Online]. Available at: <https://lore.kernel.org/linux-btrfs/ac949c74-90c2-4b9a-b7fd-1ffc5c3175c7@gmail.com/>, accessed: 03.02.2026.
- [21]. Directory is not persisted after writing to the file within directory if system crashes [Online]. Available at: <https://lore.kernel.org/linux-btrfs/03c5d7ec-5b3d-49d1-95bc-8970a7f82d87@gmail.com/>, accessed: 31.10.2025.
- [22]. File system corruption after renaming directory and creating a new file with same name if system crashes. [Online]. Available at: <https://lore.kernel.org/linux-btrfs/7bbc9419-5c56-450a-b5a0-efeae7457113@gmail.com/>, accessed: 15.12.2025.
- [23]. Directory is not persisted after creating 100s of files inside, writing to another file and renaming it if system crashes. [Online]. Available at: <https://lore.kernel.org/linux-btrfs/84c4e713-85d6-42b9-8dcf-0722ed26cb05@gmail.com/>, accessed: 15.12.2025.
- [24]. Increased reports since 6.15.3 of corruption within the log tree. [Online]. Available at: <https://lore.kernel.org/linux-btrfs/fce139db-4458-4788-bb97-c29acf6cb1df@cachyos.org/>, accessed: 31.10.2025.
- [25]. iomap: Fix broken data integrity guarantees for O_SYNC writes. [Online]. Available at: <https://lore.kernel.org/all/20250730102840.20470-2-jack@suse.cz/>, accessed: 31.10.2025.
- [26]. Fedora KDE no longer booting, likely filesystem (BTRFS) corruption. The Fedora Project. [Online]. Available at: <https://discussion.fedoraproject.org/t/fedora-kde-no-longer-booting-likely-filesystem-btrfs-corruption/157232>, accessed: 31.10.2025.

Информация об авторах / Information about authors

Вячеслав Максимович КОВАЛЕВСКИЙ – аспирант и инженер Института прикладных компьютерных наук университета ИТМО. Сфера научных интересов: динамический и статический анализ программ, фаззинг.

Vyacheslav Maksimovich KOVALEVSKY – postgraduate student and engineer at the Institute of Applied Computer Science of the ITMO University. Research interests: dynamic and static software analysis, fuzzing.

Валерий Владимирович КЕЧИН – аспирант и инженер Института прикладных компьютерных наук университета ИТМО. Сфера научных интересов: динамический и статический анализ программ, фаззинг.

Valeriy Vladimirovich KECHIN – postgraduate student and engineer at the Institute of Applied Computer Science of the ITMO University. Research interests: dynamic and static software analysis, fuzzing.

Александр Сергеевич ЯНИН – студент магистратуры и инженер Института прикладных компьютерных наук университета ИТМО. Сфера научных интересов: динамический и статический анализ программ, фаззинг.

Alexander Sergeevich YANIN – graduate master student and engineer at the Institute of Applied Computer Science of the ITMO University. Research interests: dynamic and static software analysis, fuzzing.

Владимир Михайлович ИЦЫКСОН – кандидат технических наук, доцент Института прикладных компьютерных наук университета ИТМО. Сфера научных интересов: статический и динамический анализ программ, верификация программного обеспечения, методы обнаружения дефектов в исходном коде, методы автоматизации тестирования программ.

Vladimir Mikhailovich ITSYKSON – Cand. Sci. (Tech.), associate professor of the Institute of Applied Computer Science of the ITMO University. Research interests: static and dynamic software analysis, software verification, methods for detecting defects in source code, methods for automating software testing.

DOI: 10.15514/ISPRAS-2026-38(1)-8



Использование аугментации при обучении на некомплектной выборке

^{1,2} Д.О. Лазарев, ORCID: 0000-0002-6253-6447 <lazarev@ispras.ru>

¹ А.В. Шокуров, ORCID: 0000-0002-6801-7728 <shok@ispras.ru>

¹ С.А. Фомин, ORCID: 0000-0002-1151-2189 <fomin@ispras.ru>

¹ Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

² Московский физико-технический институт (национальный исследовательский университет),

Россия, 141701, Московская область, г. Долгопрудный, Институтский переулок, д.9.

Аннотация. Исследуется влияние метода аугментации и балансировки некомплектной выборки, содержащей пропущенные значения признаков, на точность прогноза. Результаты для некомплектной выборки сравниваются с результатами для выборки, значения признаков которой полностью заполнены. Предложен новый алгоритм сэмплирования с удалением для аугментации и балансировки некомплектной выборки. В рамках теории вероятностно приближенно корректного (ВПК) обучения авторами была исследована задача обучения на некомплектной выборке. Был оценен рост размерности Вапника-Червоненкиса множества функций при заполнении пропущенных значений фиксированным значением из конечного множества. Было доказано, что требуемый размер выборки для ВПК обучения с достаточной точностью, растет логарифмически медленно с ростом размера этого множества. Установлено, что метод аугментации сэмплированием с удалением, позволяет получить наиболее высокую сбалансированную точность для некомплектных линейно разделимых выборок малого размера. При обучении на выборках среднего и большого размера, во всех рассмотренных случаях, аугментация позволяет получить большее увеличение целевых метрик для некомплектных выборок, чем для полностью заполненных. Таким образом, особенно эффективна аугментация при обучении на некомплектной выборке.

Ключевые слова: машинное обучение; аугментация данных; аугментация табличной выборки; обучение на некомплектной выборке; обучение на выборке малого размера; вероятно приближенно корректное обучение; вложенная кросс-валидация.

Для цитирования: Лазарев Д.О., Шокуров А.В., Фомин С.А. Использование аугментации при обучении на некомплектной выборке. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 93–112. DOI: 10.15514/ISPRAS-2026-38(1)-8.

Благодарности: Исследования поддержаны фондом отдела теоретической информатики Института системного программирования им. В.П. Иванникова РАН. Результаты получены с использованием услуг Центра коллективного пользования Института системного программирования им. В.П. Иванникова РАН – ЦКП ИСП РАН.

Data Augmentation for Machine Learning on Missing Data

^{1,2} D.O. Lazarev, ORCID: 0000-0002-6253-6447 <lazarev@ispras.ru>

¹ A.V. Shokurov, ORCID: 0000-0002-6801-7728 <shok@ispras.ru>

¹ S.A. Fomin, ORCID: 0000-0002-1151-2189 <fomin@ispras.ru>

¹ *Ivannikov Institute for System Programming of the Russian Academy of Science,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

² *Moscow Institute of Physics and Technology (National Research Institute),
9, Institute alley, Dolgoprudny, Moscow Region, 141701, Russia.*

Abstract. The impact of missing data augmentation and class balancing methods on the prediction accuracy is studied. Results for missing and complete samples are compared. A new algorithm called “drop and replace” for missing data imputation, augmentation and class balancing is proposed. The quality of the “drop and replace” algorithm is compared to the quality of efficient and widely studied augmentation methods, such as SMOTE and mixup. For comparison, 7 samples with sizes from 38 up to 70000 are used. The target metric is balanced accuracy, and the nested cross validation algorithm is used for validation. For the imputation of samples of small sizes, the mean imputation method is used. For the imputation of average or large samples, the MICE method of multiple imputation by chained equations is used.

The problem of training machine learning algorithms on missing data is studied in the probably approximately correct learning setting, or PAC learning. The increase of Vapnik-Chervonenkis dimension, or VC dimension, of the function set due to data imputation with one value from the finite filling set is evaluated. It was shown that the needed sample size for PAC learning with sufficient accuracy, grows logarithmically slow with the increase of the filling set size.

It was proved that “drop and replace” augmentation allows to maximize balanced accuracy when learning on small linearly separable samples containing missing values. For larger samples, SMOTE and mixup augmentation methods are most effective. For 8 samples out of 11, data augmentation, which increments sample size, increases classification accuracy; in 2 of 3 other cases, sample balancing alone can increment classification accuracy; in the final case, the initial sample is balanced and augmentation cannot increase classification accuracy. When learning on average and large-sized samples, in all 3 cases, augmentation more significantly improves accuracy for missing datasets, then for complete ones. So, augmentation is more effective when learning on missing data.

Keywords: machine learning; data augmentation; tabular data; machine learning on missing data; learning on small sized dataset; probably approximately correct learning; nested cross validation.

For citation: Lazarev D.O., Shokurov A.V., Fomin S.A. Data augmentation for machine learning on missing data. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 93-112 (in Russian). DOI: 10.15514/ISPRAS-2025-38(1)-8.

Acknowledgements. The research was supported by the fund of the Department of Theoretical Computer Science of Ivannikov Institute for System Programming of the RAS. The results were obtained using the services of the Shared Use Center of the Ivannikov Institute for System Programming of the PAS – the SUC ISP RAS.

1. Введение

Обучение на некомплектной выборке, или на выборке, содержащей незаполненные значения некоторых признаков, требуется во многих реальных приложениях, включая медицину и биологию [1-2]. Наличие пропущенных значений в выборке может быть обусловлено разными причинами, например, неправильным функционированием измерительных приборов, отсевом участников в продолжение наблюдения или нежеланием участников исследования отвечать на вопрос, например, о своем доходе.

Сформулируем постановку задачи обучения на некомплектной выборке.

Построить алгоритм машинного обучения, который позволяет строить прогноз класса для некомплектных объектов. Требуется выбрать способ валидации, минимизирующий

переобучение, и получить несмещенную оценку точности прогноза для выборки малого размера. При обучении, восстановить пропущенные значения и сделать аугментацию обучающей выборки. Исследовать влияние способа аугментации выборки и метода заполнения пропущенных значений на точность прогноза. При исследовании, будут использоваться несколько выборок.

В работе [1] приводится следующая классификация некомплектных выборок по вероятности отсутствия значения признака:

- Пропущенные значения полностью случайны (missing completely at random, MCAR). Вероятность отсутствия каждого значения признака не зависит от этого значения и от класса, которому соответствующий объект принадлежит.
- Пропущенные значения случайны (missing at random, MAR). Вероятность отсутствия каждого значения не зависит от самого этого значения, но может зависеть от класса, которому объект принадлежит.
- Пропущенные значения неслучайны (missing not at random, MNAR). Вероятность того, что значение пропущено, может зависеть от этого значения.

Отметим, что во всех случаях, включая MCAR и MAR, вероятности отсутствия значения для любых двух различных признаков одной выборки могут различаться.

Метод вложенной кросс-валидации, используемый для валидации модели машинного обучения, позволяет получать несмещенную оценку точности для любого типа некомплектной выборки. Но при восстановлении пропущенных значений предполагается, что некомплектная выборка принадлежит одному из типов MCAR или MAR.

Таким образом, в настоящей работе рассмотрены некомплектные выборки, являющиеся либо MAR, либо MCAR выборками. Исследуем выборки следующих типов: 2 некомплектных выборки [3, 4] типа MAR, для которых вероятность отсутствия признака зависит от класса и 5 полностью заполненных выборок [5, 6, 7, 8, 9], которые преобразуем в некомплектные MCAR-выборки случайным удалением значений признаков.

При восстановлении, пропущенные значения признаков заполняются неточно. Поэтому, в настоящей работе предложена аугментация выборки с восстановлением пропущенных значений. Таким образом, восстанавливаемые пропущенные значения из обучающей выборки, заполняются не одним фиксированным значением, а целым множеством значений из вероятностного распределения, построенного на основе заполненных значений соответствующего признака.

Похожий подход применялся в работе [10], где адаптивная аугментация некомплектной выборки с использованием генеративно-состязательных сетей позволяла повысить точность прогноза в случае, когда тестовая и обучающая выборки имели различные распределения пропущенных значений.

Наиболее часто аугментация применяется при обучении на таких типах многомерных выборок, как изображения [11], аудиоданные [12] или для выборок, возникающих при обучении больших языковых моделей [13]. Однако, многие методы аугментации данных, применяемые преимущественно для многомерных выборок, такие, например, как метод mixup [14] “перемешивания” с помощью построения выпуклых комбинаций из пар объектов, могут также применяться при обучении на табличных выборках. Метод SMOTE [15], наиболее широко применяемый для балансировки выборки, т.е. для того, чтобы сделать равными размеры классов в выборке, также может быть использован для аугментации табличной выборки. Все рассматриваемые в настоящей работе методы аугментации данных, также позволяют балансировать выборку.

В настоящей работе предложен метод сэмплирования с удалением для аугментации и балансировки выборки. Наиболее высокую точность данный метод аугментации позволяет получить при обучении на некомплектной линейно разделимой выборке малого размера.

Предлагаемый подход аугментации данных для обучения на некомплектной выборке является новым методом, который ранее в литературе не встречался.

Также исследовалось влияние аугментации некомплектной выборки на точность прогноза. Результаты для некомплектной выборки сравнивались со случаем полностью заполненной выборки. Для 3 из 4 некомплектных выборок размера более 500, аугментация позволяла улучшить точность прогноза. Для оставшейся выборки РРМ1, было достаточно балансировки выборки для увеличения точности. Для полностью заполненных выборок, аугментация позволяла увеличить точность прогноза в 2 случаях из 4, а увеличение точности при этом наблюдалось менее существенное, чем для некомплектных выборок. Таким образом, проведенные исследования свидетельствуют о том, что для выборок достаточно большого размера, аугментация позволяет увеличить точность прогноза для некомплектных выборок.

В следующем разделе, приведем формальную постановку решаемой задачи в рамках теории [16] Валианта вероятно приближенно корректного обучения (ВПК обучения) и теоретически исследуем ВПК-обучаемость задачи обучения на некомплектной выборке.

2. Исследование и постановка задача обучения на некомплектной выборке в рамках теории ВПК обучения

Теория вероятно приближенно корректного (ВПК) обучения [16, 17] позволяет теоретически исследовать алгоритмы машинного обучения. В рамках классической задачи обучения с учителем, имея выборку $\{(x_i, y_i)_{i=1}^m\}$, $x_i \in X, y_i \in Y$, теория ВПК обучения позволяет выбрать множество $\mathcal{H}$ гипотез, которому должна принадлежать обобщающая функция $h: X \rightarrow Y$, являющаяся результатом обучения, таким образом, чтобы одновременно уменьшить ошибку аппроксимации и избежать переобучения.

Элементы теории ВПК обучения, включающие определения агностической ВПК-обучаемости, VC-размерности Вапника и Червоненкиса и лемму Зауэра-Шелаха, приведены в приложении 1.

Сформулируем задачу обучения на некомплектной выборке в терминах теории ВПК обучения [17]. Пространство признаков X_i для каждого признака i дополним значением $\perp$, означающим то, что данный признак пропущен. Соответственно, множество гипотез определим на множестве $X^\perp = \prod_{i=1}^n (X_i \cup \perp)$ некомплектных выборок.

Дадим общую постановку, взяв за основу некоторое множество гипотез $\mathcal{H}$, определенное только на полностью заполненных объектах из X . При этом, заполненные значения исходной выборки могут содержать случайный шум. Аналогично [18], будем использовать алгоритм $Fill: X^\perp \rightarrow X$ для восстановления пропущенных значений.

Приведем определение агностического ВПК обучения на некомплектной выборке и оценим сверху необходимый размер выборки для ВПК обучения с заданными параметрами вероятности и аппроксимации.

Пусть $X = \prod_{i=1}^n X_i$ – исходное пространство объектов (считаем значения признаков X_i вещественными), а $X^\perp = \prod_{i=1}^n (X_i \cup \perp)$ – пространство некомплектных выборок. Пусть также задано множество $CFill$ функций $Fill: X^\perp \rightarrow X$ восстановления пропущенных значений. Пусть $\mathcal{H}$ – множество гипотез, являющееся классом характеристических функций $h: X \rightarrow \{0,1\}$.

Определение ВПК обучения на некомплектной выборке требует, чтобы множество $CFill$ функций для восстановления пропущенных значений было задано. Определим агностическое обучение на некомплектной выборке $X^\perp$, используя обучение на полностью заполненной выборке X для множества гипотез $\mathcal{H}$, являющегося классом характеристических функций $h: X \rightarrow \{0,1\}$. Определим на некомплектной выборке $X^\perp$ класс гипотез $\mathcal{H}'(X^\perp) = (\mathcal{H}(CFill(X^\perp)))$: $X^\perp \rightarrow \{0,1\}$ как композицию восстановления пропущенных значений

алгоритмов $Fill$ из класса $CFill$ и применения гипотезы из множества $\mathcal{H}$ к комплектному объекту из множества X .

Определение (агностическое ВПК обучение на некомплектной выборке). Говорим, что множество $(\mathcal{H}, CFill)$ – агностически ВПК-обучаемое, тогда и только тогда, когда множество гипотез $\mathcal{H}'(X^\perp) = (\mathcal{H}(CFill(X^\perp)))$: $X^\perp \rightarrow \{0,1\}$, получаемых в результате применения к некомплектным объектам сначала одной функции $Fill$ для восстановления пропущенных значений из множества $CFill$, а затем – некоторой характеристической функции $h \in \mathcal{H}$, является агностически ВПК-обучаемым.

Отметим, что бинарные метки объектов $y_i \in \{0,1\}$ – полностью заполнены для всех i .

Следующая фундаментальная теорема ВПК обучения дает критерий обучаемости для некомплектной выборки.

Теорема 1 (Фундаментальная теорема ВПК обучения). Для некомплектной выборки, множество $(\mathcal{H}, CFill)$ – агностически ВПК-обучаемое тогда и только тогда, когда $VCdim(\mathcal{H}'(\mathcal{H}, CFill)) < \infty$.

Приведем полученные в настоящей работе новые теоретические результаты, связанные с VC -размерностью некомплектной выборки при заполнении пропущенных значений фиксированными элементами из некоторого, заранее известного множества.

Лемма 1 (Агностическая ВПК-обучаемость на некомплектной выборке). Пусть $CFill_{const}$ – множество методов $Fill: X^\perp \rightarrow X$ восстановления пропущенных значений, представляет из себя множество методов заполнения константой из множества $X_{fill} = \{x_{fill}^j\}_{j=1}^k, x_{fill}^j \in X \forall 1 \leq j \leq k$ конечного размера k . Пусть $\mathcal{H}$ – множество гипотез $h: X \rightarrow \{0,1\}$, имеющее конечную VC -размерность: $VCdim(\mathcal{H}) = d$.

Тогда выполняется следующее:

1. При $|X_{fill}| = 1$, $VCdim(\mathcal{H}'(\mathcal{H}, CFill_{const})) \leq d$.
2. При $|X_{fill}| = k > 1$, $VCdim(\mathcal{H}'(\mathcal{H}, CFill_{const})) < 2d \log_2 (k^{1/d} + 31)$.

(В лемме, $\mathcal{H}'(\mathcal{H}, CFill_{const}) = \mathcal{H}(CFill_{const}(X^\perp))$ – множество гипотез, получаемых в результате применения к некомплектным объектам сначала одного метода $Fill$ для восстановления пропущенных значений из множества $CFill_{const}$, а затем – некоторой характеристической функции $h \in \mathcal{H}$).

Лемма 1 доказана в приложении 2.

Практическая значимость леммы 1 обусловлена медленным ростом мультипликатора $2 \log_2 (k^{1/d} + 31)$ в неравенстве $VCdim(\mathcal{H}'(\mathcal{H}, CFill_{const})) < 2d \log_2 (k^{1/d} + 31)$ в зависимости от размера k множества X_{fill} . При заполнении каждого из n_f пропущенных значений множеством точек из сетки, являющейся декартовым произведением n_f отрезков, содержащих N точек, мультипликатор $2 \log_2 (k^{1/d} + 31)$ в ограничении сверху у $VCdim(\mathcal{H}'(\mathcal{H}, CFill_{const}))$ растет со скоростью $O(\frac{n_f}{d} \log N)$, т.к. $k = N^{n_f}$.

Согласно количественной формулировке фундаментальной теоремы ВПК обучения [19], для множества гипотез $\mathcal{H}$: $VCdim(\mathcal{H}) = d < \infty$, существуют такие константы $c_1 > c_0 > 1$, что для случайной выборки размера не менее $m(\epsilon, \delta)$, удовлетворяющего неравенству $c_0 \frac{d + \ln(1/\delta)}{\epsilon^2} < m(\epsilon, \delta) < c_1 \frac{d + \ln(1/\delta)}{\epsilon^2}$, выполняется условие агностического ВПК обучения: с вероятностью не менее $1 - \delta$, минимизирующий эмпирический риск ВПК-обучатель, возвращает такую гипотезу $h \in \mathcal{H}$, для которой выполняется: $L_D(h) \leq L_D(\mathcal{H}) + \epsilon$.

Соответственно, при обучении на некомплектной выборке, при котором сначала заполняем пропущенное значение одной из N^{n_f} констант из сетки, требуемый размер $m(\epsilon, \delta)$ для обучения с заданной точностью увеличится в $\frac{n_f}{d} \ln N$ раз по сравнению со случаем полностью заполненной выборки при выборе точки для заполнения пропущенных значений по сетке и

при выборе гипотезы $h \in \mathcal{H}$. Иллюстрация заполнения пропущенных значений элементами конечного множества приведена на рис. 1.

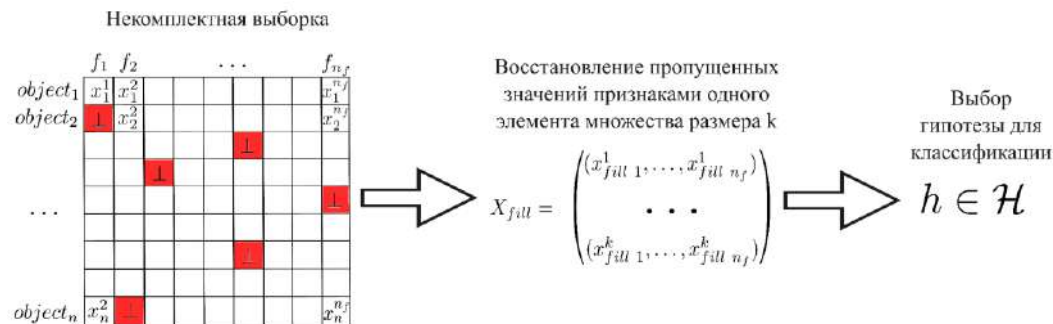


Рис. 1. Заполнение пропущенных значений элементом множества X_{fill} при обучении на некомплектной выборке.

Fig. 1. Missing data imputation with an element of finite set X_{fill} when learning to solve classification task.

2.1 Формальная постановка задачи обучения на некомплектной выборке

Приведем формальную постановку решаемой задачи обучения на некомплектной выборке и исследования влияния аугментации выборки на точность прогноза. Дана выборка $S = \{(x_i, y_i)\}_{i=1}^n$, являющаяся объединением численных представлений объектов $x_i \in X^1, i \in \{1, \dots, n\}$ и ответов $y_i \in Y, i \in \{1, \dots, n\}$, принадлежащих конечному множеству Y . Все элементы выборки S выбраны независимо в совокупности согласно некоторому неизвестному распределению D . Объекты могут быть некомплектными, т.е. часть значений признаков может быть пропущена. Полностью заполненные ответы y_i позволяют определить, к какому классу относится объект.

Целью работы является:

- Построение функции прогноза класса $\hat{y}(x, S)$, которая для каждого, возможно – некомплектного объекта $x \in X^1$, строит прогноз ответа. При этом стремимся минимизировать ожидаемую ошибку $E_{(x,y) \sim D}(l(\hat{y}(x, S), y)) \rightarrow \min$ для метрики $l: Y^2 \rightarrow \mathbb{R}$.
- Выбор целевой функции и метода валидации для получения несмещенной оценки точности прогноза с малой дисперсией.
- Разработка и сравнение методов аугментации некомплектной выборки, использующих методы восстановления пропущенных значений и позволяющих получать высокую точность прогноза при обучении на некомплектной выборке алгоритма машинного обучения. Сравнение результатов со случаем полностью заполненной выборки.

3. Обзор существующих решений

Приведем обзор существующих решений и сравним их с решением, предложенным в настоящей работе.

В работе [20], в рамках фреймворка VIME, предложен метод аугментации табличной выборки, позволяющий получить высокую точность прогноза. Метод аугментации также может быть применен для восстановления пропущенных значений. Идея искажения выборки с последующим ее восстановлением из фреймворка VIME была использована в настоящей работе при построении метода аугментации сэмплирования с удалением. Однако, в отличие

от настоящей работы, где исследуется обучение с учителем, фреймворк VIME предназначен для использования при обучении с частичным подкреплением или при обучении без учителя. При обучении с частичным подкреплением, при обучении используются как размеченные, так и неразмеченные объекты выборки, а при обучении без учителя, все объекты выборки не размечены.

В работе [21], предложен автокодировщик с шумоподавлением для восстановления пропущенных значений. Такой автокодировщик обучен восстанавливать пропущенные значения путем подачи ему на вход зашумленной выборки.

Автокодировщики позволяют получить более высокую точность прогноза при наличии зависимости вероятности пропуска значения выборки от данного значения [22]. Таким образом, в случае MNAR, автокодировщики позволяют более точно восстанавливать пропущенные значения по сравнению с другими методами восстановления пропущенных значений. В случаях MCAR и MAR, метод восстановления пропущенных значений максимизацией эмпирического правдоподобия, превосходит автокодировщик по точности прогноза.

Точность обучения при использовании автокодировщика с шумоподавлением для восстановления пропущенных значений сравнивалась в [21] с точностью при использовании метода MICE цепных уравнений [23]. Для некомплектной выборки типа MNAR, восстановление пропущенных значений автокодировщиком с подавлением шума, позволяет получить более высокую точность прогноза, чем восстановление пропущенных значений методом MICE цепных уравнений.

В отличие от работы [21], в настоящей работе используются некомплектные выборки типов MCAR и MAR. Для восстановления пропущенных значений используется метод MICE цепных уравнений. Сравниваются между собой в настоящей работе не методы восстановления пропущенных значений, а методы аугментации выборки. Также, точность обученной модели после восстановления пропущенных значений, сравнивается со случаем обучения на некомплектной выборке.

В настоящей работе, также исследована ВПК-обучаемость в случае обучения на некомплектной выборке. Аналогичная задача исследовалась ранее, например в [18, 24]. Отличие предложенной в настоящей работе постановки заключается в том, что в [18] исследуется задача ВПК обучения, а не агностического ВПК обучения. Класс функций $\mathcal{H}$ представлен в [18] булевыми формулами. В работе [24] делается предположение о возможности сколь угодно точного восстановления пропущенных значений для выборки достаточно большого размера. Таким образом, в моделях [18, 24] случай зашумленной некомплектной выборки не рассматривается, что ограничивает практическое применение полученных результатов. В настоящей работе, исследуется агностическая ВПК обучаемость на некомплектной выборке, что позволяет моделировать обучение на зашумленной некомплектной выборке.

4. Методы обучения и валидации на некомплектной выборке

При обучении моделей машинного обучения на некомплектной выборке, выделим следующие этапы:

1. Выбор метода обучения на некомплектной выборке
2. Восстановление пропущенных значений в обучающей и в тестовой выборках
3. Выбор целевой метрики для оценки качества прогноза
4. Выбор схемы валидации модели
5. Выбор метода аугментации выборки

На начальном этапе, нужно сделать выбор метода обучения: будем ли обучаться на некомплектной выборке, или предварительно обучим алгоритм восстановления

пропущенных значений. Выбрав восстановление пропущенных значений, предварительно, будем удалять из выборки объекты, у которых пропущено слишком много значений, т.к. точность прогноза данных объектов может быть значительно ниже среднего.

Выбор целевой метрики должен учитывать несбалансированность классов выборок. Выбор метода валидации модели машинного обучения, особенно важен для выборок малого размера, для которых стандартная схема разбиения исходной выборки на обучающую и тестовую выборки, дает слишком большую дисперсию полученной оценки точности прогноза.

4.1 Обучение на некомплектной выборке

Согласно [2], для некомплектной выборки, может быть использован один из следующих подходов для обучения ML-модели:

- Удаление некомплектных объектов, или тех объектов, у которых пропущены некоторые значения признаков. Данный подход может привести к удалению значительного числа объектов из выборки. Позволяет получить несмещенную оценку точности только в случае, когда выборка относится к типу MCAR или MAR. В случае MAR нужно сделать балансировку выборки, т.к. после удаления некомплектных объектов, баланс классов может измениться.
- Обучение ML-модели, предназначенной для работы с некомплектной выборкой. К таким моделям относятся алгоритмы машинного обучения, использующие лишь попарные корреляции между признаками и модель CatBoost [25].
- Восстановление пропущенных значений (англ. imputation).

Часть алгоритмов машинного обучения не предназначена для работы на некомплектных выборках, другие алгоритмы могут показывать значительное снижение точности при обучении на некомплектной выборке по сравнению с обучением на выборке, пропущенные значения в которой восстановлены.

Поэтому, в настоящей работе, применялись первый и последний подходы. Из обучающей и из тестовой выборок удалялись объекты, у которых пропущено слишком большое число наиболее значимых признаков. Выбор наиболее значимых признаков для обучения с использованием t-критерия Уэлча позволял уменьшить количество удаляемых объектов. Затем пропущенные значения в выборке восстанавливались.

Результаты сравнивались со случаем обучения на некомплектной выборке без предварительного восстановления пропущенных значений. Для рассмотренной выборки, точность прогноза снизилась при обучении на некомплектной выборке по сравнению со случаем восстановления пропущенных значений перед обучением ML-модели.

4.2 Восстановление пропущенных значений

Разделим, согласно [2], методы восстановления пропущенных значений в выборке на 2 типа:

- Одномерные методы. К данным методам относится восстановление пропущенного значения либо средним медианным значением для непрерывных признаков, или модой для категориальных. Восстановление пропущенного значения признака происходит с использованием имеющихся в исходной выборке значений этого же признака признаков у других объектов.
- Многомерные методы восстановления пропущенных значений. Существуют различные методы многомерного восстановления: метод k ближайших соседей KNN [26], восстановление с использованием метода нечеткой кластеризации C-средних FCM [27], восстановление с использованием байесовского метода главных компонент [28], восстановление методом MICE цепных уравнений [23] и др. При

восстановлении пропущенных значений одним из данных методов, учитываются значения заполненных признаков в исходной выборке у того объекта, пропущенные значения которого восстанавливаются.

В [29] сравнивались 6 методов восстановления пропущенных значений, включая одномерный метод заполнением средним и многомерные подходы: метод k ближайших соседей, метод FCM нечеткой кластеризации C -средними и байесовский метод главных компонент. Наиболее высокая точность достигалась у двух последних методов, причем метод FCM позволял получить наибольшую точность при восстановлении значений выборок малого размера.

В работе [30], исследовались методы восстановления пропущенных значений некомплектной выборки типа MNAR. Сравнивались методы восстановления пропущенных значений средним, MICE цепных уравнений, k ближайших соседей, SoftImpute. Наиболее высокую точность восстановления пропущенных значений позволяет получить MICE, а метод k ближайших соседей позволяет получить наиболее высокую точность классификации. В [31], к сравнению добавлено восстановление пропущенных значений с использованием генеративного искусственного интеллекта. Последние показывали лучшую точность при большей доле пропущенных значений в выборке (60% и выше), тогда как при относительно малой доле пропущенных значений (40% и менее), наибольшая точность восстановления пропущенных значений достигается при использовании метода MICE цепных уравнений. Согласно проведенным в работе [32] вычислительным экспериментам, наибольшую точность восстановления пропущенных значений позволяют получить различные модификации метода K ближайших соседей.

В настоящей работе, сравнивались методы восстановления пропущенных значений средним, метод MICE цепных уравнений, метод нечеткой кластеризации C -средних FCM, метод K ближайших соседей и метод Miss Forest [33]. Для выборок малого размера, содержащих до 50 объектов, наибольшую точность классификации позволял получить метод восстановления пропущенных значений средним; а для выборок большего размера, содержащих более 500 объектов в обучающей выборке, наиболее эффективен метод MICE цепных уравнений. Поэтому методы восстановления пропущенных значений средним и методом MICE цепных уравнений, и были использованы в настоящей работе.

Отметим, что имеется различие при восстановлении пропущенных значений в обучающей и в тестовой выборке. В первом случае, класс, к которому принадлежит объект, можем использовать при восстановлении пропущенных значений; в случае восстановления пропущенных значений тестовой выборки, класс объекта не может быть использован, в противном случае возникает проблема утечки данных.

4.3 Используемые метрики

Все выборки, кроме MNIST, являются несбалансированными. Наибольший дисбаланс классов наблюдается для выборки PPMI [3], где 91.7% объектов принадлежат одному классу из двух. Поэтому, стандартной точности *accuracy* недостаточно для оценки точности прогноза на несбалансированной выборке [10]. Для преодоления этого недостатка будем использовать метрики сбалансированная точность *balanced accuracy* и коэффициент корреляции Мэтьюса [34] для оценки точности прогноза. Сбалансированная точность равна точности *accuracy* в случае сбалансированной выборки.

4.4 Выбор метода валидации модели

При использовании схемы кросс-валидации при обучении с подкреплением на выборке малого размера, из-за переобучения полученная точность может падать с ростом размера исследуемой выборки [35]. Происходит это из-за переобучения, т.к. исходная выборка

используется как для обучения модели, так и для выбора ее гиперпараметров. Поэтому на выборках малого размера, очень важен выбор схемы валидации, позволяющий получить несмещенную оценку точности, минимизируя при этом переобучение.

Валидация ML-модели на отложенной тестовой выборке, позволяет получить несмещенную оценку точности. Однако, при этом, для выборки малого размера, получается большая дисперсия оценки точности, т.к. на точность прогноза будет существенно влиять выбор тестовой выборки. Таким образом, валидацию на отложенной тестовой выборке будем применять лишь для выборок большого размера, содержащих более 10000 объектов.

Согласно [36], для получения несмещенной оценки точности с малой дисперсией, следует использовать метод вложенной кросс-валидации [37] для сравнения методов аугментации выборок малого размера. Будем использовать этот метод для всех выборок малого и среднего размера, содержащих менее 2000 объектов до аугментации.

В схеме вложенной кросс-валидации, выделяются следующие шаги:

1. Выбор обучающей и тестовой выборок. Элементы обучающей выборки выбираются из исходной случайным сэмплированием некоторого числа объектов без повторения. В тестовую выборку входят все объекты, не вошедшие в обучающую. В настоящей работе, размер обучающей и тестовой выборки на каждом шаге вложенной кросс-валидации составляет около 90% и 10% от размера исходной выборки соответственно.
2. Затем, обучающая выборка балансируется и аугментируется одним из 5 методов, качество которых сравнивается между собой и со случаем отсутствия аугментации и балансировки выборки. Далее, на обучающей выборке после аугментации обучается стандартизатор, который будет применяться как при обучении, так и при прогнозе на тестовой выборке. На выбранных признаках обучается модель машинного обучения, например, машина опорных векторов (SVM) или градиентный бустинг. Гиперпараметры модели выбираются с использованием кросс-валидации на обучающей выборке. Данная кросс-валидация называется внутренней кросс-валидацией в схеме вложенной кросс-валидации.

В дальнейшем, шагом вложенной кросс-валидации, будем называть одно повторение шагов 1, 2.

3. На данном шаге выполняется внешняя кросс-валидация. Шаги 1, 2 повторяются большое число раз (в проведенных вычислительных экспериментах, число повторений составляет 200 для малых выборок размером менее 50 объектов и 20 раз – на больших выборках, размером более 500). Каждый раз происходит обучение для нового случайно выбранного обучающего набора объектов. Точность прогноза класса оценивается на объединении всех тестовых выборок, полученных на шаге внутренней кросс-валидации. Некомплектные объекты со слишком большим числом пропущенных значений из выборки удаляются, а пропущенные значения для оставшихся объектов восстанавливаются. Итоговая точность оценивается на объединении прогнозов по всем тестовым выборкам. При этом, каждый объект с достаточным количеством заполненных значений признаков, входит в это объединение прогнозов одинаковое количество раз.

Диаграмма схемы вложенной кросс-валидации, используемой в настоящей работе, приведена на рис. 2.

При обучении на 5 выборках, размер которых менее 2000, для оценки точности прогноза ML-модели, в настоящей работе используем схему вложенной кросс-валидации. Для оставшихся 2 выборок, размер которых превосходит 10000, используем независимую тестовую выборку для валидации. Согласно [38], обе данные схемы валидации позволяют получить несмещенные оценки точности. Для выборки малого размера, оценка точности на

независимой тестовой выборке может быть неточной из-за сильной зашумленности получаемой таким образом оценки. Поэтому, при обучении на малых выборках, важно использовать схему вложенной кросс-валидации.

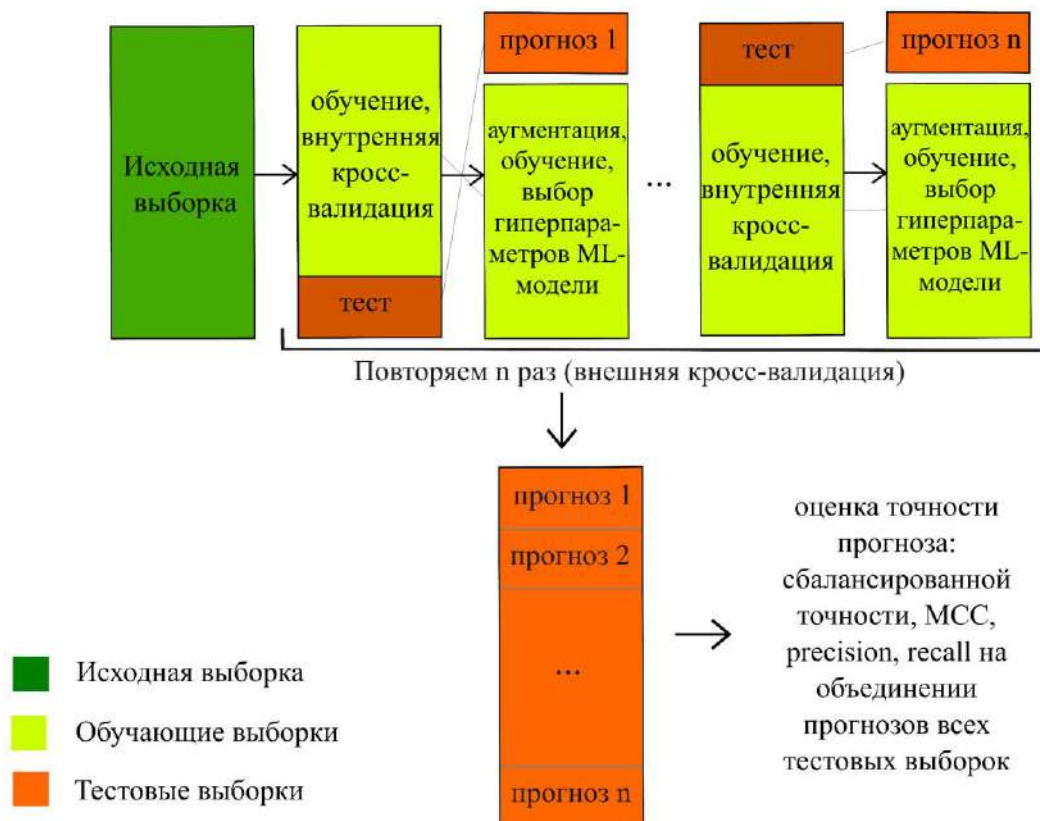


Рис. 2. Схема вложенной кросс-валидации.

Fig. 2. Nested cross validation scheme.

4.5 Аугментация выборок

Сравним предложенный в настоящей работе метод аугментации сэмплированием с удалением и другие методы аугментации и балансировки выборки. Эти методы аугментации можно разделить на 2 категории: во-первых, широко применяемые используемые ранее методы аугментации SMOTE и mixup, и, во-вторых, простые методы, включающие в себя минимально изменяющий выборку подход сэмплирования без удаления и строящий случайные комбинации значений признаков каждого класса метод случайного сэмплирования. Ниже подробно опишем все методы аугментации выборок, которые будем сравнивать между собой и со случаем отсутствия аугментации:

1. **Аугментация сэмплированием без удаления.** Строим синтетическую выборку случайным сэмплированием из исходной выборки нужного числа объектов, у которых заполнено достаточное количество значений признаков. При наличии пропущенных значений в выборке, используется метод MICE цепных уравнений, или одномерный метод восстановления пропущенных значений средним значением. К восстановленному значению признака добавляется также случайный шум с меньшей дисперсией, чем дисперсия значения соответствующего признака в исходной некомплектной выборке. Особенность подхода заключается в том, что, за

исключением восстановления пропущенных значений, в выборку не добавляются новые синтетические значения, а вместе с ними и новые связи между значениями признаков.

2. **Аугментация сэмплированием с удалением.** Сначала также, как и при аугментации сэмплированием без удаления, сэмплируем нужное число объектов из имеющейся, возможно некомплектной, выборки. Далее удаляем каждое заполненное значение признака с вероятностью $\frac{1}{2}$, после чего восстанавливаем все пропущенные значения. Преимуществом данного подхода является как сохранение части заполненных значений некомплектной выборки, так и добавление новых, зашумленных значений. Последнее можно рассматривать как регуляризацию модели машинного обучения [39] на уровне выборки.

Похожий подход, заключающийся в искажении и последующем восстановлении выборки, применяется и во фреймворке VIME [20] обучения со слабым подкреплением.

3. **Аугментация случайным сэмплированием**, или случайным выбором значений признаков. Простейший метод аугментации, где новые объекты строим случайным сэмплированием из набора значений каждого признака для каждого класса исходной выборки. При данном методе аугментации, значение каждого признака синтетического объекта выбирается из соответствующего класса исходной выборки случайным образом. Недостаток такого подхода заключается в том, что могут теряться важные для прогноза зависимости и связи между значениями признаков. Метод может давать высокую точность прогноза в случае малой корреляции между значениями различных признаков внутри классов.

Отличие подхода от аугментации сэмплированием без удаления заключается в сэмплировании отдельных значений признаков, а не целых объектов.

4. **Использование метода SMOTE** [15] построения синтетических объектов с использованием метода k ближайших соседей. При восстановлении пропущенных значений, используем метод цепных уравнений MICE или одномерный метод восстановления пропущенных значений средним.
5. **Метод mixup** [14] аугментации. Заключается в создании новых синтетических объектов выпуклой комбинацией пар уже имеющихся объектов внутри каждого из классов.

Для каждого из методов аугментации выборки, помимо аугментации, также производится балансировка выборки, в результате которой размеры всех классов после аугментации имеют одинаковый размер.

На рис. 3., визуализированы различные методы аугментации данных полностью заполненной выборке MNIST, содержащей нарисованные от руки изображения цифр.

5. Описание выборок

Опишем 7 выборок, которые будем использовать для сравнения различных методов аугментации. 6 выборок из 7 являются медицинскими выборками. На практике, данный тип выборок часто содержит некомплектные объекты. Каждая из 7 выборок содержит хотя бы один категориальный признак, т.к. целевой класс, к которому принадлежит объект, является категориальным.

1. **Prodromal PD.** Выборка [4] для прогноза болезни Паркинсона на продромальной стадии по биомаркерам в гуморальных средах. Объект в выборке – пациент, признаки – его параметры крови. Доля пропущенных значений – 23.7%. Выборка подвергалась предварительной обработке – удалению тех признаков, которые заполнены менее, чем у половины объектов.

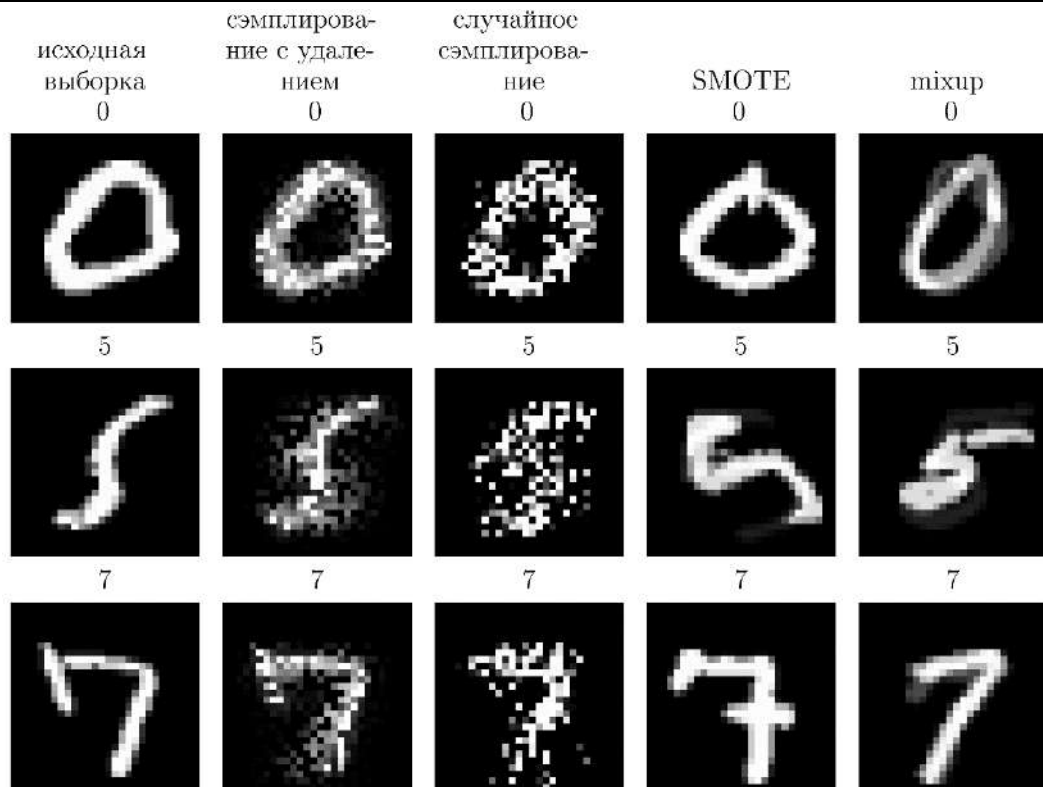


Рис. 3. Визуализация различных методов аугментации данных с использованием выборки MNIST.
Fig. 3. Augmentation method comparison on MNIST dataset.

2. **Acute Leukemia.** В выборке [5] собраны почти 7000 генов у 72 пациентов. Пациенты делятся на две группы: пациенты с острым лимфобластным лейкозом и острым миелобластным лейкозом. Выборка включает в себя обучающую выборку размером 38 и тестовую размером 34. Тестовая и обучающая выборка получены в разных лабораториях, исследования проведены по различным гуморальным средам. Также, тестовая выборка включала пациентов детского возраста, а обучающая не включала. Поэтому используем только обучающую выборку, из которой оставим только 50 наиболее значимых генов согласно t-тесту Уэлча.
3. **PPMI.** Инициативой PPMI поиска маркеров прогрессирования БП [3], была собрана выборка, содержащая более 300 здоровых пациентов из контрольной группы и более 3300 пациентов с диагнозом БП. Выборка содержит демографические данные, снимки ПЭТ-томографии, параметры крови, биомаркеры в спинномозговой жидкости и другие признаки.
4. **MNIST.** Выборка MNIST [6] содержит нормированные рукописные изображения цифр от 0 до 9 размером 28×28 пикселей. Таким образом, данные содержат 784 признаков. В обучающей выборке содержится 60000 изображений, размер тестовой выборки – 10000. Изображения в выборке MNIST центрированы, т.е. центр масс каждого изображения совпадает с центром изображения.
5. **EyeState.** В выборке [8] значения признаков получены с использованием ЭЭГ. Объекты принадлежат двум классам: для первого закрыты глаза при измерении ЭЭГ, для второго – открыты.

- 6. **Breast Cancer Wisconsin.** Выборка [9] содержит цифровизированные значения, полученные из снимков тонкоигольной аспирационной биопсии опухоли молочной железы. Выборка содержит доброкачественные и злокачественные опухоли.
- 7. **Cardiotocography.** Выборка [7] содержит измерения кардиотокограммы плодов, разделенные на 3 класса опытными акушерами-гинекологами: нормальный результат, диагноз под вопросом и наличие патологии.

Табл. 1. Описание использованных в настоящей работе выборок.

Table 1. Sample description.

Название выборки	Число объектов в выборке	Доля пропущенных значений	Число признаков	Число категориальных признаков	Число классов	Доля максимального класса
Prodromal PD	45	23.7%	11	1	2	57.8%
Acute Leukemia	38	0%	51	1	2	71.1%
PPMI	3625	3.8%	35	5	2	91.7%
MNIST	70000	0%	785	1	10	10%
EyeState	14980	0%	15	1	2	55.1%
Breast Cancer Wisconsin	569	0%	31	1	2	62.7%
Cardiotocography	2126	0%	42	12	3	77.8%

Первые 2 выборки размером менее 50, будем называть малыми, MNIST и EyeState, содержащие более 10000 объектов назовем большими, а оставшиеся 3 выборки размером от 500 до 4000, назовем средними.

Заметим, что из 7 выборок только 2 содержат пропущенные значения и являются некомплектными. Выборки PPMI и prodromal PD имеют тип MAR, т.к. в контрольной группе у них больше доля некомплектных значений, чем в группе риска. 5 полностью заполненных выборок, за исключением MNIST, будем преобразовывать в некомплектные выборки типа MCAR удалением части значений признаков из выборки случайным образом.

В 5 из 7 выборок, задача классификации бинарная; в выборке “Cardiotocography”, количество классов равняется 3, выборка MNIST содержит 10 классов изображений. Все выборки, за исключением prodromal PD и PPMI, находятся в открытом доступе.

6. Результаты вычислительных экспериментов

В настоящем разделе приведем результаты следующих вычислительных экспериментов:

- Сравним точность классификации в случае восстановления пропущенных значений и без него.
- Выясним, насколько аугментация позволяет увеличить точность классификации, сравним разные методы аугментации выборки и найдем наиболее эффективные.

Выберем критерии сравнения точности классификации. Пусть для двух сравниваемых измерений получены оценки α_1 и α_2 сбалансированной точности. Во-первых, будем использовать увеличение сбалансированной точности $\alpha_2 - \alpha_1$. Однако, одно изменение сбалансированной точности не позволяет объективно оценить точность прогноза, т.к. увеличение точности на одинаковую величину при малом α_1 (при качестве прогноза, близкому к случайному угадыванию), и при $\alpha_1 \approx 1$ (близкой к оптимальной точности прогноза), дает разное увеличение точности прогноза. Поэтому будем использовать также приближение прогноза к оптимуму $(\alpha_2 - \alpha_1)/(1 - \alpha_1)$.

Будем говорить, что точность прогноза незначительно улучшилось, или ухудшилось, если приближение к оптимуму $|(\alpha_2 - \alpha_1)/(1 - \alpha_1)| < 0.1$. При приближении к оптимуму из полуинтервала $[0.1, 0.2)$, говорим, что качество прогноза умеренно увеличилось. А при приближении к оптимуму в 0.2 раза и более, говорим, что имеет место существенное увеличение точности.

6.1 Восстановление пропущенных значений и точность классификации

Сравним точность классификации при восстановлении пропущенных значений методом MICE и при обучении модели машинного обучения на некомплектной выборке. Для сравнения используем выборку breast cancer Wisconsin, из которой удалены случайным образом 40% значений. Обучим на ней ML-модель категориального бустинга CatBoost [25]. Для оценки точности прогноза здесь и далее, используем метрику сбалансированная точность, оцененную с помощью вложенной кросс-валидации.

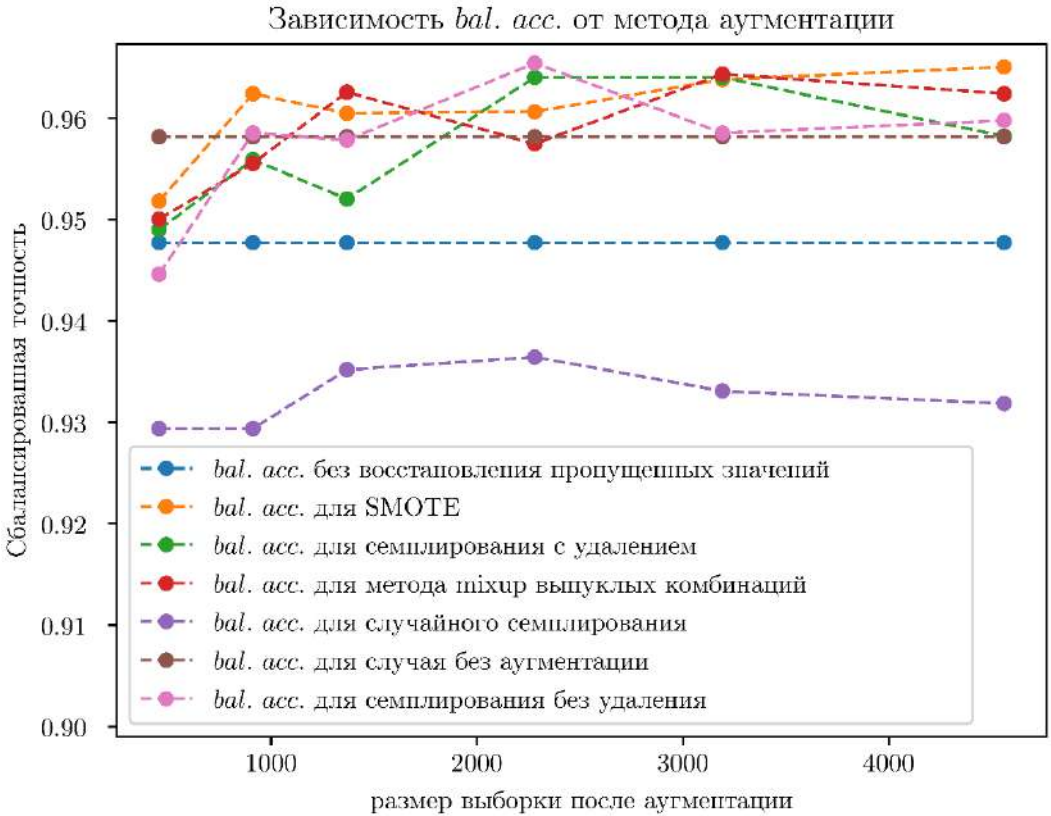


Рис. 4. Сравнение точности классификации при восстановлении пропущенных значений и при обучении ML-модели CatBoost на некомплектной выборке, а также при аугментации выборки.

Fig. 4. Accuracy comparison when learning on missing data and when using data imputation.

Сравним сбалансированную точность в случае восстановления пропущенных значений (коричневые точки) и в случае обучения на исходной некомплектной выборке (синие точки). В рамках использования схемы вложенной кросс-валидации, обучаем 20 раз на выборке, содержащей 512 объектов, строим прогноз на оставшихся 57 значениях, результаты которого сравниваем с известными ответами для получения сбалансированной точности.

- Некомплектная выборка. Сбалансированная точность: 0.948.

- Выборка, значения которой восстановлены с помощью метода MICE цепных уравнений. Сбалансированная точность: 0.959.
- Аугментация SMOTE. Сбалансированная точность изменяется от 0.96 до 0.965 в зависимости от размера выборки после аугментации.

Таким образом, при восстановлении пропущенных значений, сбалансированная точность прогноза увеличивается на 1.1%. При этом, приближаемся к оптимуму на 22%. Таким образом, восстановление пропущенных значений позволяет существенно увеличить точность прогноза. В дополнение этого, при использовании SMOTE для аугментации и балансировки выборки, получим увеличение сбалансированной точности от 0.1% до 0.6%. Это позволяет умеренно увеличить точность прогноза.

6.2 Аугментация выборки и точность классификации

Сравним влияние метода аугментации выборки на точность прогноза. В приведенной ниже таблице 2, сравним различные методы аугментации выборки между собой и со случаем отсутствия аугментации. Так как при аугментации также проводим балансировку выборки, то говорить об увеличении точности классификации из-за аугментации будем только тогда, когда после аугментации увеличивается точность как по сравнению со случаем отсутствия аугментации, так и со случаем балансировки сэмплированием без удаления.

Заметим, что предложенный метод аугментации сэмплированием с удалением позволяет увеличить сбалансированную точность и коэффициент корреляции Мэтьюса прогноза для выборок малого размера: prodromal PD и acute leukemia. При этом, обучается ML-модель опорных векторов SVM для классификации. Для выборок большего размера, наиболее эффективны методы аугментации SMOTE и mixup. При этом, обучаются ML-модели градиентного и категориального бустинга.

Для полностью заполненной выборки MNIST, аугментация данных не позволяет увеличить точность прогноза. Для несбалансированной выборки PPMI, наибольшее увеличение точности достигается из-за балансировки выборки, а не из-за ее аугментации.

Также отметим, что для выборок EyeState, breast cancer Wisconsin и cardiocography, наибольшее относительное увеличение точности прогноза из-за аугментации достигается при обучении на некомплектной выборке. Так, для разреженной выборки EyeState, аугментация позволяет умеренно увеличить точность прогноза, а для полностью заполненной выборки, изменение точности прогноза незначительное. Для разреженной выборки Cardiotocography, аугментация позволяет существенно увеличить точность прогноза, а для полностью заполненной выборки, увеличение точности прогноза умеренное.

7. Заключение

Задача обучения на некомплектной выборке сформулирована в рамках теории ВПК-обучения. Исследована ВПК-обучаемость задачи обучения на некомплектной выборке. Был оценен рост VC-размерности выборки при восстановлении пропущенных значений фиксированным значением из конечного ограниченного множества. Доказано, что требуемый размер выборки для ВПК-обучения с фиксированной точностью, растет логарифмически медленно с ростом размера этого множества. Было доказано, что при заполнении пропущенных значений элементами сетки, требуемый размер выборки $m(\epsilon, \delta)$ для обучения с заданной точностью ϵ с вероятностью не менее $1 - \delta$, увеличится незначительно по сравнению со случаем полностью заполненной выборки.

Для разреженной выборки breast cancer Wisconsin, сравнивалась точность классификации при восстановлении пропущенных значений методом MICE цепных уравнений и при обучении модели машинного обучения CatBoost на некомплектной выборке. Восстановление пропущенных значений позволило существенно увеличить точность прогноза.

Таблица 2. Сравнение методов аугментации выборки со случаем отсутствия аугментации.
Table 2. Comparison of augmentation methods with the case of no augmentation.

Название выборки	Доля пропущенных значений	ML-модель	Метод восстановления пропущенных значений	Сбалансированная точность без аугментации	Лучшая сбалансированная точность с аугментацией	Лучший метод аугментации	Приближение точности к оптимальной
Prodromal PD	23.7%	SVM	Восстановление пропущенных значений средним	0.765	0.825	сэмплирование с удалением	25.6%
Acute Leukemia	0%	SVM	Восстановление пропущенных значений средним	0.861	0.933	случайное сэмплирование	52.1%
Acute Leukemia	40%	SVM	Восстановление пропущенных значений средним	0.825	0.875	сэмплирование с удалением	28.7%
PPMI	3.8%	gradient boosting	MICE	0.661	0.667	сэмплирование без удаления	1.5%
MNIST	0%	CatBoost	Восстановление пропущенных значений средним	0.925	0.926	сэмплирование без удаления	1.3%
EyeState	0%	gradient boosting	MICE	0.922	0.924	SMOTE	2.9%
EyeState	40%	gradient boosting	MICE	0.879	0.891	SMOTE	10.1%
Breast Cancer Wisconsin	0%	gradient boosting	MICE	0.944	0.961	mixup	30.1%
Breast Cancer Wisconsin	30%	gradient boosting	MICE	0.926	0.954	SMOTE	38.2%
Cardiotocography	0%	gradient boosting	MICE	0.978	0.982	SMOTE	18.9%
Cardiotocography	40%	gradient boosting	MICE	0.944	0.966	mixup	39.5%

Аугментация и балансировка позволяют увеличить точность прогноза для всех 4 рассмотренных некомплектных средних и больших выборок, содержащих более 500 объектов. Тогда как только для 2 из 4 полностью заполненных выборок размера более 500: breast cancer Wisconsin и cardiotocography, аугментация позволяет увеличить точность классификации. Наибольшее увеличение точности аугментация позволяет достигнуть для некомплектной выборки. Отсюда можно сделать вывод о том, что аугментацию целесообразно использовать при обучении на некомплектной выборке.

Список литературы / References

- [1]. Little, Roderick JA, and Donald B. Rubin. Statistical analysis with missing data. John Wiley & Sons, 2002. 389 p.
- [2]. Thomas, Rajat M., et al. Dealing with missing data, small sample sizes, and heterogeneity in machine learning studies of brain disorders. Machine learning, Academic Press, 2020, pp. 249-266. DOI: 10.1016/B978-0-12-815739-8.00014-6.

- [3]. Marek, Kenneth, et al. "The Parkinson progression marker initiative (PPMI). *Progress in neurobiology*, 95.4, 2011, pp. 629-635. DOI: 10.1016/j.pneurobio.2011.09.005.
- [4]. Katunina, Elena A., et al. Searching for biomarkers in the blood of patients at risk of developing Parkinson's disease at the Prodromal Stage. *International Journal of Molecular Sciences*, 24.3, 2023, pp. 1842-1860. DOI: 10.3390/ijms24031842.
- [5]. Golub, Todd R., et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286.5439, 1999, pp. 531-537. DOI: 10.1126/science.286.5439.53.
- [6]. Deng, Li. "The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29.6, 2012, pp. 141-142. DOI: 10.1109/MSP.2012.2211477.
- [7]. Campos, D. & Bernardes, J. Cardiotocography [Dataset]. UCI Machine Learning Repository, 2000. DOI: 10.24432/C51S4N.
- [8]. Roesler, O. EEG Eye State [Dataset]. UCI Machine Learning Repository, 2013. DOI: 10.24432/C57G7J.
- [9]. Wolberg William, H., W. N. Street, and O. L. Mangasarian. Breast cancer wisconsin (diagnostic) data set, 1995. DOI: 10.24432/C5DW2B.
- [10]. Liu, Tongyu, et al. Adaptive data augmentation for supervised learning over missing data. *Proceedings of the VLDB Endowment*, 14.7, 2021, pp. 1202-1214.
- [11]. Han, Dongmei, Qigang Liu, and Weiguo Fan. A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications*, 95, 2018, pp. 43-56. DOI: 10.1016/j.eswa.2017.11.028.
- [12]. Nanni, Loris, Gianluca Maguolo, and Michelangelo Paci. Data augmentation approaches for improving animal audio classification. *Ecological Informatics*, 57, 2020, pp. 1-26. DOI: 10.1016/j.ecoinf.2020.101084.
- [13]. Zhou, Yue, et al. A survey on data augmentation in large model era. *arXiv preprint*, 2024, pp. 1-33. DOI: 10.48550/arXiv.2401.15422.
- [14]. Zhang, Hongyi. Mixup: Beyond empirical risk minimization. *arXiv preprint*, 2017, pp. 1-13. DOI: 10.48550/arXiv.1710.09412.
- [15]. Chawla, Nitesh V., et al. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, 2002, pp. 321-357. DOI: 10.1613/jair.953.
- [16]. Valiant, Leslie G. A theory of the learnable. *Communications of the ACM*, 27.11, 1984, pp. 1134-1142.
- [17]. Vapnik, Vladimir N., and A. Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Measures of complexity: festschrift for Alexey Chervonenkis*. Cham: Springer International Publishing, 2015, pp. 11-30.
- [18]. Michael, Loizos. Partial observability and learnability. *Artificial Intelligence*, 174.11, 2010, pp. 639-669.
- [19]. Blumer, Anselm, et al. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36.4, 1989, pp. 929-965.
- [20]. Yoon, Jinsung, et al. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33, 2020, pp. 1-11.
- [21]. Gondara, Lovedeep, and Ke Wang. Mida: Multiple imputation using denoising autoencoders. *Pacific-Asia conference on knowledge discovery and data mining*. Springer International Publishing, 2018, pp. 260-272. DOI: 10.1007/978-3-319-93040-4_21.
- [22]. Nelwamondo, Fulufhelo V., Shakir Mohamed, and Tshilidzi Marwala. Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, 2007, pp. 1514-1521. DOI: 10.48550/arXiv.0704.3474.
- [23]. Rubin, Donald B. Multiple imputation. *Flexible imputation of missing data*, second edition. Chapman and Hall/CRC, 2018, pp. 29-62.
- [24]. Campagner, Andrea. Missing but not Missed: On Learnability Under Imputation. *Preprint*, 2025, 1-18. DOI: 10.1007/978-3-032-06078-5_20.
- [25]. Prokhorenkova, Liudmila, et al. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018, pp. 1-11.
- [26]. Troyanskaya, Olga, et al. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17.6, 2001, pp. 520-525. DOI: 10.1016/B978-0-12-815739-8.00014-6.
- [27]. Li, Dan, et al. Towards missing data imputation: a study of fuzzy k-means clustering method. *Springer Berlin Heidelberg*, 2004, pp. 1-5. DOI: 10.1007/978-3-540-25929-9_70.

- [28]. Oba, Shigeyuki, et al. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics* 19.16, 2003, pp. 2088-2096.
- [29]. Schmitt, Peter, Jonas Mandel, and Mickael Guedj. A comparison of six methods for missing data imputation. *Journal of biometrics & biostatistics*, 6.1, 2015, pp. 1-7. DOI: 10.17485/ijst/2017/v10i19/110646.
- [30]. Pereira, Ricardo Cardoso, Pedro Henriques Abreu, and Pedro Pereira Rodrigues. Vae-bridge: Variational autoencoder filter for bayesian ridge imputation of missing data. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020. pp. 1-14.
- [31]. Pereira, Ricardo Cardoso, et al. Imputation of data Missing Not at Random: Artificial generation and benchmark analysis. *Expert Systems with Applications*, 249, 2024, pp. 1-14. DOI: 10.1016/j.eswa.2024.123654.
- [32]. Choudhury, Arkopal, and Michael R. Kosorok. Missing data imputation for classification problems. *arXiv preprint*, 2020, pp. 1-27. DOI: 10.48550/arXiv.2002.10709.
- [33]. Stekhoven, Daniel J., and Peter Bühlmann. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28.1, 2012, pp. 112-118. DOI: 10.1093/bioinformatics/btr597.
- [34]. Matthews, Brian W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405.2, 1975, pp. 442-451.
- [35]. Cawley, Gavin C., and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11, 2010, pp. 2079-2107.
- [36]. Berrar, Daniel. Cross-validation. 2018, pp. 542-554. DOI: 10.1016/B978-0-12-809633-8.20349-X.
- [37]. Stone, Mervyn. "Cross-validated choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36.2. 1974, pp. 111-133. DOI: 10.1111/j.2517-6161.1974.tb00994.x.
- [38]. Vabalas, Andrius, et al. Machine learning algorithm validation with a limited sample size. *PloS one* 14.11, 2019, pp. 1-20. DOI: 10.1371/journal.pone.0224365.
- [39]. Bishop, Chris M. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7.1, 1995, pp. 108-116. DOI: 10.1162/neco.1995.7.1.108.
- [40]. Вапник, В.Н., and. Червоненкис А.Ю. Теория распознавания образов: статистические проблемы обучения. Наука, 1974, 416 p. (in Russian).
- [41]. Haussler, David. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and computation*, 100.1, 1992, pp. 78-150.
- [42]. Sauer, Norbert. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13.1, 1972, pp. 145-147. DOI: 10.1016/0097-3165(72)90019-2.

Информация об авторах / Information about authors

Денис Олегович ЛАЗАРЕВ является специалистом кафедры теоретической информатики Института системного программирования им. В.П. Иванникова РАН. Научные интересы включают машинное обучение, вероятностный метод и алгоритмы упаковки.

Denis Olegovich LAZAREV – specialist of the Department of Theoretical Computer Science of Ivannikov Institute for System Programming of the RAS. Research interests include machine learning, probabilistic methods and packing algorithms.

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.), Prof., 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.

Станислав Александрович ФОМИН – ведущий программист. Область научных интересов: теория сложности, алгоритмы дискретной оптимизации, верификация ПО, архитектура информационных систем.

Stanislav Alexandrovich FOMIN – leading programmer. Research interests: complexity theory, program verification, discrete optimization algorithms, information systems architecture.

Приложение 1. Элементы теории ВПК-обучения

Приведем, следуя работе [17] Вапника и Червоненкиса, определение агностически ВПК-обучаемого множества гипотез $\mathcal{H}$. Пусть X – пространство признаков, а Y – пространство ответов. Предположим, что объекты выборки (x_i, y_i) , независимы в совокупности и распределены согласно вероятностному распределению D для всех i . Пусть $l: Y \times Y \rightarrow \mathbb{R}$ – функция, с помощью которой строим функцию потерь. Наиболее распространенным вариантом l , соответствующим функции потерь *accirasy*, является (характеристическая) функция, принимающая на вход две переменных, и равная 1 в случае их равенства и 0 иначе. Определим функцию потерь L_D для гипотезы $h \in \mathcal{H}$ следующим образом: $L_D(h) = E_{(x,y) \sim D}(l(h(x), y))$. Определим функцию потерь по множеству гипотез $L_D(\mathcal{H})$ как $\inf \{L_D(h): h \in \mathcal{H}\}$. Данную величину будем называть ошибкой аппроксимации множества гипотез $\mathcal{H}$ по распределению D .

Определение 1 (Агностическое ВПК обучение, Вапник и Червоненкис [17, 40]; Хаусслер [41]). Пусть X и Y – пространства признаков и ответов соответственно. Пусть $\mathcal{H}: X \rightarrow Y$ – множество гипотез. Тогда называем алгоритм A агностическим ВПК-обучателем для множества $\mathcal{H}$ тогда и только тогда, когда существует функция сложности $m: (0,1)^2 \rightarrow \mathbb{Z}^+$, такая, что для каждой пары параметров $\epsilon, \delta \in (0,1)$, для каждого распределения D над $X \times Y$, существует $m(\epsilon, \delta) \in \mathbb{Z}^+$, такое, что для выборки $S_m = \{(x_i, y_i)_{i=1}^m\}: (x_i, y_i) \sim D \quad \forall 1 \leq i \leq m$ размера m , состоящей из m независимых в совокупности объектов x_i и ответов y_i , случайно выбранных согласно распределению D , алгоритм A , получив выборку S_m , с вероятностью не менее $1 - \delta$, возвращает такую гипотезу $h \in \mathcal{H}$, для которой выполняется:

$$L_D(h) \leq L_D(\mathcal{H}) + \epsilon.$$

Задача ВПК обучения является частным случаем агностического ВПК обучения, в котором существует оптимальная гипотеза $h \in \mathcal{H}$, для которой $h(x_i) = y_i \quad \forall 1 \leq i \leq m$.

Если для множества гипотез $\mathcal{H}$ существует агностический ВПК-обучатель, то множество гипотез $\mathcal{H}$ называем агностически ВПК-обучаемым.

Определение агностической ВПК-обучаемости из работы [41] требует также полиномиальности вычислительной сложности алгоритма A от ϵ и $\log(1/\delta)$. Однако данный аспект мы не будем рассматривать в настоящей работе.

Дадим определение размерности Вапника-Червоненкиса, или VC-размерности для класса функций, тесно связанное с ВПК-обучаемостью.

Определение 2 (Разбиение множества классом функций). Пусть $\mathcal{H}$ – класс характеристических функций $h: X \rightarrow \{0,1\}$. Класс функций $\mathcal{H}$ разбивает множество $S \subseteq X$, если для каждого подмножества $S' \subseteq S$, существует такая функция $h' \in \mathcal{H}$, которая принимает единичные значения на S' и нулевые на $S \setminus S'$.

Определение 3 (VC-размерность). Пусть $\mathcal{H}$ – класс функций $h: X \rightarrow \{0,1\}$. VC-размерность $\mathcal{H}$ – наибольший размер подмножества $S \subseteq X$, который разбивается классом $\mathcal{H}$. Обозначаем VC-размерность класса $\mathcal{H}$ как $VCdim(\mathcal{H})$.

В [42] была доказана фундаментальная теорема ВПК обучения, утверждающая, в частности, что множество гипотез $\mathcal{H}$, отображающих множество X в $\{0,1\}$, агностически ВПК-обучаемо тогда и только тогда, когда $VCdim(\mathcal{H}) < +\infty$.

Определение 4 (Функция роста). $\tau_{\mathcal{H}}: \mathbb{N} \rightarrow \mathbb{N}$ – функция роста для множества гипотез $\mathcal{H}$, содержащего характеристические функции $h: X \rightarrow \{0,1\}$, если и только если $\tau_{\mathcal{H}}(m)$ – максимальное по всем множествам $A \subseteq X$ размера m число способов, которыми множество гипотез $\mathcal{H}$ может разделить A на 2 подмножества (одно A_0^h – переводится функцией $h \in \mathcal{H}$ в 0, A_1^h переводится в 1).

В частности, при $m \leq VCdim(\mathcal{H})$, функция роста составляет $\tau_{\mathcal{H}}(m) = 2^m$.

При доказательстве фундаментальной теоремы ВПК обучения, а также при доказательстве полученных в настоящей работе результатов, используется следующая лемма [42]:

Лемма Зауэра-Шелаха. Пусть $\mathcal{H}$ – множество гипотез $h: X \rightarrow \{0,1\}$. Пусть $VCDim(\mathcal{H}) < +\infty$. Тогда, выполняется следующее ограничение сверху на функцию роста $\tau_{\mathcal{H}}(m)$ для класса $\mathcal{H}$:

$$\tau_{\mathcal{H}}(m) \leq \sum_{i=0}^{VCDim(\mathcal{H})} \binom{m}{i} \leq \left(\frac{em}{VCDim(\mathcal{H})}\right)^{VCDim(\mathcal{H})} \quad \forall m > VCDim(\mathcal{H}).$$

Приложение 2. Доказательство леммы 1

Утверждение 1 леммы очевидно ввиду детерминированного заполнения пропущенных значений признаков единственным значением соответствующего признака точки X_{fill} .

Докажем утверждение 2 леммы. Рассмотрим функцию роста $\tau_{\mathcal{H}'}$ для множества $\mathcal{H}' = \mathcal{H}'(\mathcal{H}, CFill_{const})$. Пусть $\mathcal{H}_j' = \mathcal{H}'(\mathcal{H}, CFill_j)$ – т.е. множество гипотез $\mathcal{H}$, определенных на некомплектной выборке после ее заполнения единственным значением $x_{fill}^j \in X_{fill}$. Тогда $\mathcal{H}' = \bigcup_{j=1}^k \mathcal{H}_j'$, стало быть, выполняется соотношение $\tau_{\mathcal{H}'}(m) \leq \sum_{j=1}^k \tau_{\mathcal{H}_j'}(m) \quad \forall m \in \mathbb{N}$. Рассмотрим $m' = 2d \log_2(k^{1/d} + 31)$, где $d = VCDim(\mathcal{H})$. Т.к. $m' > VCDim(\mathcal{H})$, то, по лемме Зауэра-Шелаха, выполняется, $\tau_{\mathcal{H}_j'}(m') \leq \left(\frac{em'}{d}\right)^d$. Таким образом, $\tau_{\mathcal{H}'}(m') \leq k\left(\frac{em'}{d}\right)^d$. Докажем, что $\tau_{\mathcal{H}'}(m') < 2^{m'}$. Отсюда будет следовать требуемое утверждение о том, что $VCDim(\mathcal{H}'(\mathcal{H}, CFill_{const})) \leq m'$.

$\frac{\tau_{\mathcal{H}'}(m')}{2^{m'}} \leq \frac{k\left(\frac{em'}{d}\right)^d}{2^{m'}}$. Таким образом, для получения требуемой оценки сверху на $VCDim(\mathcal{H}'(\mathcal{H}, CFill_{const}))$, достаточно доказать, что

$k\left(\frac{em'}{d}\right)^d < 2^{m'}$ для выбранных значений m' и d .

$k\left(\frac{em'}{d}\right)^d < 2^{m'} \Leftrightarrow k^{1/d} \frac{em'}{d} < 2^{m'/d}$. Подставив $\frac{m'}{d} = 2 \log_2(k^{1/d} + 31)$, получим $k^{1/d} 2e \log_2(k^{1/d} + 31) < 2^{2 \log_2(k^{1/d} + 31)} = (k^{1/d} + 31)^2$.

Последнее неравенство верно как следствие двух неравенств:

1. $k^{1/d} < k^{1/d} + 31$
2. $2e \log_2 x - x < 0$ при $x \geq 32$, где за x обозначили $k^{1/d} + 31$. Для доказательства неравенства заметим, во-первых, его истинность при $x = 32$, и, во-вторых, что производная непрерывно дифференцируемой при $x > 0$ функции в левой части составляет $(2e \log_2 x - x)'_x = \frac{2e}{x \ln 2} - 1 < 0$ при $x > 8 > \frac{2e}{\ln 2}$ $\square$

DOI: 10.15514/ISPRAS-2026-38(1)-9



Оценка паразитной ёмкости цепей и быстродействия блока СБИС с помощью машинного обучения на этапе размещения

Н.В. Желудков, ORCID: 0009-0009-0501-5686 <nvigel@cs.niisi.ras.ru>

Ю.А. Кулемина, ORCID: 0009-0008-5774-8921 <kulemina@cs.niisi.ras.ru>

*НИЦ «Курчатовский институт» – НИИСИ,
Россия, 117218, г. Москва, Нахимовский просп., 36, к. 1.*

Аннотация. В работе рассматривается решение задачи предсказания паразитной ёмкости цепей при проектировании цифровых блоков СБИС до этапа трассировки на основе данных о размещении стандартных ячеек. Реальные значения паразитной ёмкости могут быть получены только после детальной трассировки и экстракции паразитных параметров в специализированном САПР, что занимает длительное время и не позволяет на раннем этапе проектирования точно оценить быстродействие блока. Классические методы оценки ёмкости не позволяют достичь баланса между точностью модели и времязатратами. В работе для решения этой задачи предложен ряд моделей машинного обучения (МО): многослойные линейные регрессионные модели с одним выходом и реализующие многозадачную регрессию, а также модели на основе градиентного бустинга над решающими деревьями – XGBoost и CatBoost. Для обучения и валидации моделей был собран набор данных, содержащий признаки и метки цепей, полученных из топологии 35 блоков по техпроцессу 40 нм. При обучении моделей использовалась кросс-валидация, метрики точности – MAE, MAPE и коэффициент детерминации R^2 . Наибольшую точность продемонстрировала модель XGBoost_M_A на основе градиентного бустинга в режиме мультизадачной регрессии – MAE 0,57 фФ, MAPE 10,3%, что на порядок ниже ошибки, полученной классическим методом на основе Wireload-модели. Среднее ускорение получения паразитных параметров по сравнению с эталонным методом после трассировки – 90 раз. Предсказанные ёмкости цепей были использованы для оценки финальной частоты блока на этапе размещения – точность оценки составила 99,1%, а среднее ускорение расчёта в 86 раз по сравнению с оценкой после проведения детальной трассировки. Предложенные методы на основе машинного обучения позволяют с высокой точностью и скоростью оценить на раннем этапе проектирования паразитные ёмкости цепей, а также оценить частоту работы блока СБИС.

Ключевые слова: сверхбольшие интегральные схемы СБИС; регистровая модель RTL; список ячеек и цепей электрической схемы нетлист; библиотека машинного обучения XGBoost; библиотека CatBoost.

Для цитирования: Желудков Н.В., Кулемина Ю.А. Оценка паразитной ёмкости цепей и быстродействия блока СБИС с помощью машинного обучения на этапе размещения. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 113–132. DOI: 10.15514/ISPRAS-2026-38(1)-9.

Благодарности: Работа выполнена в рамках государственного задания НИЦ "Курчатовский институт" – НИИСИ по теме FNEF-2024-0003.

Machine Learning-Based Estimation of Interconnect Capacitance and Block Performance in VLSI Design at Placement Stage

N.V. Zheludkov, ORCID: 0009-0009-0501-5686 <nvgel@cs.niisi.ras.ru>
Y.A. Kulemina, ORCID: 0009-0008-5774-8921 <kulemina@cs.niisi.ras.ru>

*NRC «Kurchatov Institute» – SRISA,
Building 1, 36, Nakhimovsky Prospekt, Moscow, 117218, Russia.*

Abstract. This work addresses the challenge of predicting interconnect parasitic capacitance in digital VLSI blocks during the pre-routing design stage using standard cell placement data. Accurate parasitic capacitance values are typically obtained only after detailed routing and parasitic extraction in specialized EDA tools, which is time-consuming and prevents accurate timing estimation at early design stages. Conventional capacitance estimation methods fail to achieve an optimal balance between accuracy and computational efficiency. To overcome these limitations, we propose a machine learning approach employing multiple architectures: multi-layer linear regression models supporting both single-output and multi-task regression, along with gradient boosting algorithms (XGBoost and CatBoost). For model training and validation, we compiled a dataset containing features and labels extracted from the topology of 35 blocks implemented in 40nm technology. Using cross-validation with MAE, MAPE, and R^2 metrics, the XGBoost_M_A model demonstrated superior performance with MAE of 0.57 fF and MAPE of 10.3%, representing an order of magnitude improvement over the traditional Wireload-model. The method achieves $90\times$ acceleration in parasitics extraction compared to reference approaches. When applied to frequency estimation at placement stage, it maintains 99.1% accuracy while providing $86\times$ speedup compared to detailed routing. The proposed machine learning methods enable fast and accurate estimation of interconnect parasitics and operating frequency at early design stages, significantly accelerating the VLSI design process.

Keywords: VLSI; RTL; netlist; XGBoost; CatBoost.

For citation: Zheludkov N.V., Kulemina Y.A. ML-models for Wire Capacitance Estimation in VLSI Design. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 113-132 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-9.

Acknowledgements. This research was performed as part of the State Research Task (Project FNEF-2024-0003) of the National Research Center «Kurchatov Institute» – SRISA.

1. Введение

В процессе проектирования цифровых блоков сверхбольших интегральных схем (СБИС) множество параметров и характеристик блока определяется паразитными ёмкостями внутренних элементов. Время переключения стандартных элементов, из которых состоит цифровой блок, определяется табличными значениями задержек в liberty-файлах (файлы для описания временных характеристик, электрических параметров, функционального поведения библиотек стандартных ячеек). Задержка элемента (а точнее конкретной временной арки) зависит от значения длительности фронта входного сигнала и суммарной ёмкости на выходном контакте. Чем больше фронт входного сигнала, тем дольше ячейка выдает новое состояние на выходном контакте. Такая же зависимость наблюдается и для ёмкости – чем больше выходная ёмкость (то есть выходная нагрузка), тем дольше элемент будет переключаться. Выходная ёмкость элемента (с одним выходом) определяется как сумма ёмкостей контактов, управляемых рассматриваемой ячейкой, и ёмкости межсоединений, соединяющий выход ячейки с ее нагрузками. Ёмкость контактов складывается из следующих компонент:

- ёмкость затворов транзисторов, к которым подключен контакт; зависит от площади затворов;
- ёмкость диффузионных областей – ёмкости p-n переходов сток-истоков транзисторов;

- ёмкость внутренних металлических соединений внутри самой ячейки (обычно, от входного контакта до затворов транзисторов).

Важно отметить, что ёмкость контакта конкретной ячейки в библиотеке элементов, спроектированных по одному технологическому процессу, является константой и определяется в liberty-файлах. Для современных технологических процессов (<90 нм) ёмкость контакта может составлять от единиц до десятков фемтофард. Ёмкость межсоединения (то есть ёмкость цепи, соединяющих ячейки) состоит из следующих компонент:

- ёмкость, образующаяся между металлическим проводником и кремниевой подложкой; характерна и превалирует для самых нижних слоев металлизации (первого и второго слоя металлизации);
- ёмкость, образующаяся между металлическими проводниками, находящимися в разных слоях;
- ёмкость, образующаяся между металлическими проводниками, находящимися в одном слое металлизации; определяется толщиной проводника, длиной общей параллельной части (чем дольше два проводника идут рядом, тем больше ёмкость между ними), а также расстоянием между проводниками – с уменьшением зазора ёмкость растёт.

Сокращение технологических норм увеличивает роль паразитных ёмкостей межсоединений в определении быстродействия цифровых блоков. В ряде случаев паразитная ёмкость межсоединений может быть сопоставима с ёмкостью контактов, или даже превосходить ее, что влечет за собой влияние на задержки и быстродействие проектируемого блока. Реальная ёмкость межсоединений может быть рассчитана только после проведения таких этапов топологического проектирования СБИС как размещение стандартных ячеек и трассировки межсоединений. После этапа топологического проектирования блока с разведенными цепями требуется запустить специализированную систему автоматизации проектирования (САПР) для расчёта паразитных параметров СБИС (кроме ёмкости такие САПР способны рассчитывать сопротивление и индуктивность). К таким САПР относятся:

- Quantus от компании Cadence (коммерческий САПР);
- StarRC от компании Synopsys (коммерческий САПР);
- OpenRCX [3] – САПР с открытым исходным кодом.

На основе данных о топологии блока (чаще всего передаваемых в САПР с помощью DEF-файла, содержащего данные о физическом размещении и трассировке элементов интегральной схемы) и о технологических файлах, содержащихся в PDK (набор технологических правил и моделей, определяющий, как элементы интегральной схемы должны быть спроектированы и проверены для конкретного полупроводникового процесса) для расчёта паразитных параметров (в случае выбора коммерческих маршрутов проектирования в качестве такого технологического файла используется qrcTechFile), указанные выше САПР рассчитывают паразитные параметры цепей в топологии блока и для дальнейшего их использования сохраняют данную информацию в SPEF-файл (файл с паразитными RC-параметрами межсоединений схемы), построенного на основе ставшего стандартом в индустрии формата данных для передачи и хранения паразитных ёмкостей и сопротивлений цепей в СБИС. После получения SPEF-файла можно точно рассчитать задержки ячеек в блоке с учетом паразитных параметров цепей, и, как следствие, рассчитать быстродействие блока. Отметим, что суммарная ёмкость на выходе ячейки влияет не только на быстродействие ячейки, но и на потребляемую мощность, данные для расчёта которой также содержатся в liberty-файлах, однако в рамках данной работы влияние паразитных параметров цепей на потребляемую мощность не будет рассматриваться.

Ввиду длительности прохождения этапов проектирования для реальной оценки паразитных ёмкостей (этап трассировки для блока с более чем миллионом стандартных ячеек может длиться больше суток) существует проблема в точной и быстрой оценке паразитных ёмкостей уже на этапе размещения стандартных ячеек. Так как одна итерация трассировки может занимать длительное время, то существует вероятность того, что нарушение временных характеристик блока (запас по времени удержания и времени установки) может быть выявлено только на более поздних этапах проектирования, по сравнению с этапом размещения стандартных ячеек, где нарушение выявлено не будет. Отсутствие точных методов оценки паразитных ёмкостей на этапе размещения приводит к существенным погрешностям в прогнозировании временных характеристик и необходимости множественных длительных и ресурсоемких итераций трассировки. В то же время ранняя оценка паразитных параметров и корректный расчёт быстродействия на их основе помогает инженеру заранее определять проблему в блоке и вносить изменения в процесс его проектирования, например, выбирать другой план размещения, применять другую расстановку макроблоков, изменять расположение портов ввода-вывода.

В рамках данной работы будет представлено:

- создание набора данных для обучения и валидации моделей оценки ёмкости, полученных в процессе проектирования 35 СФ-блоков СБИС;
- набор предложенных методов на основе машинного обучения, оценивающих паразитную ёмкость цепей на основе данных и признаков, доступных на этапе размещения, без выполнения этапа трассировки и использования специализированных САПР для генерации SPEF-файлов.
- сравнение предложенных методов по различным метрикам точности, а также сравнение с эталонными значениями;
- оценка точности прогнозирования финальной частоты работы блока на основе данных из лучшей предложенной модели.

2. Обзор связанных работ

К наиболее ранним и широко использовавшимся подходам для оценки параметров межсоединений до проведения детальной трассировки относятся методы, основанные на аналитических расчётах и эмпирических данных. Наиболее ярким примером в данной категории является Wireload-модель. Данная модель применима на этапе логического синтеза (без какой-либо топологии блока), а также на этапе размещения стандартных ячеек в процессе топологического проектирования. Основное предположение, которое заложено в данную модель, заключается в том, что средняя длина проводника, а, следовательно, и его паразитная ёмкость, сильно коррелирует с таким параметром цепи как разветвление на выходном контакте (fanout-разветвление), показывающим сколько контактов ячеек управляется сигналом с выходного контакта ячейки, соединенной одной цепью. Также этот параметр отражает количество нагрузок, подключенных к цепи. Данные для использования модели Wireload, обычно, содержатся в liberty-файлах используемых стандартных ячеек. Алгоритм расчёта паразитной ёмкости с использованием данной модели выглядит следующим образом:

- для рассматриваемой цепи в нетлисте (это представление цифровой схемы в виде списка логических элементов и их соединений) блока определяется параметр fanout;
- в соответствующем liberty-файле, исходя из площади блока, выбирается необходимая таблица с данными для использования модели Wireload (ввиду того что в данной модели заложено предположение, что средняя длина цепи в блоке большей площади будет больше, чем средняя длина цепи в блоке меньшей площади с таким же fanout);

- на основе табличных данных определяется длина цепи в зависимости от количества нагрузок, в случаях, когда оно выходит за табличные значения, проводится линейная экстраполяция от последней точки в табличных значениях;
- вычисленная длина цепи умножается на указанный в таблице коэффициент ёмкости, имеющий размерность отношения ёмкости к длине фФ/нм (10^{-15}Ф , 10^{-9}м); полученное значение считается паразитной ёмкостью цепи.

На рис. 1 изображен пример зависимости средней длины цепи от параметра разветвления при использовании модели Wireload из работы [1]. Несмотря на вычислительную эффективность и глубокую интеграцию в классические потоки проектирования, данная модель обладает рядом недостатков, особенно критичными для современных норм проектирования СБИС:

- отсутствие учета размещения ячеек – модель полностью игнорирует физическое расположение ячеек на кристалле; две цепи с одинаковым разветвлением могут значительно отличаться друг от друга по длине ввиду физических особенностей и ограничений, накладываемых их расположением в топологии;
- излишний пессимизм для цепей с большим разветвлением, выходящим за пределы табличных значений модели Wireload; это может привести к пессимистичным прогнозам временных характеристик и не оптимальным решениям, принимаемым САПР в процессе оптимизаций.

В связи с этими недостатками, использование Wireload-модели в настоящее время считается одним из основных препятствий для достижения сходимости результатов статического временного анализа на этапе размещения и на этапе трассировки. Несмотря на то, что эти модели представляют собой фундаментальный этап в развитии методов предварительной оценки паразитных параметров цепей, их неточность стала мотивацией для разработки более совершенных и физически обоснованных решений.

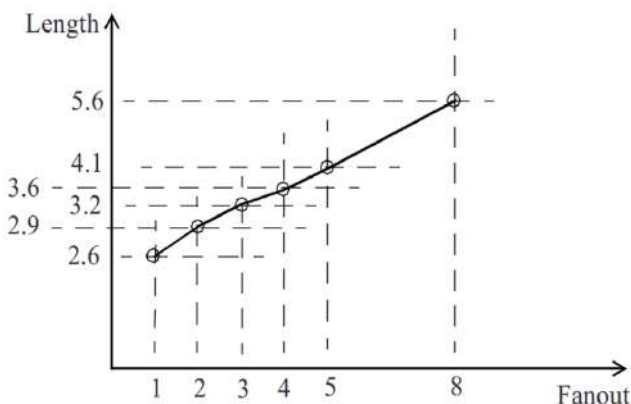


Рис. 1. Зависимость средней длины цепи от разветвления в модели Wireload.
Fig. 1. Average net length versus fanout in Wireload models.

В одной из ранних работ [2] авторы предлагают улучшение Wireload-модели для более точного расчёта потребляемой мощности на этапе размещения ячеек. В своей работе авторы использовали не названную субмикронную технологию с 6 доступными слоями металлизации для трассировки. Ключевая идея работы заключается в предположении, что цепи, обладающие большей длиной, располагаются в более высоких слоях металлизации, обладающих отличной от нижних слоев паразитной ёмкостью. Авторы предлагают следующий алгоритм для уточнения паразитной ёмкости на этапе размещения:

- оценить длины цепей с помощью метода HPWL (half-perimeter wirelength);

- сортировать цепи по длинам;
- рассчитать суммарную длину цепей;
- сформировать 3 группы цепей – короткие, средние и длинные;
- цепи из “длинной” группы относят к 5 и 6 слоям металлизации; цепи из “средней” группы – к 3 и 4 слоям металлизации; цепи из “короткой” группы – к 1 и 2 слоям металлизации;
- длина каждой цепи умножается на коэффициент ёмкости, характерной определенным слоям металлизации – таким образом вычисляется паразитная ёмкость для каждой цепи.

Предложенный метод позволил авторам снизить ошибку в оценке паразитной ёмкости по сравнению с оригинальной моделью Wireload в среднем на 2-13% для рассмотренных сложно-функциональных блоков (СФ-блоков), а также уменьшить ошибку расчёта мощности в среднем на 18,5% [2].

В последние годы методы машинного обучения активно исследуются как альтернатива традиционным подходам для устранения разрыва между предварительной и финальной оценкой параметров. Данные подходы демонстрируют потенциал в прогнозировании паразитных эффектов на основе данных, доступных на ранних этапах проектирования. Ярким примером специализированного подхода является работа [4], посвященная прогнозированию паразитных ёмкостей в SRAM-блоках (специализированные макроблоки статической памяти) на этапе схемотехнического проектирования. Авторы подчеркивают проблему значительных расхождений между результатами предварительного моделирования паразитной ёмкости и моделирования после этапа трассировки, что приводит к многочисленным итерациям при проектировании энергоэффективных SRAM. Для её решения предложена двухэтапная модель глубокого обучения, комбинирующая графовые нейронные сети и полносвязные сети. Ключевыми особенностями метода являются:

- использование графовой нейронной сети для формирования эмбедингов (векторного представления цепей) с эффективным учетом топологии схемы для подачи на полносвязную сеть для классификации цепей;
- применение Focal Loss (функция потерь, расширяющая кросс-энтропийные потери) для решения проблемы дисбаланса классов, так как в выборке присутствовало много цепей с паразитной ёмкостью менее 1 фФ.

Эксперименты на 4 реальных SRAM-блоках показали, что предложенный метод превосходит актуальную модель, обеспечивая снижение ошибки в 19 раз и ускорение процесса симуляции в 598 раз [4]. Также стоит отметить, что данная работа сфокусирована на оценке паразитных параметров блоков на транзисторном уровне и не рассматривает эту оценку для блоков, проектируемых на уровне абстракций, предоставляемых стандартными ячейками.

Ряд работ направлен на применение моделей машинного обучения (МО) не для предсказания паразитных параметров на этапе размещения, а для непосредственной оценки после этапа трассировки. В работе [5] представлена модель GNN- C_{eff} – одна из первых моделей на основе графовых нейронных сетей для прогнозирования паразитной ёмкости цепей на этапе трассировки. В работе проводится расчёт такого параметра как эффективная ёмкость – единая ёмкость, включающая в себя паразитную ёмкость цепи и ёмкость полезной нагрузки, используемая для расчёта задержек ячеек. К основным достижениям работы стоит отнести:

- высокую точность модели – средняя абсолютная ошибка составляет 0,01 фФ;
- значительное ускорение расчёта ёмкости за счет параллелизации вычислений на GPU – до 929 раз по сравнению с последовательным актуальным алгоритмом;

- модель валидирована для технологического процесса ASAP 7 в САПР OpenROAD.

Модель GNN- C_{eff} представляет собой значительный прорыв в скорости и точности расчёта ёмкости для цепей после трассировки. Однако её фундаментальное ограничение в контексте нашей задачи заключается в том, что она требует завершённой топологии в качестве входных данных. Таким образом, хотя метод и исключает необходимость в итеративных SPICE-симуляциях (моделирования электрических характеристик схем на транзисторном уровне) для расчёта эффективной ёмкости, он не решает проблему зависимости от самого трудоемкого этапа – трассировки.

Настоящее исследование направлено на решение этой проблемы на более раннем этапе. В то время как GNN- C_{eff} предсказывает эффективную ёмкость после трассировки, наша цель – предсказать физическую ёмкость до её начала, используя лишь данные, доступные на этапе размещения, что может потенциально сократить время на проектирование топологии блока.

3. Методология

В данном разделе подробно представлена информация о подготовке репрезентативного набора данных, выборе признаков для обучения, проектировании дополнительных признаков (feature engineering), анализе данных, представленных в наборе данных, а также описан процесс создания моделей для оценки паразитной ёмкости цепей и процесс их обучения.

3.1 Создание набора данных

Для создания набора данных были выбраны 35 открытых RTL-моделей (описание цифровой системы на уровне регистров, комбинаторной логики и управляющих сигналов) цифровых блоков СБИС, в состав которых не входили макроблоки (крупные функциональные блоки интегральной схемы с заранее определённой внутренней структурой) и блоки памяти. В качестве источника этих моделей были выбраны 3 ресурса:

- открытый набор данных OpenABC-D [6];
- проект OpenCores [7];
- проект NoC-Verrilog [8];

Блоки, выбранные из данных проектов, были синтезированы в САПР Cadence Genus для получения изначального нетлиста. Топологическое проектирование блоков проводилось в САПР Cadence Innovus. В качестве технологии был выбран процесс с технологической нормой 40 нм. Ниже приведены основные особенности проектирования блоков:

- для трассировки сигнальных цепей использовались 7 слоев металлизации;
- сетка земли-питания была построена с 1 по 7 металл в следующей конфигурации – 1 горизонтальный металл, 4 и 6 металл – вертикальные, 5 и 7 горизонтальные;
- быстродействие оценивалось в одном углу проектирования (фиксированная комбинация параметров процесса изготовления, напряжения питания и температуры, используемая для анализа работы интегральной схемы) – низкое быстродействие ячеек (“slow-slow” процесс), пониженное напряжение питания 0,99 В (при стандартном напряжении 1,1 В), повышенная температура 125 °C; в данном углу наибольшие задержки элементов, что позволяет оценить критическую частоту работы блока;
- расчёт паразитных параметров проводился в условиях “rcmax” – худшие паразитные значения ёмкости и сопротивления; расчёт паразитных параметров и выписка SPEF-файла производилась в САПР Cadence Quantus;
- план размещения блока имел соотношение сторон 1:1;
- начальная плотность ячеек для всех блоков составила 50%;

- порты ввода-вывода были расставлены по левой границе; в случае невозможности размещения всех портов слева – оставшаяся часть была расставлена справа;
- в технологии доступны две библиотеки стандартных ячеек в зависимости от высоты ячеек в треках – 9 и 12-и трековые ячейки; для создания набора данных были выбраны 12-и трековые ячейки, обладающие меньшей задержкой;
- в выбранной библиотеке стандартных ячеек были доступны ячейки с разным пороговым напряжением транзисторов – низким (lvt-ячейки, меньше задержка, но выше потребление), высоким (hvt-ячейки, больше задержка, но ниже потребление), и стандартным (svt-ячейки, средние задержка и потребление); в процессе оптимизации быстройдействия, площади и мощности в САПР использовались все 3 типа ячеек, тактовое дерево было построено на lvt-ячейках;
- согласно рекомендациям фабрики для приведенного выше угла проектирования использовался коэффициент, учитывающий разброс характеристик ячеек на кристалле (OCV, On Chip Variation) в 12%, который накладывался на задержки ячеек путей тактового сигнала до триггеров, захватывающих данные;
- значение джиттера (временная нестабильность фронтов сигнала) для тактового сигнала в процессе статического временного анализа равнялось 25 пс;
- после создания плана размещения и построения сетки земли-питания запускался автоматический маршрут, включающий в себя этапы размещения стандартных ячеек, построения дерева синхросигналов и трассировки. После этапа трассировки проводилась оценка паразитных параметров в Quantus и выписка признаков цепей для обучения моделей.

Перечисленные параметры блоков и настройки проектирования являются характерными при проектировании цифровых блоков для данной технологии и позволяют получить реалистичный набор данных. В табл. 1 представлен список из 35 блоков, для которых был пройден маршрут проектирования и собран набор данных.

3.2 Анализ данных и отбор признаков

Собранный набор данных представляет собой набор из 35 csv-файлов, содержащих признаки и метки цепей, для каждого из блоков. Всего в наборе данных содержится 564898 уникальных цепей. При создании набора данных не учитывались цепи, для которых САПР не смог провести экстракцию паразитов (вычисление паразитных RC-параметров схемы по её топологии) и для которых это значение было равно 0. В наборах данных представлены следующие метки цепей:

- “cap” – паразитная ёмкость цепи, полученная из SPEF-файла (ед. измерения фФ);
- “length” – длина цепи после детальной трассировки (ед. измерения мкм);
- “segs” – число сегментов, из которых состоит цепь (ед. измерения шт.).

Помимо основной метрики “cap” в наборе данных также собиралась информация о длине цепей и о числе сегментов (отдельных металлических проводников). Данные метки были собраны для проверки гипотезы, что совместное прогнозирование взаимосвязанных параметров может улучшить обобщающую способность модели и точность предсказания основной метки за счет выявления общих скрытых закономерностей в данных. Метрики “length” и “segs” были выбраны ввиду предположения, что паразитная ёмкость сильно зависит от длины цепи и числа ее сегментов. Распределение выбранных меток в наборе данных представлено в табл. 2 (единицы измерения соответствуют обозначенным выше в разделе; P25, P50, P75, P95 – соответствующие перцентили).

Табл. 1. Список блоков в наборе данных.

Table 1. List of blocks in dataset.

№	Название блока	Описание	Число ячеек	Число цепей
1	ac97_top	Блок контроллера Audio Codec '97	5996	3851
2	aes128_core	Блок аппаратной реализации алгоритма шифрования AES со 128 битным ключом	16736	14350
3	aes_cipher_top	Блок, реализующий полный конвейерный или последовательный алгоритм AES	9279	7907
4	aes_core	Блок ядра алгоритма шифрования	12058	9482
5	des3	Блок симметричного блочного шифра Triple DES	1517	1410
6	dft_top	Блок дискретного преобразования Фурье	10394	66076
7	dynamic_node	Блок динамически переконфигурируемого узла в сетевых системах	7303	5678
8	FIR_filter	Фильтр с конечной импульсной характеристикой	1282	1164
9	fpu	Блок арифметических операций над числами с плавающей точкой	13216	8831
10	jpeg_encoder	Блок для сжатия изображений по стандарту JPEG	30691	21221
11	mc_top	Блок контроллера статической памяти	3236	2376
12	mips_core	Блок RISC-ядра процессора архитектуры MIPS	6562	4977
13	mrisc_top	Реализация ядра RISC-V	863	560
14	Network	Блок накристалльной сети	29893	21578
15	pci_bridge32	Блок 32-битного моста между шиной PCI и другими устройствами	9143	6507
16	pcm_slv_top	Блок Pulse Code Modulation	214	149
17	picosoc	Блок системы на кристалле на основе RISC-V	32816	22444
18	plic_core	Блок контроллера прерываний в RISC-V системах	1999	1732
19	psram_core	Блок контроллера статической памяти	626	698
20	risc16f84_clk2x	8-битный микроконтроллер с ядром PIC-совместимой архитектуры	1076	729
21	RocketTile	Ядро RISC-V, сгенерированное с помощью генератора Chisel Rocket Chip	3475	2389
22	RsDecodeTop	Дешифратор Рида-Соломона	31883	19974
23	rs_encode	Реализация кодера Рида-Соломона	1830	1127
24	RsEncodeTop	Шифратор Рида-Соломона	977	613
25	sasc_top	Блок интегрированного контроллера для управления датчиками и исполнительными механизмами	334	260
26	sdram	Контроллер синхронной статической памяти	563	488
27	sha256	Блок криптографической хэш-функции, преобразующий входные данные произвольной длины в фиксированный 256-битный хеш	4246	3907
28	Sha3BlackBox	Блок хеширования данных с использованием алгоритма SHA-3	15765	11512
29	simple_spi_top	Блок SPI, реализованный в микроконтроллере Motorola MC68HC11E	419	285
30	spi_top	Блок контроллера интерфейса SPI	1161	1012
31	PipelinedCipher	Конвейерный криптографический сопроцессор	96872	78257
32	usb_phy	Блок физического уровня USB	274	192
33	vga_enh_top	Контроллер VGA-дисплея	47143	31564
34	wb_conmax_top	Блок многоуровневого арбитра и коммутатора для шины Wishbone	18628	16092
35	wb_dma_top	Блок контроллера прямого доступа к памяти с интерфейсом Wishbone	1618	1298

Анализ распределения паразитной ёмкости выявил выраженную положительную асимметрию. Медианное значение ёмкости составляет 1,0 фФ, что значительно ниже среднего значения 4,5 фФ. Распределение сильно сконцентрировано в области малых величин: 75% всех цепей имеют ёмкость не более 3,0 фФ, в то время как 95-й перцентиль достигает 21,0 фФ, а максимальное значение – 219,0 фФ. Стандартное отклонение (9,2 фФ), существенно превышающее среднее значение, дополнительно подтверждает высокую неоднородность данных. Также для цепей в наборе данных было рассчитано отношение паразитной ёмкости к ёмкости контактов в нагрузке – среднее значение этой характеристики оказалось равным 1,4 со стандартным отклонением 3,1, 75-перцентиль – 1, 95-перцентиль – 4. Из этого можно сделать вывод, что для блоков, спроектированных по выбранной технологии, значение паразитной ёмкости часто оказывается сопоставимо с полезной нагрузкой, поэтому точное предсказание паразитной составляющей актуально для данной технологической нормы. Корреляционный анализ по методу Пирсона показал, что коэффициент корреляции метрики “cap” и метрик “length” и “segs” составляет 0,98 и 0,72, соответственно, что свидетельствует о высоком уровне корреляции между выбранными метриками.

Табл. 2. Распределение меток в наборе данных.
Table 2. Labels distribution in dataset.

Метка	Среднее	Стандартное отклонение	Мин.	Макс.	P25	P50	P75	P95
“cap”	4,5	9,2	1	219	1	1	3	21
“length”	32,4	72,8	0,84	1812	4,6	7,4	20,9	156,1
“segs”	19,0	33,7	1	559	5	8	16	74

Каждая цепь в наборе данных обладает 26 признаками: 19 из этих признаков были получены из САПР, а еще 7 признаков были сконструированы вручную в процессе обработки данных. Список и описание признаков цепей приведено в табл. 3. Первоначальный набор признаков был сформирован на основе эмпирических знаний о физических факторах, влияющих на паразитную ёмкость цепей в цифровых СБИС. Данный набор основан на предположении, что выбранные признаки коррелируют с предсказываемой величиной.

К 7 признакам, созданным вручную из других имеющихся в наборе данных признаков в процессе обработки данных, относятся следующие:

- gcell_area – площадь прямоугольника, который образуется вокруг ячеек, соединенных рассматриваемой цепью. Полупериметр этого прямоугольника является метрикой HPWL. Данная метрика была создана из предположения, что площадь такого прямоугольника будет положительно коррелировать с паразитной ёмкостью цепи;
- gcell_ar – отношение ширины к высоте HPWL прямоугольника. Ожидается, что цепи, находящиеся в прямоугольниках вырожденной формы, будут отличаться своей длиной, а как следствие, ёмкостью от цепей в прямоугольниках близких к форме квадрата;
- dist_to_right, dist_to_left, dist_to_top, dist_to_bottom – данные признаки показывают, в какой части блока находится цепь. Ввиду того, что наибольшее число ячеек находится в центре блока, существует вероятность того, что САПР придется в местах повышенной плотности разводки переходить в верхние металлы, чтобы провести трассировку;
- block_area – площадь проектируемого блока. Ожидается, что две цепи с близкими признаками будут отличаться длиной в блоках разного размера – чем больше площадь блока, тем больше длина цепи в нем.

Табл. 3. Список признаков.

Table 3. Feature list.

№	Признак	Описание признака
1	fanin	Число входных контактов ячейки, управляющей цепью
2	fanout	Число нагрузок цепи
3	fanin_area	Площадь fanin ячеек
4	fanout_area	Площадь ячеек, соединенных с цепью
5	llx	Левая нижняя координата X HPWL прямоугольника
6	lly	Левая нижняя координата Y HPWL прямоугольника
7	urx	Правая верхняя координата X HPWL прямоугольника
8	ury	Правая верхняя координата Y HPWL прямоугольника
9	hpxl	Метрика “полупериметр” для цепи
10	hpxl_min	Мин. “полупериметр” для цепи
11	hpxl_max	Макс. “полупериметр” для цепи
12	hpxl_mean	Среднее значение “полупериметра” для цепи
13	hpxl_std	Стандартное отклонение “полупериметра” для цепи
14	cell_density	Плотность ячеек в прямоугольнике HPWL
15	pin_cap	Ёмкость контактов в нагрузке
16	wireload_fan_cap	Метрика Wireload на основе fanout
17	wireload_hpxl_cap	Метрика Wireload на основе HPWL
18	block_area	Площадь проектируемого блока
19	gcell_area	Площадь HPWL прямоугольника
20	gcell_ar	Соотношение сторон HPWL прямоугольника
21	dist_to_right	Расстояние от центра HPWL прямоугольника до правого края блока
22	dist_to_left	Расстояние от центра HPWL прямоугольника до левого края блока
23	dist_to_top	Расстояние от центра HPWL прямоугольника до верхнего края блока
24	dist_to_bottom	Расстояние от центра HPWL прямоугольника до нижнего края блока
25	block_width	Ширина блоков
26	block_height	Высота блока

Первоначально сформированный на физических принципах набор признаков может содержать избыточную информацию. Для построения компактной и эффективной модели был проведен статистический отбор признаков. В первую очередь был проведен корреляционный анализ – рассчитаны коэффициенты корреляции Пирсона признаков с целевой метрикой. После получения этих коэффициентов из набора признаков были убраны те, чей коэффициент корреляции составлял по модулю меньше 0,1. Признаками, слабо коррелирующими с целевой метрикой паразитной ёмкости, оказались: абсолютные координаты HPWL прямоугольника (llx, lly, urx, ury – коэффициенты 0,02, -0,0005, 0,08, 0,06, соответственно), размер блока (block_area, block_width, block_height – 0,07, 0,06, 0,06), относительное расположение центра HPWL прямоугольника (dist_to_right, dist_to_left, dist_to_top, dist_to_bottom – -0,08, 0,02, 0,01, 0,03). Корреляционная матрица для оставшихся в наборе признаков цепей представлена на Рис. 2.

Вторая часть отбора признаков заключалась в устранение мультиколлинеарности – ситуации сильной корреляции признаков между собой, которая приводит к неустойчивости оценок весов модели и затрудняет их интерпретацию. Для учета мультиколлинеарности,

обычно, исследуют попарные корреляции между признаками и в зависимости от порогового значения коэффициента взаимной корреляции признаков, оставляют тот признак, который демонстрировал более высокую корреляцию с целевой переменной. Поскольку выбор пороговых значений корреляции и методы устранения мультиколлинеарности не гарантируют автоматического улучшения качества моделей, в данной работе был применен сравнительный подход. Были сформированы четыре альтернативных набора признаков, варьирующих строгость отбора:

- набор признаков “А” – порог корреляции больше 0,1, устранение мультиколлинеарности не применялось;
- набор признаков “В” – порог корреляции больше 0,3, устранение мультиколлинеарности не применялось;
- набор признаков “С” – порог корреляции больше 0,1, устранение мультиколлинеарности при взаимной корреляции больше 0,9;
- набор признаков “D” – порог корреляции больше 0,3, устранение мультиколлинеарности при взаимной корреляции больше 0,9.

Признаки, которые попали в соответствующие наборы, приведены в табл. 4.

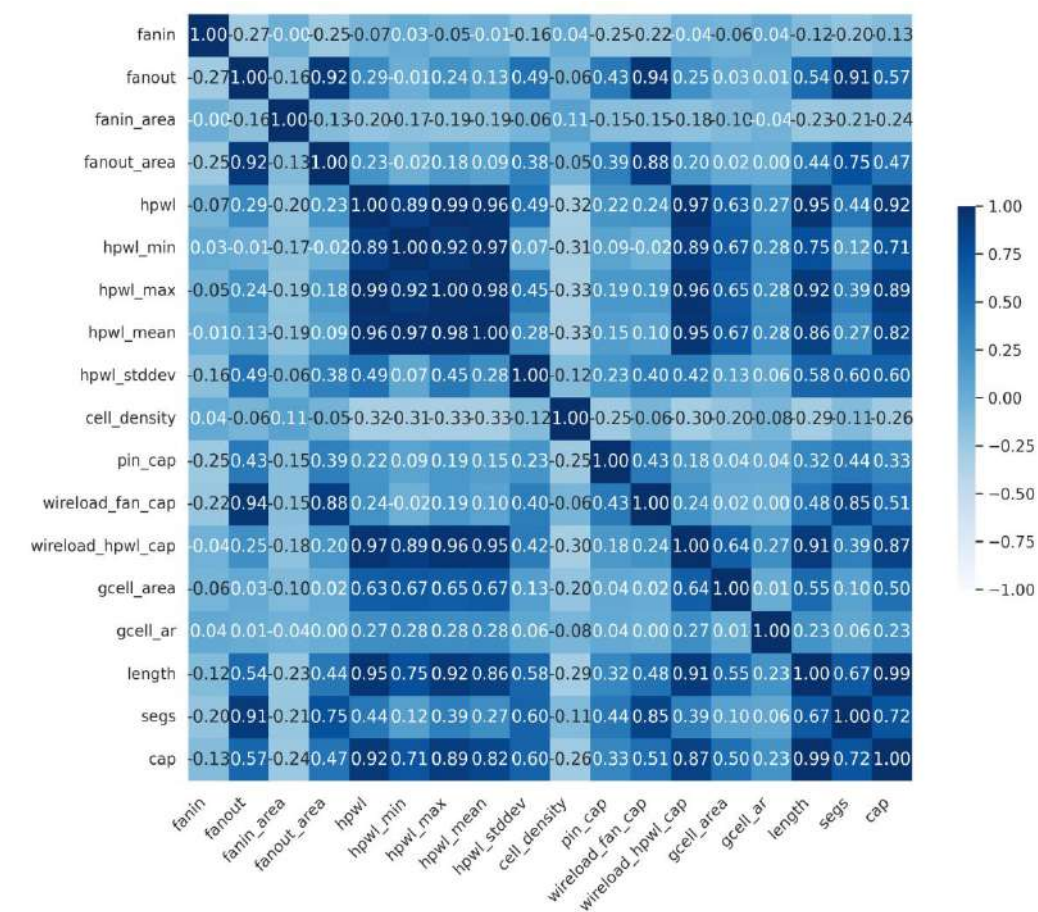


Рис. 2. Корреляционная матрица.
Fig. 2. Correlation matrix.

Сформированные наборы признаков послужили основой для сравнительного анализа, представленного в следующем разделе. Каждый из четырех наборов использовался для обучения и оценки идентичных моделей машинного обучения, что позволило количественно оценить влияние различных стратегий отбора признаков на итоговое качество прогнозирования паразитных ёмкостей.

Табл. 4. Наборы признаков.

Table 4. Feature sets.

Набор признаков	Признаки, входящие в набор	Число признаков
A	'fanin', 'fanout', 'fanin_area', 'fanout_area', 'hplwl', 'hplwl_min', 'hplwl_max', 'hplwl_mean', 'hplwl_stddev', 'cell_density', 'pin_cap', 'wireload_fan_cap', 'wireload_hplwl_cap', 'gcell_area', 'gcell_ar'	15
B	'fanout', 'fanout_area', 'hplwl', 'hplwl_min', 'hplwl_max', 'hplwl_mean', 'hplwl_stddev', 'pin_cap', 'wireload_fan_cap', 'wireload_hplwl_cap', 'gcell_area'	11
C	'fanin', 'fanout', 'fanin_area', 'hplwl', 'hplwl_min', 'hplwl_stddev', 'cell_density', 'pin_cap', 'gcell_area', 'gcell_ar'	10
D	'fanout', 'hplwl', 'hplwl_min', 'hplwl_stddev', 'pin_cap', 'gcell_area'	6

3.3 Предложенные модели

Для решения задачи предсказания паразитной ёмкости цепи в данной работе предложены различные варианты архитектур нейронных сетей. Рассмотрим архитектуры этих моделей: Первая модель MLP_C состоит из двух основных компонент: кодировщика признаков и регрессора. Кодировщик состоит из 3 линейных слоев размерностью 64, 32 и 16 нейронов. После каждого слоя используются слой нормализации (LayerNorm) и слой Dropout (0,2) для устойчивости модели и борьбы с переобучением. На выходе слоев используется функция активации ReLU. После кодировщика вектор представления передается в регрессионную часть (2 полносвязных слоя из 16 и 8 нейронов) для предсказания одной метки “cap” – паразитной ёмкости.

Вторая модель MLP_CL отличается от первой тем, что на основе векторного представления после кодировщика предсказывает длину цепи, конкатенирует это предсказание с выходом кодировщика и подает на регрессор, предсказывающий ёмкость. Основная мотивация использовать многозадачные модели (эту и последующие) заключается в том, что многозадачное обучение может позволять выявлять общие закономерности в данных и использовать синергетический эффект при совместном прогнозировании связанных величин. В то же время при многозадачном обучении кодировщик может создавать более содержательные представления, релевантные сразу для нескольких связанных физических характеристик, что может улучшить обобщающую способность модели.

Третья модель MLP_CLS схожа со второй моделью, но кроме предсказания длины также предсказывает количество сегментов цепи, после чего представление кодировщика конкатенируется с предсказанием длины цепи и числом ее сегментов для подачи на регрессор.

Четвертая модель MLP_C_L_S обладает таким же кодировщиком на входе, как и предыдущие модели, однако затем представление передается параллельно в 3 независимых регрессора для предсказания ёмкости, длины и числа сегментов. Каждая выходная метрика оптимизирует свою функцию потерь, а общая функция потерь модели – взвешенная сумма частных потерь.

Пятая модель основана на алгоритме градиентного бустинга над решающими деревьями XGBoost [9]. Модель предсказывала паразитную ёмкость на основе признаков цепи из набора данных. Модель была создана со следующими параметрами: количество деревьев – 100, максимальная глубина – 10, скорость обучения – 0,1.

Шестая модель основана на еще одном алгоритме градиентного бустинга над решающими деревьями – CatBoost [10]. Модель была создана с такими же параметрами, как и XGBoost. Данные алгоритмы демонстрируют высокую эффективность при работе с табличными данными за счет встроенной регуляризации и отсутствия необходимости в предобработке данных.

Седьмая и последняя архитектура модели XGBoost_M реализует многозадачную регрессию для одновременного предсказания ёмкости, длины и числа сегментов цепи. Модель создана с теми же параметрами, что и другие модели градиентного бустинга в этой работе.

Предложенные выше модели будут сравниваться в точности с классической метрикой Wireload (в наборе данных представлена в качестве признака “wireload_fan_cap”).

Графическое изображение предложенных моделей представлено в GitHub репозитории авторов данной статьи [11].

3.4 Обучение моделей

При обучении и валидации приведенных в предыдущем разделе моделей для объективной оценки использовалась кросс-валидация по блокам методом k-Fold [12]. Каждый блок из набора данных рассматривался как отдельная единица: в каждом цикле обучения одна подгруппа блоков выделялась в качестве тестовой выборки, в то время как остальные использовались для обучения и валидации. Такой подход позволяет избежать утечек данных между обучающими и тестовыми блоками и обеспечивает более реалистичную проверку способности моделей к обобщению на новые, ранее не виденные блоки. В данной работе набор данных из 35 блоков разбивался в процессе обучения на 7 частей (фолдов) по 5 блоков. На каждой итерации обучения 6 фолдов использовались для обучения модели, а 1 фолд для ее валидации. Процесс повторялся 7 раз таким образом, что каждый блок побывал в тестовом наборе только один раз.

В качестве метрик точности для задачи регрессии были выбраны средняя абсолютная ошибка (MAE), средняя абсолютная ошибка в процентах (MAPE) и коэффициент детерминации (R^2). В разделе с результатами итоговые значения метрик вычислялись как среднее по всем фолдам. В качестве функции потерь при обучении использовалась среднеквадратичная ошибка (MSE). В качестве оптимизатора – Adam [13] с параметром $\text{weight decay} = 0,0001$, используемом для L2-регуляризации в качестве борьбы с переобучением [14]. Модели обучались на протяжении 20 эпох.

Для обеспечения сходимости алгоритмов и устранения влияния масштаба признаков на процесс обучения была применена стандартизация данных. Все числовые признаки масштабировались с использованием метода StandardScaler [15], который преобразует данные к распределению с нулевым средним значением и единичной дисперсией. Процедура выполнялась отдельно для каждого фолда кросс-валидации: параметры масштабирования (среднее значение и стандартное отклонение) вычислялись исключительно на тренировочной выборке, после чего применялись к тестовой выборке. Такой подход предотвращает информационную утечку и обеспечивает корректную оценку обобщающей способности моделей.

Обучение проводилось локально на компьютере с ОС Windows 10, процессор Intel Core i5 9400F, 16 ГБ оперативной памяти формата DDR4, видеокарта NVIDIA GeForce GTX 1660 SUPER с 6 ГБ видеопамяти с 1408 CUDA ядрами. В качестве окружения использовался Jupyter Lab (версия 4.3.6), Python 3.12, PyTorch 2.6.0+cu124 и PyTorchGeometric 2.6.1. Версия XGBoost 2.1.4, версия CatBoost 1.2.8. Все вычисления выполнялись на GPU.

4. Результаты

В данном разделе будут представлены результаты обучения предложенных моделей оценки паразитной ёмкости, приведено сравнение их точности и сделаны основные выводы о преимуществах предложенных архитектур, а также о выборе стратегии отбора признаков для обучения. По итогам сравнения будет выбрана лучшая модель, результаты которой будут использоваться для решения задачи оценки частоты работы блока на этапе размещения ячеек.

4.1 Модели оценки паразитной ёмкости

Стоит отметить, что суммарное число моделей, рассмотренных в сравнении, будет равняться 29 – 7 предложенных архитектур будут обучаться на 4 наборах признаков, представленных в табл. 4, а также классическая метрика Wireload. Это позволит определить среди всех архитектур и подходов к отбору признаков наилучший вариант. Название моделей с учетом используемых входных признаков в данной статье строится следующим образом – после названия выбранной архитектуры через нижнее подчеркивание указывается название набора признаков из табл. 4. То есть модель MLP_C_A – первая архитектура нейронной сети, обученная на наборе признаков “А”. В табл. 5 представлены итоговые метрики 29 моделей по итогам обучения с кросс-валидацией (значения в таблице строятся как среднее $\pm$ стандартное отклонение по всем фолдам) для задачи предсказания паразитной ёмкости цепей.

Отметим главные выводы из сравнения точности моделей:

- все предложенные модели нейронных сетей превосходят классическую метрику Wireload – улучшение метрик MAE и MAPE составляет в среднем 10 раз;
- модели на основе XGBoost показывают наилучшие результаты среди всех моделей, превосходя в том числе и модели на основе CatBoost;
- анализ четырех наборов входных признаков выявил, что набор А – демонстрирует стабильно высокие результаты среди всех моделей; набор D – показывает лучшие результаты для MLP архитектур;
- агрессивный отбор признаков (с порогом отбора признаков больше 0,3 и устранение мультиколлинеарности) не всегда приводит к улучшению качества модели;
- применение многозадачного обучения дает сравнительно небольшой прирост точности по сравнению с обучением модели только на предсказании ёмкости – в среднем на 7%; самая эффективная из нейросетевых моделей – MLP_C_L_S_D на основе многозадачной архитектуры имеет среднюю ошибку в 14,7%;
- наименьшую ошибку показывают 2 модели – XGBoost_M_A и XGBoost_A; с небольшим отрывом лидирует XGBoost_M_A – на основе градиентного бустинга в режиме мультизадачной регрессии для набора признаков А без агрессивного отбора (порог корреляции больше 0,1 без устранения мультиколлинеарности); MAE предсказания паразитной ёмкости составляет 0,57 фФ, а MAPE 10,3%.

Также был проведен анализ времени, затрачиваемого на работу лучшей модели XGBoost_M_A:

- среднее время выписки признаков на кросс-валидации для модели 39 секунд;
- среднее время работы модели на кросс-валидации 3 секунды;
- среднее ускорение по сравнению с эталонным методом получения паразитных ёмкостей (после проведения трассировки и экстракция в САПР) на кросс-валидации составляет 90 раз.

Табл. 5. Сравнение точности моделей в задаче предсказания паразитной ёмкости.
Table 5. Comparison of the accuracy of models in the problem of predicting parasitic capacitance.

№	Модель	MAE, фФ	MAPE, %	R ²
1	Wireload	7,87 ± 3,84	169,7 ± 80,2	0,23 ± 0,15
2	MLP_C_A	0,85 ± 0,38	16,4 ± 3,0	0,90 ± 0,06
3	MLP_C_B	0,83 ± 0,35	20,2 ± 4,2	0,91 ± 0,05
4	MLP_C_C	0,80 ± 0,34	16,3 ± 3,5	0,92 ± 0,05
5	MLP_C_D	0,77 ± 0,31	15,6 ± 3,6	0,92 ± 0,04
6	MLP_CL_A	0,79 ± 0,35	15,2 ± 4,2	0,92 ± 0,04
7	MLP_CL_B	0,79 ± 0,35	16,3 ± 3,9	0,91 ± 0,05
8	MLP_CL_C	0,82 ± 0,33	17,5 ± 2,8	0,92 ± 0,05
9	MLP_CL_D	0,78 ± 0,37	15,1 ± 2,9	0,92 ± 0,05
10	MLP_CLS_A	0,81 ± 0,40	16,9 ± 5,3	0,92 ± 0,04
11	MLP_CLS_B	0,79 ± 0,39	15,8 ± 3,2	0,92 ± 0,05
12	MLP_CLS_C	0,83 ± 0,38	17,3 ± 5,5	0,91 ± 0,06
13	MLP_CLS_D	0,78 ± 0,37	15,1 ± 2,9	0,92 ± 0,05
14	MLP_C_L_S_A	0,79 ± 0,33	15,2 ± 3,3	0,91 ± 0,06
15	MLP_C_L_S_B	0,77 ± 0,27	18,9 ± 7,9	0,93 ± 0,04
16	MLP_C_L_S_C	0,77 ± 0,37	15,3 ± 4,8	0,93 ± 0,05
17	MLP_C_L_S_D	0,73 ± 0,32	14,7 ± 2,4	0,93 ± 0,04
18	XGBoost_A	0,58 ± 0,30	10,4 ± 2,1	0,93 ± 0,05
19	XGBoost_B	0,59 ± 0,28	10,9 ± 1,8	0,93 ± 0,05
20	XGBoost_C	0,59 ± 0,28	10,6 ± 2,1	0,93 ± 0,04
21	XGBoost_D	0,60 ± 0,23	12,1 ± 2,5	0,91 ± 0,06
22	CatBoost_A	0,63 ± 0,30	12,8 ± 2,2	0,93 ± 0,04
23	CatBoost_B	0,63 ± 0,28	12,7 ± 2,1	0,94 ± 0,04
24	CatBoost_C	0,60 ± 0,26	12,2 ± 1,7	0,94 ± 0,04
25	CatBoost_D	0,61 ± 0,24	12,6 ± 2,0	0,94 ± 0,04
26	XGBoost_M_A	0,57 ± 0,30	10,3 ± 2,1	0,94 ± 0,05
27	XGBoost_M_B	0,59 ± 0,28	10,9 ± 1,8	0,93 ± 0,05
28	XGBoost_M_C	0,59 ± 0,28	10,6 ± 2,1	0,93 ± 0,04
29	XGBoost_M_D	0,60 ± 0,23	12,1 ± 2,5	0,91 ± 0,06

4.2 Оценка быстродействия блока

Результаты оценки паразитной ёмкости, полученные моделью XGBoost_M_A, были применены в решении другой важной практической задачи – оценке быстродействия блока

на этапе размещения ячеек. В данной задаче требуется оценить, какая будет частота работы блока после этапа трассировки (вносящего изменения в блок и расчёт временных задержек), обладая лишь информацией о расположении ячеек без данных о детальной трассировке. Поскольку учет паразитной ёмкости на этапе трассировки является основной причиной, влияющей на изменение быстродействия, мы предлагаем следующий алгоритм оценки быстродействия на этапе размещения с помощью созданной модели из предыдущего раздела:

- первый этап – выписка признаков блока, доступных на этапе размещения;
- второй этап – расчёт паразитных ёмкостей цепей с помощью модели, то есть инференс модели; формирование скрипта с набором команд, которые зададут в САПР паразитные ёмкости на всех рассматриваемых цепях;
- выполнение скрипта установки паразитных ёмкостей в САПР и расчёт частоты работы блока через оценку запаса по времени установки (Worst Negative Slack, WNS) во время статического временного анализа.

Оценка быстродействия предложенным методом (для краткости обозначим его ML) сравнивалась со следующими методами:

- Route – эталон, данные о частоте блока получены после проведения детальной трассировки и экстракции паразитных параметров из САПР;
- Wireload – быстродействие оценивалось на этапе размещения, паразитная ёмкость цепей рассчитывалась с помощью Wireload-модели;
- ER (Early Route) – быстродействие оценивалось на этапе размещения с помощью специального инструмента Early Route в САПР Innovus. Он позволяет выполнить быструю трассировку, но с возможными нарушениями правил проектирования, несовпадением проводников с рекомендованными треками для данной технологии и рядом других допущений; такая ранняя разводка используется в САПР для оценки разводимости (congestion), а также для более точной оценки временных параметров (в том числе и быстродействия).

В табл. 6 представлено сравнение точности оценки быстродействия указанными выше методами, а также ускорение времени работы относительно эталонного алгоритма с использованием детальной трассировки и экстракции паразитных параметров в САПР Quantus. Данные в таблице представлены в формате “среднее” $\pm$ “стандартное отклонение”, поскольку расчёт метрик проходил внутри 7 фолдов на кросс-валидации – на 6 фолдах проходило обучение модели предсказывать паразитную ёмкость цепей, на блоках из оставшегося фолда проходила оценка быстродействия.

Табл. 6. Сравнение точности оценки быстродействия и затрат времени на работу модели.

Table 6. Comparison of the accuracy of the performance and the time required to run the model.

Модель	Ускорение	Точность оценки частоты, %
Route	1x	100
Wireload	$(97 \pm 90)x$	$91,2 \pm 12,6$
ER	$(8 \pm 6)x$	$99,1 \pm 0,7$
ML	$(86 \pm 80)x$	$99,1 \pm 0,9$

Отметим главные выводы из сравнения точности моделей оценки быстродействия, а также времязатрат:

- предложенный ML-метод (на основе модели XGBoost_M_A для оценки паразитной

ёмкости) демонстрирует сравнимые по точности оценки быстродействия блока с внутренним методом ER из САПР Innovus (точность оценки 99,1%), обладает высокой стабильностью оценки (стандартное отклонение 0,9%), а также обеспечивает значительное ускорение (~ 86 раз) по сравнению с эталонным методом и ускорение по сравнению с методом ER (~ 10 раз);

- ML-метод превосходит оценку с использованием Wireload-модели по точности (99,1 % против 91,2%), а также значительно опережает в стабильности (отклонение 0,9% против 12,6%);
- ML-метод незначительно уступает Wireload по времени работы (в среднем на 13%) – дополнительное время тратится на выписку признаков и инференс модели XGBoost_M_A).

Также был проведен статистический анализ абсолютной ошибки оценки частоты работы блока – рассчитан 95% доверительный интервал для 3 методов. Результаты анализа:

- Wireload – от 8,3 до 24,9 МГц;
- ER – от -2,1 до -0,1 МГц;
- ML – от 0,7 до 2,3 МГц.

Интерпретация доверительных интервалов данных методов показывает, что метод ER систематически завышает частоту, метод ML систематически занижает частоту и обеспечивает более консервативную оценку, а метод Wireload демонстрирует систематическое заметное занижение частоты.

5. Заключение

В данной работе решалась задача оценки паразитной ёмкости цепей в цифровом блоке СБИС на этапе размещения ячеек с помощью методов машинного обучения. Был предложен ряд моделей машинного обучения для решения данной задачи: линейные модели в однозадачном и многозадачном режиме регрессии, модели на основе градиентного бустинга. Был собран набор данных для обучения моделей по технологии 40 нм для 35 RTL-моделей с открытым исходным кодом. Проведен отбор признаков и обучение 28 моделей с различными наборами признаков. Наибольшую точность продемонстрировала модель XGBoost_M_A – модель на основе градиентного бустинга в режиме мультизадачной регрессии для набора признаков A без агрессивного отбора (порог корреляции больше 0,1 без устранения мультиколлинеарности). Средняя абсолютная ошибка на кросс-валидации составила 0,57 фФ (в процентах 10,3%), что на порядок ниже ошибки, полученной классическим методом на основе Wireload-модели. Среднее ускорение получения паразитных параметров по сравнению с эталонным методом – 90 раз.

На основе модели XGBoost_M_A была предложена модель ML для решения задачи оценки быстродействия блока на этапе размещения. Ее результаты сравнивались с результатами моделей на основе Wireload, на основе инструмента Early Route из САПР Innovus, а также с эталонным значением после трассировки и экстракции паразитных параметров. Предложенная модель ML продемонстрировала высокую точность в оценке быстродействия – 99,1% (сравнимо с ER-моделью), значительно превосходит в точности модель на основе Wireload (точность 91,2%), а также обеспечивает значительное ускорение (~ 86 раз) по сравнению с эталонным методом, а также по сравнению с методом ER (~ 10 раз), что делает данную модель перспективной для ее использования при проектировании цифровых блоков СБИС для оценки быстродействия на раннем этапе проектирования ввиду баланса ее высокой точности и быстром времени работы по сравнению с другими методами.

Список литературы / References

- [1]. Bhasker J., Chadha R. Static timing analysis for nanometer designs: A practical approach. Boston: Springer Science & Business Media, 2009, 572p.
- [2]. Windschiegi A., Zuber P., Stechele W. A wire load model for more accurate power estimation // The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002. Tulsa, OK, USA: IEEE, 2002. С. I-376– I-9.
- [3]. Kahng A. B., Spyrou T. The OpenROAD project: Unleashing hardware innovation. In Proc. GOMAC, 2021, pp. 1-6.
- [4]. Shen S. и др. Deep-Learning-Based Pre-Layout Parasitic Capacitance Prediction on SRAM Designs, 2025. Available at <https://arxiv.org/pdf/2507.06549>, accessed 18.02.2026.
- [5]. Dogan E., Guthaus M. R. Effective Capacitance Modeling Using Graph Neural Networks, 2025. Available at <https://arxiv.org/pdf/2507.03787>, accessed 18.02.2026.
- [6]. OpenABC Project, “OpenABC: Automated Block-Level RTL Synthesis Flow,” GitHub repository, <https://github.com/NYU-MLDA/OpenABC>, accessed Dec. 10, 2025.
- [7]. OpenCores, “OpenCores — Open Source Hardware IP Cores,” <https://opencores.org/>, accessed Dec. 10, 2025.
- [8]. B. Akhshali pour, “NoC-Verilog: Network-on-Chip Verilog RTL Models,” GitHub repository, <https://github.com/bakhshali pour/NoC-Verilog>, accessed Dec. 10, 2025.
- [9]. Chen T., Guestrin C. Xgboost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 785-794.
- [10]. Prokhorenkova L. et al. CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems, 2018, vol. 31.
- [11]. N. Zheludkov, “ML-Based Estimation of Interconnect Capacitance,” GitHub repository. <https://github.com/NZheludkov/ML-Based-Estimation-of-Interconnect-Capacitance>. Accessed: Dec. 10, 2025.
- [12]. Bengio Y., Grandvalet Y. No Unbiased Estimator of the Variance of K-Fold Cross-Validation. Journal of Machine Learning Research, 2004, vol, pp. 1089–1105.
- [13]. Kingma D. P., Ba J. L. Adam: A method for stochastic optimization, 2014. Available at <https://arxiv.org/pdf/1412.6980>, accessed 05.11.2025.
- [14]. Loshchilov I., Hutter F. Decoupled weight decay regularization, 2017. Available at <https://arxiv.org/abs/1711.05101>, accessed 05.11.2025.
- [15]. De Amorim L. B. V., Cavalcanti G. D. C., Cruz R. M. O. The choice of scaling technique matters for classification performance. Applied Soft Computing, 2023, vol. 133, pp. 109924.

Информация об авторах / Information about authors

Никита Владимирович ЖЕЛУДКОВ – заведующий сектором, младший научный сотрудник в НИЦ "Курчатовский институт" – НИИСИ. Сфера научных интересов: проектирование СБИС, систем на кристалле, применение методов машинного обучения в проектировании СБИС, автоматизация проектирования.

Nikita Vladimirovich ZHELUDKOV – Head of Sector, Junior Research Fellow at the NRC “Kurchatov Institute” – SRISA. Area of scientific interests: VLSI design, system-on-chip, application of machine learning methods in VLSI design, design automation.

Юлия Андреевна КУЛЕМИНА – инженер и аспирант в НИЦ "Курчатовский институт" – НИИСИ. Сфера научных интересов: проектирование аналоговых блоков СБИС, применение методов машинного обучения в процессе разработки аналоговых блоков, автоматизация проектирования.

Yuliya Andreevna KULEMINA – Engineer and Postgraduate Student at the NRC “Kurchatov Institute” – SRISA. Area of scientific interests: Analog IC design, application of machine learning methods in the analog design flow, design automation.

DOI: 10.15514/ISPRAS-2026-38(1)-10



A New Interpretation of the Tense System of Vakh-Vasyugan Khanty on the Up-to-date Corpus Data of the LingvoDoc Lab

^{1,2} V.V. Vorobeve, ORCID: 0000-0003-1559-8810 <victoriavorobeve@mail.ru>

² Z. V. Fedorinova, ORCID 0000-0002-6700-0617 <fav1@rambler.ru>

¹ Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

² National Research Tomsk Polytechnic University,
30, Lenin ave., Tomsk, 634050, Russia.

Abstract. This article presents findings resulting from an analysis of the system of verbal tense markers in Vakh-Vasyugan Khanty. The aim of this study is to establish a range of meanings of verbal temporal suffixes in terms of their typology and to provide their new interpretation. It has been established that the temporal system of Vakh-Vasyugan Khanty is represented by a two-member opposition of “past / non-past” types. The system includes suffix representatives that are unevenly distributed over the corresponding fragments of the temporal field. The non-past suffixes *-l* / *-w* / *-wəl* / *-wəl* mark present and future actions. For an action in the past, it is necessary to select from markers *-s*, *-yas* / *-yäs* or *-yal* / *-yäl* indicating different remoteness of the past action from the moment of speech. The semantics of the zero marker $\emptyset$ seems to be the most difficult to interpret because the linguistic data shows the ability of these suffixes to attribute an action to both the past and present temporal domains. All four past tense markers are consistently used in the samples of the modern speech of the Vakh-Vasyugan Khanty. Calculations on the LingvoDoc platform show that the past tense marker on *-s* is the most frequently used.

Keywords: Vakh-Vasyugan Khanty; field data; text corpora; LingvoDoc; data analysis; language documentation; verbal morphology; tense; temporal markers.

For citation: Vorobeve V.V., Fedorinova Z. V. A new interpretation of the tense system of Vakh-Vasyugan Khanty on the up-to-date corpus data of the LingvoDoc Lab. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 1, 2026, pp. 133-148. DOI: 10.15514/ISPRAS-2026-38(1)-10.

Acknowledgements. The authors are grateful to the anonymous reviewers for their insightful comments to the article. This research was funded by the Russian Science Foundation, project no. 25-78-20002 “Capabilities of Artificial Intelligence for Comparative-Historical Study of Low-Resource Languages of the Peoples of the Russian Federation”. The results were obtained using the services of the Ivannikov Institute for System Programming (ISP RAS) Data Center

Новая интерпретации временной системы вах-васюганского хантыйского на современных корпусных данных платформы ЛингвоДок

^{1,2} В.В. Воробьева, ORCID: 0000-0003-1559-8810 <victoriavorobeva@mail.ru>

² З.В. Федоринова, ORCID 0000-0002-6700-0617 <fav1@rambler.ru>

¹ Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, Москва, ул. Александра Солженицына, 25.

² Национальный Исследовательский Томский политехнический университет,
Россия, 634050, проспект Ленина, д. 30.

Аннотация. В статье анализируется система временных (темпоральных) показателей глагола в вах-васюганском хантыйском. Цель исследования заключается в установление спектра значений глагольных темпоральных суффиксов с точки зрения их типологического своеобразия по сравнению с подобными маркерами других диалектов хантыйского. Установлено, что временная система вах-васюганского хантыйского представлена двучленной оппозицией по типу «прошедшее/непрошедшее», суффиксальные репрезентанты которой неравномерно распределяются по соответствующим фрагментам временного поля. Непрошедшее время маркируется суффиксами *-l/-w/-wəl/-wəl*, в то время как для обозначения действия в прошлом требуется выбрать суффиксальный показатель *-s*, *-yas/-yās* или *-yal/-yäl*, указывающих на различную отдаленность прошлого действия от момента речи. Семантика нулевого маркера $\emptyset$ представляется наиболее сложной для интерпретации, поскольку языковой материал указывает на способность данной граммы относить действие как прошедшему, так и настоящему времени. Все четыре маркера прошедшего времени стабильно употребляются в образцах современной речи вах-васюганских ханты. Вычисления на платформе ЛингвоДок показывают, что маркер прошедшего времени на *-s* самый частоупотребимый.

Ключевые слова: Вах-васюганский хантыйский; полевые данные; текстовые корпуса; платформа LingvoDoc; глагольные категории; грамматическое время; темпоральные маркеры.

Для цитирования: Воробьева В.В., Федоринова З.В. Новая интерпретация временной системы вах-васюганского хантыйского на современных корпусных данных платформы ЛингвоДок. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 133–148 (на английском языке). DOI: 10.15514/ISPRAS–2026–38(1)-10

Благодарности. Авторы выражают искреннюю благодарность анонимным рецензентам за их ценные комментарии к статье. Исследование выполнено при финансовой поддержке РФФ, грант № 25-78-20002 «Возможности искусственного интеллекта для сравнительно-исторического изучения малоресурсных языков народов РФ». Результаты получены с использованием услуг Центра коллективного пользования Института системного программирования им. В.П. Иванникова РАН – ЦКП ИСП РАН.

1. Introduction

The purpose of this article is to clarify the range of meanings of temporal verb markers and to demonstrate the distinctive features of the tense system of Vakh-Vasyugan Khanty, utilizing available online tools on the LingvoDoc platform.

The study was conducted using corpora of the Vakh and Vasyugan varieties of Khanty, belonging to the Eastern group [1-2]. The language data that formed the basis of this research was obtained by the author during fieldwork conducted in the summer of 2019 in the National Village of Korliki, located in the Khanty-Mansi Autonomous Okrug – Yurga. The data was elicited from E.A. Kunina (born in 1943) and Zh.A. Khokhlyankina (born in 1967).

Both informants were brought up in Khanty-speaking families who lived on their ancestral lands. Family communication primarily occurred in their native language. All informants are bilingual and assess their native language proficiency as 'semi-speaker' according to the classification of K. Grinvalds [3: 12-15].

The method of data collection for this research involved the use of questionnaires. To elicit the necessary data, pre-processed and translated-into-Russian questionnaires based on those developed by Östen Dahl [4: 198-206] were used. The same questionnaire, focused on revealing the temporal and aspectual characteristics of the Khanty verb, was administered twice to each respondent, i.e., E.A. Kunina and Zh.A. Khokhlyankina. The analysis also included prose texts, stories, and fairy tales recorded from the indigenous inhabitants of the village. Two main fairy tales, "Two Sister Birds" and "The Mouse's Song", narrated by the storyteller T. F. Katkaleva, a retired Khanty language teacher, were transcribed by E. A. Kunina. The volume of the collected and analyzed language data corpus is approximately 5,000 tokens

The study presented in the article was conducted based on the theoretical principles of general and typological linguistics [4-12].

Among the multitudinous definitions of tense in the literature, we may cite those given in Comrie: tense is grammaticalisation of location in time [13: 6].

In typological literature, three types of temporal deixis are distinguished: absolute (deictic, primary), relative (anaphoric, secondary), and metric. Grammmemes of absolute temporal deixis link a situation to the present, past, or future field of the temporal axis; for example, eats, ate, will eat. Grammmemes of relative temporal deixis localize a situation within the temporal interval of present, past, and future times; for example, English had read, would read. Grammmemes of metric temporal deixis additionally distinguish degrees of remoteness from the moment of speech. In the temporal systems of the world's languages, metric temporal systems are typologically rare [4-9].

The tense system of the Khanty verb is a two-member opposition of "past/non-past" types. The verb form contains morphemes expressing voice, aspect, tense, and person-number. The absolute tense deixis is represented by the morphemes of the past and non-past tense. Khanty lacks aspectual oppositions of perfective aspect / imperfective aspect, see, for example, [14: 55-70; 15:103-123; 16: 47-52]. Aspect is mixed with modes of action [17: 215-216; 18: 92; 19]. The latter position, along with the denial of the category of aspect, was until recently traditional in relation to most other Uralic languages, see, for example, [20: 23; 21: 70-80; 22: 26].

The existing studies of the verbal morphology of the northern dialects of Khanty allow us to conclude that they possess an aspect-tense verb system. When researching the mood-tense category in the Northern dialects, A. D. Kaksin [23] noted some aspectual nuances (for example, duration and perfection) revealed in the meaning of tense verb forms. However, the most widespread meanings from the semantic zone of aspectuality involved in the mood-tense category system of Khanty are the oppositional meanings of perfection and imperfection. For the means of conveying the meaning of perfection in Northern Khanty, see [23]. According to the author, the expression of one or another aspectual meaning depends not only on the choice of tense form but also on the conditions of the context corresponding to the circumstance of time and the lexical meaning of the verb.

Northern Khanty and Eastern Khanty have different inventories of tense forms. The significant uniqueness of Eastern Khanty, particularly the Vakh-Vasyugan dialect, is its possession of four tense forms. For example, in the Northern Khanty dialects, the system of tense forms is not characterized by the criterion of closeness/remoteness, while in Eastern Vakh-Vasyugan Khanty, this difference is present. In addition, such a variety of past tense markers (four) is not found in any other dialect of Khanty. In Surgut Khanty, which is geographically closest to Vakh Khanty, one of the past tense forms, marked by the suffix -s, has practically lost its use in the modern language [24].

Researchers of Vakh-Vasyugan Khanty (Tereshkin, Filchenko Gulya, Honti) offered different analyses of the meanings of past tense forms.

N.I. Tereshkin interprets the four past tense markers as being distinguished by two semantic parameters: remoteness and definiteness. In the scholar's analysis, the past tense marker -s indicates short distance and definiteness, while the zero marker Ø indicates long distance and definiteness. In

contrast, the marker -yäs/-yas signifies short distance and indefiniteness, and the marker -yal/-yäl signifies long distance and definiteness [25: 81-82].

Filchenko's monograph [26: 244-245], devoted to the grammatical description of the Vasyugan dialect, likewise presents the temporal opposition as a binary distinction of the "past/non-past" type. Within the domain of the past tense, the principal criteria governing the selection of a specific past tense marker are the relative temporal distance from the moment of speech and the relative relevance to the moment of speech. A relatively recent past event that is more relevant to the moment of speech is marked by the suffix *-s*, while one that is less relevant is marked by the suffix *-yas/-yäs*. Furthermore, a relatively distant past event from the speech act that is more definite or relevant to the speech act is marked by the suffix *-yal/-yäl*, while one that is less definite in its location or less relevant to the moment of speech is not marked [26: 244-245].

According to J. Guya, the past tense suffix *-s* is used to mark a recently completed or just-happened action. Furthermore, this suffix is used to denote a completed action, sometimes of short duration, in the past. The past tense suffix *-yas/-yäs* indicates an action completed in the past prior to another action marked by the suffix *-s*. The past tense suffix *-yal/-yäl* is used to express a completed, long-lasting action that occurred relatively long ago in the past. Verbs with the suffixes *-s*, *-yas/-yäs*, and *-yal/-yäl* can be associated with a specific moment in the past, but their relative ordering is not established. The suffixless past tense denotes the past without reference to a specific period [15: 110-111].

It is generally accepted among scholars that the grammatical forms contrast with respect to their temporal distance from the moment of speech: *-s* is opposed to the zero marker $\emptyset$, and *-yas/-yäs* is opposed to *-yal/-yäl*. However, scholars differ on the second semantic criterion. N.I. Tereshkin employs the notion of definiteness, whereas A.Yu. Filchenko favors relevance. The Hungarian scholars L. Honti [16: 51] and J. Gulya include completeness as a parameter in their classification of temporal markers. In our study, we analyze the three temporal markers in terms of their distance from the moment of utterance, characterizing the zero temporal marker as possessing neutral semantics.

2. Results of analysis and their interpretation

The tense system of Vakh-Vasyugan Khanty consists of a two-member opposition between past and non-past tense markers. Semantically, the past tense domain features more markers than the present-future domain, which is represented by markers of differing forms but with a single meaning.

2.1. NON-PAST tense marker -l-/w-/wəl-/wəl (NPST)

The suffix -w is used in the 2nd person singular. The suffix -wāl/-wəl is used in the 3rd person singular and plural. According to Tereshkin, the suffix -wāl/-wəl is a combination of -w and -l – markers of the non-past tense. In all other persons and numbers, the suffix -l is used [25: 80] The non-past tense markers -l/-w/-wəl/-wāl indicate situations occurring in the present or future. In Khanty, only the Vakh-Vasyugan dialect has several markers for expressing the non-past tense. These markers are morphological variants with no semantic distinction. The specific meaning (present or future) of the markers -l/-w/-wəl/-wāl is determined by context.

We obtained Narrative (1) by asking a native speaker to recount what would happen to him tomorrow. This elicited the use of the markers *-l/-w/-wəl/-wəl*, which, in this context, signify future events.

- (1a) *mä* *käs-I-äm* *möŋkäm*
 1SG find-**NPST**-SBJ.1SG snake
 (I'll find a snake.)
- (1b) *män-nə* *köy* *kätə-I-äm* *päni* *jöy-I-im* *möŋkäm-pä*
 1SG stone grab-**NPST**-SBJ.3SG and throw-**NPST**-SBJ.3SG:OBJ.SG monkey-ALL
 (I'll grab a stone and throw it at a monkey.)

(I'll grab a stone and throw it at the snake.)

(1c) *moŋkām il kōlā-wāl* (V.)

monkey down die-**NPST**.SBJ.3SG

(It will die.)

The suffix *-l/-n/-wāl/-wāl*, when used in the present, expresses what is terminologically known as the “present indefinite” or “present simple”. This form describes an action occurring in the present without specifying a particular moment, for example:

(2) *muŋi jŋŋi-nā antā wāl-wāl*... (Vas.)

crucian.carp river-LOC not live-**NPST**.3SG.S

(Crucian carp does not live in the river.)

(3) *kāt'āŋ pattā lŋŋ n'oy-a kǎŋči-l-wāl* (V.)

every autumn 3SG elk hunt-HAB-**NPST**.3SG.S

(Every autumn he hunts an elk.)

It has been noted that non-past tense form frequently employ lexical items like *timkōtl* ‘today’, *it*, ‘now’, and *kōltāŋ* ‘tomorrow’, *wǎŋŋā* ‘soon’ explicitly connecting actions to the present or future time frame, as demonstrated in example (4-5, 8, 13):

(4) *ārki jay it jis-min amās-wāl-t*

many people now cry-CVB sit-**NPST**-3PL.S

(Many people are sitting and crying.)

(5) *mā kōltāŋ nōŋ-a jāl-l-ām* (V.)

1SG tomorrow 2SG-LAT come-**NPST**-1SG.S

(I'll come to you tomorrow.)

The continuous and habitual nature of an action in the present or future can be emphasized both lexically, through adverbs such as *māčāŋ* and *t'eŋ* ‘always’ or *māt* ‘constantly’, and grammatically, via the habitual suffix *-il/-il*, for example:

(6) *t'eŋ jopl-jl-l-ām mā tet* (Vas.)

always fish-HAB-**NPST**-1SG.S 1SG here

(I always fish here.)

(7) *mā māčāŋ lōŋtant-l-ām itān-nā* (V.)

1SG always читать-**NPST**-1SG.S вечер-LOC

(I read in the evenings.)

The inchoative suffix *-ākātā/-ākātā*, which marks the mode of action, can also function as a future tense marker. The polysemous nature of this suffix has been observed by several researchers [15: 98, 109; 27: 64]

(8) *kōltāŋ jāl'iys-ākātā-l-mān* (Vas.)

tomorrow fight-**INCH**-**NPST**-1DU.S

(We will start fighting tomorrow.)

The suffix *-l* is polysemous, functioning as a marker of both non-past actions and multiplicative/habitual actions. See examples (9-10):

(9) *mā əp-ŋm jay-nā erylā-l-i kǎčā-ki* (V.)

1SG father-POSS.1SG people-LOC tell-**MULT**-**NPST**.PASS.3SG pain-PRD

(They say my father is sick [constantly].)

(10) *tāltoŋ ilā-pā nōŋā-l-s-əŋən* (Vas.)

then forward move-**MULT**-**PST**1-3DU.S

(Then we went forward.)

This phenomenon is also observed in other Uralic languages, where certain present tense markers originate as aspectual suffixes of multitude [28: 186; 29:46].

Therefore, the non-past suffix *-l/-n/-wəl/-wəl* indicates that an action occurs in the present or future, potentially as a single event, a habit, or a repeated action. The specific temporal interpretation of the action depends on the situational context (time, place, setting, participants) and can be further specified by adverbs of time (e.g., ‘today’, ‘tomorrow’). This suggests that the suffix lacks an inherent “present time” or “future time” meaning in its semantic structure.

2.2 Past tense markers

The tense system in the language of the Vakh-Vasyugan Khanty is expanded in comparison with the dialects of the northern (Shuryshkar, Kazym, Obdorsk) and even other eastern Khanty living further south (Surgut dialects). In Vakh-Vasyugan Khanty four markers are distinguished in the sphere of the past tense. Linguists (Honti, Tereshkin, Filchenko, Gulya) generally agree that Vakh-Vasyugan has four past tense forms, but the meaning of the markers is interpreted ambiguously. The correlation with the past tense field is achieved by means of the morphemes: *-s*, *-yas/-yäs*, *-yal/-yäl*, and *-Ø*. They distinguish the degrees of remoteness of an action from the moment of speech. The measurement of the temporal distance between a situation in the past and the moment of speech is carried out approximately. The morphemes of the temporal distance are:

- *-s*: for marking the nearest temporal distance. In such cases, the situation occurs at the moment of speech or in the immediate vicinity of the moment of speech.
- *-yas/-yäs*: for marking close temporal distance. The situation occurred either on the same day or at a moderately close distance from the moment of speech.
- *-yal/-yäl*: for marking remote temporal distance. The situation is distant from the moment of speech by one day, an interval of several weeks, months, years, and centuries.

The semantics of the unexpressed marker *-Ø* remains contested, causing contradictory opinions among linguists and requiring further analysis. In modern Vakh Khanty, the marker *-Ø* is used to mark a distance close to the moment of speech or at the moment of speech.

2.2.1 Temporal marker *-s* (PST1)

The marker *-s* often indicates an action that occurred very recently, practically before the speaker's eyes or at the moment of speech, signifying an immediate connection to the present. For example:

(11) *muyj nuy-a nirim-tä-s-im muyj wel-s-im* (Vas.)

crucian.carp up-LAT pull.out-SMF-PST1-SG<SG.S crucian.carp kill-**PST1**-SG<1SG.S

(I pulled out the crucian carp, I got the crucian carp.)

(12) *nüŋ-än jäm-ä wer-s-i* (V.)

2SG-LOC good-ADV do-**PST1**-PASS.3SG

(You did well.)

(13) *lüy woŋj moŋ köt-ä wäyä-s päni*

3SG fox baby.animal hand-LAT take-**PST1**.3SG.S and

ata-s lüy wänñä jl kola-wəl (V.)

say-PST1.3SG.S 3SG soon down die-NPST.3SG.S

(He took the fox in his arms and said: “He will die soon”.)

The tense form *-s* in Vakh-Vasyugan Kanty describes events that happened either immediately or recently before the moment of speech, typically occurring on the same day, rarely the day before, or within a short range of days. However, there are no clear boundaries defined; the event is understood to be the first relevant event preceding the moment of speech. For example, in the following scenario

(14), the event took place yesterday. While the adverb of time *məlaqotl* ‘yesterday’ is rarely associated with the -s tense form, the focus here is on the result.

(14) *nemin möyi jəlwə lotka wə-s məlaqotl? (V.)*
 true what good boat take-**PST1.3SG.S** yesterday

(Is it true that he bought a new boat yesterday?)

The temporal suffix -s, often used to emphasize the outcome of the action (see example 15). It is particularly prominent in the unique Vakh Khanty constructions featuring the passive voice suffix in the 3rd person singular, where the subject, regardless of person or number, is marked with the locative case. See examples 16-17:

(15) *kol'a nipik qaŋča-s-tə (V.)*
 Kolya letter write-**PST1-SG<3SG.S**

(Kolya has written a letter.)

(16) *mā-nə nuy li-s-i (V.)*
 1SG-LOC up eat-**PST1-PASS.3SG**

(I have eaten.)

(17) *nūŋ-ən jəm-ə wer-s-i (V.)*
 2SG-LOC good-ADV делать-**PST1-PASS.3SG**

(You've done well.)

2.2.2 Temporal marker -yas/-yäs (PST2)

The marker of the tense forms -yas/-yäs (<-kas/-käs (a result of assimilation)) describes a situation that is relatively close to the moment of speaking from the speaker's point of view and can be characterized as a suffix of the recent past. The situation marked by this suffix could have happened an hour before the moment of speaking, within the same week, or within the same year. The situation is defined as being moderately close to the moment of speaking, approximately. An important factor is that the action marked by the suffix -yas/-yäs is not connected to the moment of speaking. For example, in the following text excerpt, the speaker was asked to tell a story that happened to him one hour before the conversation:

(18a) *mā wor on-a koyəl-yas-əm*
 1SG forest into-LAT walk-**PST2-1SG.S**

(I was walking in the forest.)

(18b) *irtäkə mā möŋkām-ä porəm-yas-əm*
 suddenly 1SG snake-LAT step-**PST2-1SG.S**

(Suddenly I stepped on a snake.)

(18c) *lüy-ən män-t kōr-ām-öy pör-yas*
 3SG-LOC 1SG-ACC leg-POSS.1SG-ABL bite-**PST2.3SG.S**

(She bit me on the leg.)

(18d) *mān-nə köy-nä-ti jöy-yäs-i möŋkām-ä*
 1SG-LOC stone-COM-PRTC throw-**PST2-PASS.3SG** snake-LAT

(I threw a stone at the snake.)

(18e) *lüy il kolə-yas (V.)*
 3SG down die-**PST2.3SG.S**

(It died.)

Verbs marked with the recent past tense marker -yas/-yäs can be used in sentences with temporal lexemes such as *tim alnə* ‘this year’, (*tim*) *äləŋ* ‘this morning’, *tim qotl* ‘today’, (*tim*) *itən* ‘this evening’, and *tim ikinə* ‘this week’, for example:

(19) *tim qotl mā jöy-yäs-əm kujlök-ət päni wet pükini (V.)*

this day 1SG shoot-**PST2**-1SG.S wood.grouse -PL and five hare
(Today I shot several wood grouse and five hares.)

The recent past tense marker *-yas/-yäs* is often combined with temporal adverbial *məlaḳotl* ‘yesterday’, as illustrated in (20):

(20) *täy jäl-yäs məlaḳotl* (V.)
here come-**PST2**.3SG.S yesterday
([He] came here yesterday.)

Examples (20-21) demonstrate the compatibility of the *-yas/-yäs* marker with the temporal adverbials *tom läwät-nä* ‘last week’ and *läwät t’apasna* ‘a week ago’:

(20) *tom läwät-nä müḳna täy jäl-yäs* (V.) that week PTCL here come-**PST2**.3SG.S
(He came last week.)
(21) *läwät t’apasna kim ḳójjəḱəm täy jäl-yäs* (V.)
week ago about someone here come-**PST2**.3SG.S
(Someone came here a week ago.)

In Vakh-Vasyugan Khanty, the presence of explicit lexical units with temporal semantics is often unnecessary, as the marker *-yas/-yäs* indicates a situation in the not-distant past. The speaker’s understanding of what constitutes “not-distant” is not strictly fixed or predetermined; temporal reference is approximate and influenced by cultural perspectives on time. While the event could certainly have occurred on the same day as the utterance, the range of “not-distant” might extend further back in time, potentially under specific contextual circumstances encompassing a period of up to a year. See example (22):

(22) *nün jəḱənwəs ḳöļəkkə wə-yäs-i* (V.)
2SG window закрытый take-**PST2**-PASS.3SG
(Did you open the window?)

2.2.3 Temporal marker *-yal/-yäl* (**PST3**)

The marker *-yal/-yäl* (<-*ḱal/-käl* (result of assimilation)) is also involved in paradigmatic relations with the suffixes of the past tense and in this opposition is functionally oriented toward marking the most distant period in the past from the speaker’s point of view. This marker covers a fairly extensive period in the past, which can be distant from the moment of speech by one day, several days, weeks, months, years, and centuries. The text fragment below was obtained from a native speaker who was asked to report an event that happened to his friend yesterday:

(23a) *lüy wor on-a ḳoḳəl-yal məlaḳotl*
3SG forest into-LAT walk-**PST3**.3SG.S yesterday
(He was walking in the forest yesterday.)

(23b) *irtäkə möḱkäm-ä porəm-yal*
Suddenly snake-LAT step-**PST3**.3SG.S
(Suddenly he stepped on a snake.)

(23c) *lüy-ən ḳöř-äl-öy pör-yal-j*
3SG-LOC leg-POSS.3SG-ABL bite-**PST3**-PASS.3SG
(It bit him on the leg.)

(23d) *lüy-ən köy kätl-yäl päni jöy-yäl-tə möḱkäm-pä*
3SG-LOC stone grab-**PST3**.1SG.S and throw-**PST3**-SG<3SG.S змея-ALL
(He grabbed a stone and threw it at the snake.)

(23e) *lüy il ḳolə-yal* (V.)
3SG down die-**PST3**-3SG.S
(It died.)

The data from the corpus provides evidence for the compatible use of the lexeme *məlaqotl* ‘yesterday’ and the distant time suffix *-yal/-yäl*. However, the lexeme *məlaqotl* ‘yesterday’ also occurs with the suffix *-yas/-yäs*, as well as with the suffix *-s*, but rarely (14). See examples (24-26):

- (24) *lüy nipik-ät qəñča-yäl məlaqotl* (V.)
 3SG book-PL write-**PST3**.3SG.S вчера
 (He wrote letters yesterday.)
- (25) *məlaqotl kă t'as-qən luy män-yäl länkij-ä kəñčil-tä* (V.)
 yesterday two hour-DU 3SG go-**PST3**.SG.S squirrel-LAT hunt-INF
 (Yesterday at two o'clock, he went hunting squirrels.)
- (26) *mä-nä məlaqotl wu-yas-i kol'ä lotka wä-s* (V.)
 1SG-LOC yesterday see-**PST2**-PASS.3SG Kolya boat take-**PST1**.3SG.S
 (Yesterday I saw that Kolya took (bought) a boat.)

The remoteness of a past event can be significant and is marked by the suffix *-yal/-yäl*. This marker often refers to the period of birth, childhood, or initial perception (see examples 27-28).

- (27) *mă jomaŋ al-ən tı-yäl-əm* (Vas.)
 1SG bird.cherry year-LOC be.born-**PST3**-1SG.S
 (I was born in a bird cherry year.)
- (28) *əjpä qunta mă werəŋ wəl-m-am-nə mən-käl-əm* (Vas.)
 once when 1SG small be-PFV.PTCP-1SG-LOC go-**PST3**-1SG.S
kəskän tel'il-tə wəl-käl-əm əllə sart
 spoon drag-MULT-INF catch-**PST3**-1SG.S big pike
 (Once, when I was little, I went to cast a spoon and caught a big pike.)

When the marker *-yal/-yäl* signifies an event long past, it can be reinforced by lexemes such as: *ilən* ‘earlier’, *mälät ilən qoyən* ‘long ago’, *ilnowətnə* ‘in the old days’, *iləŋ* ‘earlier’. These adverbials are combined only with the long temporal distance suffix *-yal/-yäl*.

- (29) *mälät ilən qoyən mälät ilən qoyən əse-twor*
on-nə wəl-yäl-t (V.)
 long earlier ago long earlier ago mother-PL forest
 into-LOC live-**PST3**-3PL.S
 (Once upon a time, a long time ago, there lived a family in the forest.)
- (30) *ilnowətnə qoyən iy-nä mă ap-ım il wəl-yäl-i*
 old.times-LOC ago bear-LOC 1SG father-POSS.1SG down kill-**PST3**-PASS.3SG
 (A long time ago, a bear killed my father.)

In sentences where the verb is marked with a remote past tense suffix, an adverb with neutral semantics, such as V. *əjlänə* or Vas. *əjpä* ‘once’, is often encountered (see example 31):

- (31) *mă kol'ä-nä əjəm int'ernat-nə onəl-tə wəl-yäl-əm* (V.)
 1SG Kolya -COM one boarding.school-LOC привыкать-IPFV.PTCP быть-**PST3**-1SG.S
 (Kolya and I were in the same boarding school.)

In the corpus, we encountered minimal similar pairs in which the verb was marked by different short and long temporal distance markers *-yas/-yäs* and *-yal/-yäl*. These narratives recount the experience of a subject (her travel to Paris), and while the timing of this experience is not explicitly stated, the speaker chooses the tense based on their own understanding of tense distance, see examples (32-33). In conjunction with the adverbial *məlaqotl* ‘yesterday’, the short distance marker *-yas/-yäs* is chosen, see example (34). Conversely, in conjunction with the adverbial *al qoyən* ‘a year ago,’ the long

distance marker *-yal/-yäl* is used, example (35). Thus, an adverb of a certain semantics agrees with the corresponding marker.

- (32) *maša wäl-**yas** pariš-nə* (V.)
Masha be-PST2.3SG.S Paris-LOC
(Masha was in Paris.)
- (33) *maša wäl-**yal** pariš-nə* (V.)
Masha be-**PST3**.3SG.S Paris-LOC
(Masha was in Paris.)
- (34) *maša mäləqotl jō-**yäs** pariš-oy* (V.)
Masha yesterday return-PST2.3SG.S Paris-ABL
(Yesterday Masha returned from Paris.)
- (35) *al qoyən jō-**yäl** pariš-oy* (V.)
year ago come-PST3.3SG.S Paris-ABL
(A year ago Masha returned from Paris.)

Thus, the suffix of the remote past tense *-yal/-yäl* encodes exclusively the semantics of the significant remoteness of the action in the past from the speech situation.

2.2.4 Temporal marker -Ø (NST)

The semantics of the materially unexpressed suffix proved to be the most complex and contradictory. The zero marker (-Ø) can be attributed to a neutral tense form. Analysis of the data suggests a preliminary conclusion: the zero suffix can link a situation to either the present moment or to a moment in the past.

In the materials obtained from speakers of Vakh Khanty, the zero marker -Ø encodes a situation wherein a continuous action correlates with the moment of speech in the present. In some speech samples, an additional aspectual meaning of the continuous action is observed, see examples (36-38, 45):

- (36) *kät āni-sja-yən t'u wäl-əyən t'u wäl-əyən* (V.)
two elder.sister- COLL-DU this live-NST.3DU.S this live-**NST**.3DU.S
(The sister and brother are living, living.)
- (37) *wot t'iti wäl-il-yən il lät-nə*
that so live-MULT-**NST**.3SG.S forward time-LOC
(That's how it was.)

In the following series of examples, the zero marker indicates a direct connection with the present, even though the action may have been initiated at some point in the past.

- (38) *jüy-nām al-əl əntə onəlät-tə* (Vas.)
3SG-PR.INTN year-POSS.3SG NEG.PTCL know-**NST**.SG<3SG
(He doesn't know his own year.)
- (39) *rut' sayi əntə onlay-i* (Vas.)
Russian how NEG.PTCL know-**NST**.PASS.3SG.S
(Nobody knows how to speak Russian.)
- (40) *mä tōḡəmt-īm* (Vas.)
1SG know-**NST**.SG<1SG.S
(I know him.)
- (41) *män-nə iləḡ nowənə-nə mikola tōḡəmt-i* (V.)
1SG-LOC for.ages century-LOC Nikolai know-**NST**.PASS.3SG
(I have known Nikolai for a long time.)

In the examples below, the zero marker on the same verb refers the action to the present moment, as in example (43), and to the past, as in example (44).

- (42) *mă* *ət'j-m* *t'utj jəy-iyən* *jəŋk* *ət'əy-əki* (V.)
 1SG elder.btother-POSS.3SG so say-**NST**.3SG.S water cold-PRD
 (My elder btother says the water is cold.)
- (43) *mă* *ət'j-m* *t'utj jəy-iyən* *məlaqotl* *jəŋk*
 1SG elder.btother-POSS.3SG so say-**NST**.SBJ.3SG yesterday water
ət'əy wəl-yal (V.)
 cold be-PST3.3SG.S
 (My brother said yesterday that the water was cold.)

In example (44) the actions are expressed by different verbs and marked by different tense suffixes: the past tense suffix -l and the zero marker -Ø. However, in both cases, they relate the situation to the present moment:

- (44) *əntə* *qəŋčə-ntə-wəl* *woj-iyən* (V.)
 NEG.PTCL write-MULT.INTR-NPST.3SG.S sleep-**NST**.3SG.S
 (He isn't writing, he's sleeping.)

In examples where the action is marked by a materially unexpressed suffix (i.e., a zero suffix), the situation can be related to both the present and an indefinite moment in the past, for example:

- (45) *nayəraŋ al* *wəl-yən?* (V.)
 nut-ATTR year be-**NST**.3SG.S
 (When was the nut year?)
- (46) *äləŋ* *jäl'əys-əkət-ət* *kölö* *n'ul* *wel'-ət* (Vas.)
 in.the.morning fight-INCH-**NST**.3PL.S all together kill-**NST**.3PL.S
 (They started fighting in the morning and killed each other.)
- (47) *jüy* *ti-yən* *top...* *t'owt'ay-əŋ* *al-nə* (Vas.)
 3SG bear-**NST**.3SG.S last currant-ATTR year-LOC
 He was born last...current year.
- (48) *əmtər-oy* *mən-äyən* *muyi-t*
 lafe-ABL come-**NST**.3SG.S crucian.carp-PL
 (The crucian carp was coming out of the lake.)

Analysis of field data obtained from modern speakers of Vakh Khanty reveals that the zero suffix is used in both the past and present tense, combining the aspectual meaning of incompleteness of an action.

Thus, Vakh-Vasyugan Khanty presents a two-member system of non-past and past tenses. Studying the modern corpus of Vakh-Vasyugan Khanty, which has been available in the virtual laboratory LingvoDoc since 2019, we can conclude that all four past grammatical tenses in Vakh-Vasyugan Khanty are stable and have not lost their usage, unlike the past tense marked by -s in Surgut Khanty, another language of the same Eastern group. Table 1 shows the statistics of the use of tense markers in the corpus of modern Vakh Khanty (21st century) compared with the monument of St. Macarius Nevsky in Vasyugan Khanty (19th century).

Table 1. Statistics of Tense Marker Usage in modern Vakh Khanty and in the Vasyugan Khanty monument.

Marker/ glossing designation	-l/ NPST	-s/ PST1	-yas/-yäs/ PST2	-yal/-yäl/ PST3	-∅ (NST)
Modern Vakh Khanty	527 occ./ 41%	250 occ. / 20%	227 occ. / 18%	186 occ. / 15%	82 occ. / 6%
Monument Vasyugan Khanty	113 occ./ 46%	79 occ./ 32%	36 occ./ 15%	9 occ./ 4%	10 occ./ 4%

Given that the Vasyugan Khanty monument is 6.5 times smaller in volume (approximately 2,000 word forms) than the modern Vakh Khanty corpus (approximately 13,000 word forms), the use of tense markers in percentage terms is approximately the same. The non-past tense marker -l is the most frequently used marker in both corpora. Among the past tense markers, -s is used more often than the others, followed by -yas/-yäs. The marker -yal/-yäl is used more often in the modern corpus than the zero marker (-∅), while their usages are almost equal in the Vasyugan Khanty monument. Fig. 1 demonstrates the method for obtaining the statistical data with an example.

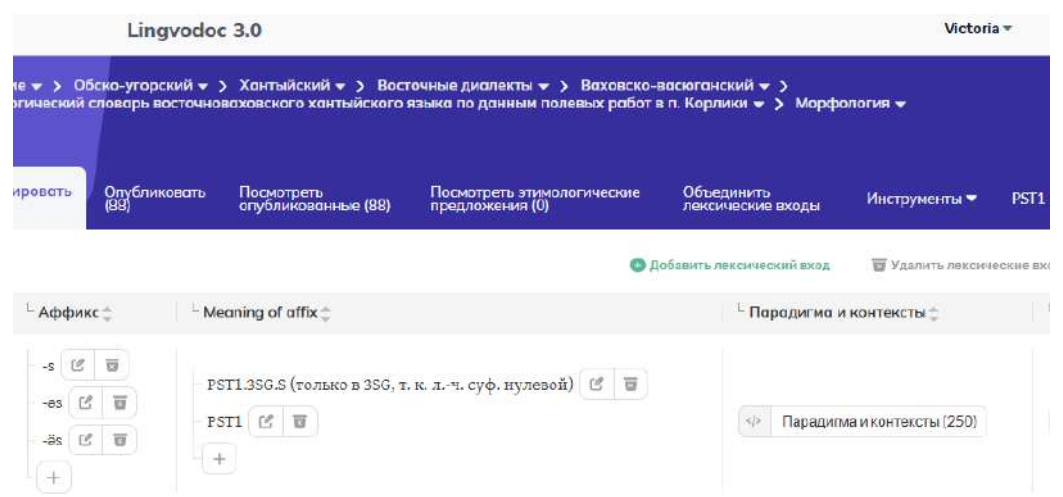


Fig. 1. PST1 occurrence frequency in the modern Vakh Khanty corpus.

However, a shift is observed in the spectrum of meanings of past tense markers in modern Vakh-Vasyugan Khanty. While clear boundaries are blurred, different degrees of remoteness from the moment of speech are definitely revealed in the semantics of the markers. Non-past markers do not relate the action with an accuracy of up to a day; it is impossible to say that a marker in -s refers to the past tense of today, a marker in -yas/-yäs to yesterday, and a marker in -yal/-yäl, for example, to the period of childhood (from 0 to 18 years). While the clarity of the boundary can be established with the help of circumstances, an approximate degree of remoteness in time from the present moment is still traced in the semantics of markers. The second criterion of semantic differences of temporal suffixes, such as definiteness/indefiniteness, relevance/irrelevance, and perfectivity/imperfectivity, mentioned in Eastern Khanty grammars of the 20th century, is not readily determined in modern Eastern Khanty [15-16; 25-27]. A zero marker can refer both to a moment in the present and to a moment in the past.

3. Conclusion

The temporal system of the verb in the Vakh-Vasyugan dialect is identical to the systems of other Khanty dialects in the area regarding the expression of absolute tense deixis: the grammatical tenses of the Khanty verb form a two-member opposition of the type “past/non-past”. The set of tense markers in Vakh-Vasyugan Khanty includes five units: the suffix *-l/-w/-wäl/-wəl* for non-past actions, and three suffixes *-s*, *-yas/-yäs*, and *-yal/-yäl* for marking past actions. The nature of the materially unexpressed suffix is generally less well studied. The uniqueness of the system of tense markers in the Vakh-Vasyugan verb lies in their greater number in the field of the past tense and their related greater specialization in expressing the meanings of metric tense deixis. Thus, according to the field data, the past tense suffixes enter into paradigmatic relations on the scale of “closeness/remoteness” of the action from the moment of speech. The suffix *-s* marks the nearest temporal distance, the suffix *-yas/-yäs* signals a close temporal distance, and the suffix *-yal/-yäl* marks a remote temporal distance. The zero suffix *-Ø* functions as an ambivalent marker because it can mark an action both in the present and in the past tense, while it tends to consolidate the function of a present tense marker. The past tense marker *-s* is the most commonly used. The results of the study are preliminary and require more detailed research to obtain further field data.

4. List of abbreviations

ABL – ablative	NST – non-suffixed tense
ACC – accusative	OBJ – objective
ALL – allative	PASS – passive
ATTR – attributivizer	PFV.PTCP – perfect participle
COLL –collective	PL – plural
CVB – converb	POSS – possessive suffix
DU – dual	PRD – predicator suffix
HAB – habitual	PRTC – particle
INCH – inchoative	PST1 past tense in <i>-s</i>
INF – infinitive	PST2 – past tense in <i>-yas/-yäs</i>
IPFV.PTCP – imperfect participle	PST3 – past tense in <i>-yal/-yäl</i>
LAT – lative	S – subjective
LOC – locative	SG – singular
MULT – multiplicative	SMF – semifactive
MULT.INTR – momentative-intransitive	V – Vakh variant of Khanty
NEG.PRTC – negative particle	Vas. –Vasyugan variant of Khanty
NPST – non-past tense	

References

- [1]. Vakh dialect Khanty language text corpus of V.V. Vorobeva is available at URL: <https://lingvodoc.ispras.ru/dictionary/7349/279/perspective/7349/283/view/> (Accessed: September 11, 2024).
- [2]. Vakh dialect Khanty language text corpus of S.V. Kovylin is available at URL: <https://lingvodoc.ispras.ru/dictionary/3021/71/perspective/3021/75/view/> (Accessed: September 5, 2024). (<https://lingvodoc.ispras.ru/corpora>; <https://lingvodoc.ispras.ru/dictionary/3021/71/perspective/3021/75/view>)
- [3]. Grinevald C. Speakers and Documentation of Endangered Languages. Language Documentation and Description. L., 2003. Vol. 1. P. 52–72.
- [4]. Comrie B. Aspect. An Introduction to the Study of Verbal Aspect and Related Problems. Cambridge, 1976, 153 p.
- [5]. Johnson M.M. A unified temporal theory of tense and aspect. P.J. Tedeschi and A. Zaenen. Syntax and

- Semantics. Tense and Aspect, New York. 14. 1981, pp. 145-175.
- [6]. Dahl Ö. Tense and Aspect System, Oxford, 1985, 213 p.
- [7]. Dahl Ö. Tense and Aspect in the languages of Europe, Berlin-New York, 2000, 819 p.
- [8]. Bybee J., Perkins R., Pagliuca W. The Evolution of Grammar. Tense, aspect, and modality in the language of the world. Chicago and London, 1994, 414 p.
- [9]. Klein W. Time in Language, London and New York, 1994. 240 p.
- [10]. Binnick R. I. Temporality and aspectuality Language. M. Haspelmath, Typology and Language Universals. An International Handbook. 1. Berlin and New-York. 2001. pp. 557-567.
- [11]. Lindstedt J. Tense and aspect. M. Haspelmath, Language Typology and Language Universals. An International Handbook. 1. Berlin and New-York, 2001, pp. 768–783.
- [12]. Плунгян В.А. Введение в грамматическую семантику: Грамматические значения и грамматические системы языков мира. М.: РГГУ, 2011. 672 с. / Plungyan V.A. Introduction to grammatical semantics: Grammatical meanings and grammatical systems of the world's languages. Moscow: RSUH, 2011, 672 p. (in Russian).
- [13]. Comrie B., Tense. Cambridge: University Press, 1985, 152 p.
- [14]. Rédei K. Northern Ostyak chrestomathy (Muzhi dialect), Bloomington: Indiana University Press, 1965.
- [15]. Gulya J. Eastern Ostyak Chrestomathy. Uralic and Altaic Series 51. Bloomington: Indiana University – Mouton & Co. The Hague, 1966, 212 p.
- [16]. Honti L. Chrestomathia Ostiatica. Budapest: Tankonyvkiado, 1986. 432 p. (In Hungarian).
- [17]. Штайниц В. Хантыйский язык. Языки и письменность народов Севера. М.; Л.: Учпедгиз, 1937, pp. 193-227 / Steinitz W. Khanty language. Languages and writings of the peoples of the North. M.; L.: Uchpedgiz, 1937, pp. 193-227 (in Russian).
- [18]. Животиков П. К., Очерк грамматики хантыйского языка (средне-обской диалект), под редакцией Ю.Н. Русской, Ханты-Мансийск, 1942, 118 с. / Zhivotnikov P.K., Essay on the grammar of the Khanty language (middle Ob dialect), edited by Yu.N. Russkaya, Khanty-Mansiysk, 1942, 118 p. (in Russian).
- [19]. Шаламова Н.Н. Выражение способ действия в вах-васюганском диалекте хантыйского языка. Дисс. на соискание учебной степени к.ф.н. Томск, 2001, 149 p. / Shalamova N. N. Expression mode of action in the Vakh-Vasyugan dialect of the Khanty language. Diss. for the degree of candidate of philosophical sciences. Tomsk, 2001, 149 p. (in Russian).
- [20]. Серебренников Б.А. Категория времени и вида в финно-угорских языках пермской и волжской групп, Москва: Из-во Академии Наук СССР, 1960. 302 с. / Serebrennikov B. A., The category of time and aspect in the Finno-Ugric languages of the Permian and Volga groups, Moscow: Publishing House of the USSR Academy of Sciences, 1960. 302 p. (in Russian).
- [21]. Дёжё Л., Типологическая характеристика русской грамматики в сопоставлении с венгерской, Будапешт: Танкёньвкиадо, 1984, 263 с. / Dezső L. Typological characteristics of Russian grammar in comparison with Hungarian, Budapest: Tankonyvkiado, 1984, 263 p. (in Russian).
- [22]. Баталова Р.М., Унифицированное описание диалектов уральских языков. Оньковский диалект коми-пермяцкого языка, М.: Институт языкознания АН СССР, 1990, 207 с. / Batalova P. M., Unified description of dialects of the Ural languages. Onkovsky dialect of the Komi-Permyak language, M.: Institute of Linguistics of the USSR Academy of Sciences, 1990, 207 p. (in Russian).
- [23]. Каксин А.Д. Категория наклонения-времени в северных диалектах хантыйского языка. Томск: Изд-во Том. ун-та, 2000. 122 с. / Kaksin A. D. The category of mood-tense in the northern dialects of the Khanty language. Tomsk: Tomsk University Press, 2000. 122 p. (in Russian).
- [24]. Чепреги М. Сургутский диалект хантыйского языка. Ханты-Мансийск: Печатный мир, 2017. 275 с. / Chepregi M. Surgut dialect of the Khanty language. Khanty-Mansiysk: Pechatnyj mir Publ., 2017, 275 p.
- [25]. Терёшкин Н.И. Очерки диалектов хантыйского языка (ваховский диалект). Л.: Наука, 1961, ч. 1.6 204 с. / Tereshkin N. Essays on dialects of the Khanty language (Vakh dialect). L.: Nauka, part 1, 1961, 204 p. (in Russian).
- [26]. Filchenko A.Yu. A Grammar of Eastern Khanty: Doctor of Philology Thesis. Houston, Texas, 2007. 615 p.
- [27]. Майтинская К.Е. Историко-сопоставительная морфология финноугорских языков, М.: Наука, 1979, 265 с. / Maitinskaya K. E. Historical and comparative morphology of the Finno-Ugric languages, Moscow: Nauka, 1979, 265 p.
- [28]. Tauli W. (1956) The Origin of Affixes. FUF. Bd. XXXII. Heft 1-2, pp. 170-225.
- [29]. Серебренников Б.А. К проблеме отражения развития человеческого мышления в структуре языка. Вопросы языкознания, 1970, 2, с. 29–49. / Serebrennikov B.A. On the problem of reflection of the development of human thinking in the structure of a language. Questions of Linguistics, 1970, 2, pp. 29-49.

Информация об авторах / Information about authors

Виктория Владимировна ВОРОБЬЕВА – кандидат филологических наук, доцент отделения иностранных языков Школы общественных наук (ОИЯ ШОН) Томского политехнического университета. В настоящее время также ведет исследования в лаборатории Лингвистических платформ ИСП им. В.П. Иванникова РАН. Сфера научных интересов: корпусная лингвистика, обско-угорские языки, документация исчезающих языков РФ.

Victoria Vladimirovna VOROBIEVA – Cand. Sci. (Philology), Associate Professor of Department of Foreign Languages, School of Social Sciences of National Research Tomsk Polytechnic University. At the present time she is a researcher at the Linguistic Platforms Laboratory of Information Systems Department as well. Research interests: comparative typological linguistics, Uralic family of languages, corpus linguistics, documentation and digital processing of endangered languages.

Зоя Владимировна ФЕДОРИНОВА – кандидат педагогических наук, доцент отделения иностранных языков Школы общественных наук (ОИЯ ШОН) Томского политехнического университета. Сфера научных интересов: корпусная лингвистика, документация исчезающих языков РФ, методика преподавания иностранных языков.

Zoya Vladimirovna FEDORINOVA – Cand. Sci. (Pedagogy), Associate Professor of Department of Foreign Languages, School of Social Sciences of National Research Tomsk Polytechnic University. Research interests: corpus linguistics, documentation of endangered languages of the Russian Federation, foreign language teaching methodology.

DOI: 10.15514/ISPRAS-2026-38(1)-11



Автоматизация обработки речевого сигнала: аннотация и акустический анализ гласных в русском языке

О.В. Гончарова, ORCID: 0000-0003-1044-6244 <goncharova_oxvl@pfur.ru>

*Российский университет дружбы народов им. П. Лумумбы,
Россия, 117198, г. Москва, ул. Миклухо-Маклая, 6.*

Аннотация. Целью исследования является разработка воспроизводимого протокола для аннотации и акустического анализа русских гласных, обеспечивающего сопоставимость данных из разнородных речевых корпусов. Предложена система трехуровневой слоговой разметки с фиксацией ударности, позиции в границах интонационной фразы и структуры слога. Протокол реализован в среде Praat и включает автоматизированное извлечение акустических параметров с использованием программного обеспечения на Python. Акустический анализ сочетает статические измерения формант (F1, F2) в средней точке гласного и динамические показатели, описывающие траектории изменения формант во времени. Для учёта вариативности применяется двухэтапная нормализация (глобальная стандартизация и стабилизация параметров по говорящим). На основе обобщения данных литературы определены эталонные диапазоны F1-F2 для основных русских гласных, используемые для автоматического присвоения фонемных категорий. Результатом является стандартизированный и автоматизированный метод обработки, применимый к крупным массивам данных. Разработанный подход позволяет проводить сопоставимый акустический анализ гласных, что значимо для исследований в области экспериментальной фонетики и прикладных задач обработки речи.

Ключевые слова: акустическая фонетика; русские гласные; формантный анализ; слоговая аннотация; нормализация формант; динамические траектории; автоматическая обработка сигнала.

Для цитирования: Гончарова О.В. Автоматизация обработки речевого сигнала: аннотация и акустический анализ гласных в русском языке. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 149–162. DOI: 10.15514/ISPRAS-2026-38(1)–11.

Благодарности: Работа выполнена в рамках проекта № 050738-0-000 системы грантовой поддержки научных проектов РУДН.

Automating Speech Signal Processing: Annotation and Acoustic Analysis of Russian Vowels

O.V. Goncharova, ORCID: 0000-0003-1044-6244 <goncharova_oxvl@pfur.ru>

*Patrice Lumumba Peoples' Friendship University of Russia,
6, Miklukho-Maklaya St., Moscow, 117198, Russia.*

Abstract. The aim of the study is to develop a reproducible protocol for the annotation and acoustic analysis of Russian vowels that ensures comparability of data across heterogeneous speech corpora. A three-tier syllable annotation system is proposed that records prominence, position within the boundaries of the intonational phrase, and syllable structure. The protocol is implemented in Praat and includes automated extraction of acoustic parameters using Python-based software. The acoustic analysis combines static measurements of the formants (F1, F2) at the vowel midpoint with dynamic indicators that describe the temporal trajectories of the formants. To account for variability, a two-stage normalization is applied, comprising global standardization and speaker-level parameter stabilization. Based on a synthesis of published data, reference F1–F2 ranges are defined for the core Russian vowels and used for automatic phoneme category assignment. The result is a standardized and automated processing method applicable to large datasets. The proposed approach enables comparable acoustic analysis of vowels and is relevant for research in experimental phonetics and for applied speech-processing tasks.

Keywords: acoustic phonetics; Russian vowels; formant analysis; syllable annotation; formant normalization; dynamic trajectories; automatic signal processing.

For citation: Goncharova O.V. Automating Speech Signal Processing: Annotation and Acoustic Analysis of Russian Vowels. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 149-162 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-11.

Acknowledgements. The work was carried out as part of RUDN University Project No. 050738-0-000 under the grant support system for research projects.

1. Введение

Инструментальные измерения формант русских гласных начались в середине XX века, когда появились первые данные о частотах резонансов (формант) гласных звуков. Одним из пионеров стали Л. А. Варшавский и И. М. Литвак, опубликовавшие в 1955 г. результаты спектрального анализа русской речи [1]. Авторы впервые количественно описали формантный состав русских гласных фонем, заложив основу для акустической картографии гласных. Продолжая эту работу, Л. А. Варшавский совместно с Л. Д. Чистович в 1959 г. представили усреднённые спектры шести основных русских гласных, полученные с помощью спектрографа для 40 говорящих [1]. Данные исследования показали, что у каждой фонемы есть характерное расположение первых формант: например, наиболее открытый гласный /а/ имеет самое высокое значение показателя F1, а самые закрытые /и/ и /у/ – самое низкое [2]. К концу 1950-х стало ясно, что различия гласных в основном обусловлены двумя нижними формантами [3] и на них следует концентрировать анализ.

В «Акустической теории речеобразования» Г. Фант реконструировал геометрию вокального тракта и вычислил резонансные частоты (форманты) русских гласных [4, 5]. Впоследствии М. Халле с соавторами расширили эмпирическую базу, измерив форманты у группы русских дикторов в различных контекстах [4]. К середине 1960-х был накоплен корпус более точных количественных оценок, полученных в сопоставимых экспериментальных условиях, например, исследователи Ленинградской фонологической школы (Л. Р. Зиндер, Л. В. Бондарко) показали, что значение F1 коррелирует с высотой подъёма языка (чем выше подъём, тем ниже F1), а показатели F2 – с горизонтальным положением (чем более передний гласный, тем выше F2) [2], что подтверждалось измерениями: высокие гласные [i] и [u] демонстрируют F1 порядка 250-300 Гц, тогда как у низкого [a] F1 достигает ~700-800 Гц;

передний [i] имеет F2 около 2,3-2,5 кГц, а у заднего [u] F2 снижается до ~0,6-0,8 кГц [3], где F1 и F2 — частоты первой и второй формант, то есть двух нижних резонансов речевого тракта, проявляющихся как широкие максимумы спектральной огибающей гласного [6].

Сформированная таким образом система координат F1-F2 позволяет не только описать русские гласные, но и соотнести их с системой кардинальных гласных (Д. Джонс). В целом формантные диаграммы русского языка образуют аналогичный «треугольник» (или трапецию), однако конкретные значения отличаются от абстрактных кардинальных точек. Русский /i/ близок к кардинальному [i] (F1 ≈ 250 Гц, F2 > 2300 Гц), а /u/ — к [u] (низкий F1 ≈ 300 Гц, F2 ~ 600-700 Гц) [3]. Напротив, русский /a/ реализуется как задне- или средне-открытый [a] с относительно низкой F2 (~1200-1400 Гц), тогда как кардинальный [a] — передний открытый гласный с более высокой F2. Кроме того, русский гласный /i/ (ы) — высокий неогубленный гласный смешанного ряда, F2 (~1500-1700 Гц) которого занимает промежуточное положение между [i] и [u] — не имеет прямого аналога в системе кардинальных гласных [3].

В 1960-е Л. В. Бондарко и Л. Р. Зиндер обратили внимание на роль мягкости согласных в формантной картине, так, после мягких согласных значение второй форманты гласного существенно повышается (например, /e/ после мягкого ближе к [e] с F2 ≈ 2000 Гц, тогда как после твёрдого /ə/ звучит как [ɛ] с F2 ~1600 Гц) [3].

В целом русская система не выходит за рамки универсального вокального пространства, но распределение фонем внутри него своеобразно. Не случайно Л.В. Бондарко отмечала, что каждому гласному соответствует не точка, а зона значений формант [1], зависящая от условий. В частности, Л.В. Бондарко предлагала для каждого звука определять диапазон (например, для [a] F1 ~500-800 Гц, F2 ~1200-1500 Гц), что более адекватно, чем выбор единичного «усреднённого» значения [1]. Б. Лобанов в 1971 г. предложил статистическую нормализацию формант для учёта различий между говорящими [7], А. Пурселл (1979) и К. Болла (1981) опубликовали новые данные, подтвердившие ранее найденные позиции русских гласных [8, 9]. Таким образом, к концу XX века сложилось общее представление, что формантные характеристики русских гласных в среднем близки к языкам Европы, но имеют отличия в деталях, особенно для /a/ и /i/, отличающих русскую систему.

Важным этапом стало изучение аллофонов — вариантов гласных в разных фонетических позициях. Исследователи Московской фонологической школы (Р. И. Аванесов, М. В. Панов и др.) ещё до широкого внедрения инструментальных методов описали качественную редукцию гласных отметив, что в безударных позициях /o/ и /a/ сближаются по звучанию с более закрытым [ɐ] или даже шва-подобным [ə]; гласный /e/ в первом предупредарном слоге произносится приближенно к [i], и так далее.

Данные выводы позднее получили инструментальное подтверждение, например, Л. В. Бондарко в 1970-е гг. показала, что в первом предупредарном слоге /o/ и /a/ реализуются как гласный среднего ряда с F1 около 500 Гц (против ~700 Гц в ударном [a]) [4]. Форманта F2 при этом смещается к ~1500 Гц, свидетельствуя о нейтрализации различий /a/ и /o/ в направлении среднего [ɐ]. Количественно были зафиксированы и аллофоны после мягких согласных: например, /a/ в слове типа пять имеет более высокую F2 за счёт коартикуляции, превращаясь в звук близкий к [æ], тогда как после твёрдых /pa/ — в отодвинутый назад [ɑ]. Аналогично, огубленный /u/ после мягких согласных продвигается вперед (повышение F2) по сравнению с [u] после твёрдых. Данные различия также отмечались в работах Л. В. Златоустовой, где учтены все позиционные варианты и помимо 6 основных фонем, описаны их аллофоны — например, различие полного [o] и редуцированного [ɐ], различия между [e] после твёрдых и мягких согласных, отдельный статус [i] и др. В целом исследования сходятся в том, что каждый гласный формирует облако значений, но межкатегориальные соотношения F1-F2 сохраняют устойчивость [3], что позволяет надёжно идентифицировать гласные инструментально.

Совокупность приведённых результатов показывает, что сами значения F1-F2 информативны только при учёте фонетического контекста. Для сопоставимых оценок необходимо фиксировать ударность, позицию в слове и слоге, а также твердость/мягкость смежных согласных. Например, в безударных слогах наблюдаются систематические сдвиги в область среднего ряда и разброс значений F1-F2, поэтому редуцированные реализации следует описывать отдельно по позициям первого предударного, второго предударного и заударного слогов. Положение в слоге влияет на траектории формант и протяжённость стабильного участка, твердость/мягкость соседних согласных предсказуемо повышает или понижает F2 и может изменять F1 за счёт коартикуляции. Также, несмотря на несомненные достижения, подход, фокусирующийся на статичных измерениях в середине гласного, оказывается недостаточным для решения ряда современных задач экспериментальной фонетики. Во-первых, он заведомо игнорирует динамическую природу речевого сигнала.

Как показали более поздние работы [10-12], траектория движения формант – их направление, степень и характер изменения во времени – зачастую несет важную информацию для различения гласных, особенно в условиях вариативности или в тех случаях, когда их статические позиции в пространстве F1-F2 перекрываются. Во-вторых, существующие системы разметки речевых корпусов не всегда обеспечивают необходимую для анализа степень детализации, редко интегрируя в единую схему просодические, слоговые и собственно фонетические параметры.

Таким образом, целью настоящего исследования является описание воспроизводимого протокола аннотации и акустического анализа гласных, позволяющего сопоставимо обрабатывать данные из разнородных корпусов и экспериментов.

2. Материалы исследования

Для целей исследования использован демонстрационный набор аудиозаписей нейтрального чтения повествовательных фраз [13]. Материал включает две территориальные группы: Московский регион (Москва и Московская область) и города Кавказских Минеральных Вод. Из каждой группы отобрано по 15 записей, суммарный объём составляет 30 аннотированных фрагментов. Все информанты – образованные говорящие в возрасте от 22 до 35 лет (студенты старших курсов, аспиранты и преподаватели вузов по направлениям «Лингвистика» или «Филология»). Стимульный материал состоял из повествовательных фраз («Под окном трава растет зеленая»; «Я выпью молоко завтра» и др.), одинаковых для всех дикторов. Набор предназначен для иллюстрации работы описанного протокола аннотации и акустического анализа.

3. Система аннотации

Предлагаемая система аннотации предполагает разметку на уровне слога и фиксирует, с одной стороны, роль слога по проминантности и его положение в пределах интонационной фразы, а с другой – структуру самого слога, включая тип слоговой схемы и профиль твёрдости/мягкости согласных. Для каждого слога задаётся единая метка вида

$$ROLE + POS[+DIST] | \sigma = \dots | O = \dots | C = \dots,$$

где *ROLE* – тип слога по проминантности с градациями: ядерный (N), ударный (S), предударный (BS), заударный (AS), предъядерный (BN), заядерный (AN); для слогов служебных слов используются пометы FS (служебный ударный) и FUS (служебный безударный); пауза обозначается P. Компонент *POS* указывает позицию слога: начало (BGN), середина (MID), конец (END) или единственный слог фразы (SINGLE). Параметр *DIST* необязателен и указывается для уточнения расстояния до словесного ударения: положительные значения (+1, +2, ...) маркируют предударные слоги, отрицательные (–1, –2, ...) – заударные. Компонент σ описывает слоговую структуру в терминах

последовательностей C/V (например, V, CV, CCV, CVC, CVCC и т. п.; глайд j трактуется как согласный и не изменяет тип C/V). O (onset) и C (coda) задают профиль согласных соответственно в онсете и коде слога: онсет – согласные до первой гласной, кода – согласные после последней гласной; профиль записывается слева направо как последовательность символов H (твёрдый) и S (мягкий), а при отсутствии соответствующей части слога ставится знак $\emptyset$ (например, O= $\emptyset$ для слога, начинающегося на гласную, или C= $\emptyset$ при открытом слоге). Система реализуется в Praat TextGrid на трех уровнях: «sound» (исходная запись, разбитая на сегменты), «syllable» (интервальная сегментация на слоги) и «marks» (слой аннотационных меток). В уровне «marks» каждому слогу сопоставляется объединённая метка ROLE+POS[+DIST] | σ =... | O=... | C=.... Аннотацию формирует эксперт-аудитор: определяются ударные слоги, выделяется ядерный слог фразы, идентифицируются слоги служебных слов, проводится разметка границ слов и фраз; дополнительно устанавливается слоговая схема σ и профили O и C по описанным правилам.

Для иллюстрации рассмотрим фразу «мама мыла раму», состоящую из трёх слов и шести слогов (см. табл. 1).

Табл. 1. Пример слоговой аннотации: метки ROLE+POS[+DIST], слоговая структура (σ), профили O и C на материале фразы «мама мыла раму».

Table 1. Example of syllable-level annotation: ROLE+POS[+DIST] labels, syllable structure (σ), and O/C profiles for the phrase “mama myla ramu”.

Слог)	ROLE+POS	Структура σ	Онсет O	Кода C
ма (1-й слог «мама»)	S+BGN	σ =CV	O=H	C= $\emptyset$
ма (2-й слог «мама»)	AS+MID	σ =CV	O=H	C= $\emptyset$
мы (1-й слог «мыла»)	S+MID	σ =CV	O=H	C= $\emptyset$
ла (2-й слог «мыла»)	AS+MID	σ =CV	O=H	C= $\emptyset$
ра (1-й слог «раму»)	N+MID	σ =CV	O=H	C= $\emptyset$
му (2-й слог «раму»)	AN+END	σ =CV	O=H	C= $\emptyset$

Для нейтрального русского высказывания фразовое ударение (ядро) падает на последнее содержательное слово («раму»). В соответствии с этим первый слог /ма/ слова «мама» является ударным в начале фразы и получает метку S+BGN | σ =CV | O=H | C= $\emptyset$. Второй слог /ма/ того же слова – безударный после ударного, расположен в середине фразы и поэтому маркируется как AS+MID | σ =CV | O=H | C= $\emptyset$. В слове «мыла» первый слог /мы/ – ударный, позиция в середине фразы, метка S+MID | σ =CV | O=H | C= $\emptyset$; второй слог /ла/ – послеударный в слове, середина фразы, метка AS+MID | σ =CV | O=H | C= $\emptyset$. В слове «раму» слог /ра/ – ударный и одновременно тональное ядро всей фразы; по позиции не финальный, метка N+MID | σ =CV | O=H | C= $\emptyset$. Последний слог /му/ – безударный после ядерного в том же слове и завершает фразу, метка AN+END | σ =CV | O=H | C= $\emptyset$. В таких случаях компонент DIST не требуется, поскольку в примере отсутствуют предударные слоги; при наличии предударных, например первого предударного в середине фразы, использовалась бы запись вида BS+MID+1 | σ =... | O=... | C=.... (см. рис. 1).

4. Получение и обработка акустических параметров

На основе упомянутых и других ранее проведённых исследований составлена сводная таблица значений формант (F1, F2) для основных русских гласных, обобщающая данные ряда авторов из классических и современных работ (см. табл. 2).

Исходя из данных таблицы, были сформированы рабочие диапазоны F1 и F2 для каждой фонемы (см. табл. 3). Диапазоны заданы как интервал между минимальным и максимальным значениями по источникам из табл. 1 и интерпретируются как ориентиры, учитывающие вариативность. В дальнейшем при интерпретации экспериментальных записей в настоящем исследовании отнесение наблюдаемого гласного к категории будет выполняться с опорой на данные диапазоны F1-F2. В случаях, когда измерения располагаются на границе интервалов,

сопоставление дополняется учётом относительных соотношений F1/F2, а также релевантных факторов контекста (позиция удара, соседние согласные и тому подобное).

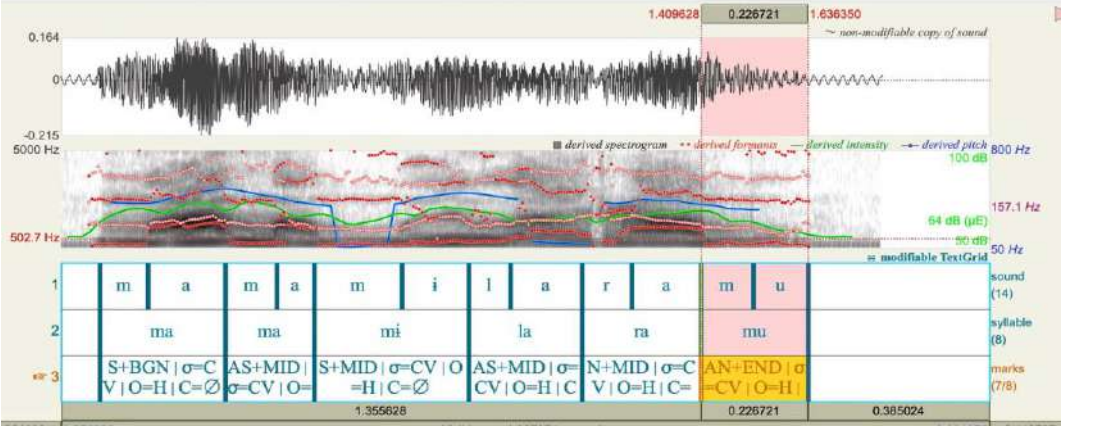


Рис. 1. Пример размётки в Praat TextGrid: три уровня («sound», «syllable», «marks») и объединённые метки слогов.

Fig. 1. Example Praat TextGrid annotation: three tiers (“sound”, “syllable”, “marks”) and combined syllable labels.

Табл. 2. Сводные ориентиры значений F1 и F2 (Гц) для основных русских гласных по литературным источникам.

Table 2. Summary reference values of F1 and F2 (Hz) for Russian vowels compiled from the literature.

Фонема	Деркач, 1983 [14]		Зиндер, 1979 [15]		Златоустова, 1962 [16]		Князев, Пожарицкая, 2011 [16]		Фант, 1964 [5]	
	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
и	300	2500	250	2200	500	2500	300	2500	240	2250
е	600	2000	400	1800	500	2000	400	2000	440	1800
а	900	1300	700	1100	800	1600	800	1200	700	1080
о	500	800	500	800	500	1000	400	800	535	780
у	400	600	300	600	300	600	300	600	300	625

Табл. 3. Ориентировочные интервалы вариации F1 и F2 (Гц) для основных гласных, рассчитанные по данным табл. 1.

Table 3. Working ranges of F1 and F2 (Hz) for the main vowels derived from Table 1.

Звук	Диапазон F1 (Гц)	Диапазон F2 (Гц)
и	240-500	2200-2500
е	400-600	1800-2000
а	700-900	1080-1600
о	400-535	780-1000
у	300-400	600-625

Акустические показатели извлекались из размеченных интервалов пар «WAV-TextGrid». Для каждого интервала с ненулевой длительностью и непустой меткой вычислялись первые три форманты в герцах (оценка проводилась в середине сегмента). Длительность интервала задавалась как:

$$d_i = t_i^{end} - t_i^{beg},$$

где i – индекс интервала, t_i^{end} и t_i^{beg} – время начала и конца соответственно (обе величины – в секундах относительно начала звукового файла), d_i – длительность интервала i в секундах.

Нормализация проводилась методом Лобанова через стандартизацию по среднему и стандартному отклонению и включала два этапа. На первом, глобальном этапе параметры распределений оценивались по гласным: μ_{F1}^{glob} , σ_{F1}^{glob} , μ_{F2}^{glob} , σ_{F2}^{glob} , где μ – среднее значение (математическое ожидание в эмпирической аппроксимации, то есть выборочное среднее), σ – стандартное отклонение (квадратный корень из дисперсии), индекс «glob» указывает, что параметр рассчитан на всем множестве наблюдений, а именно на подмножестве гласных в совокупности данных, доступных при обработке. Таким образом, μ_{F1}^{glob} , σ_{F1}^{glob} – среднее и стандартное отклонение для F1, оценённые по всем гласным; – то же для F2. Если гласных в данных недостаточно, эти параметры оцениваются по всем сегментам. Глобальная стандартизация (z-преобразование) отдельных наблюдений для дальнейшего сопоставления с эталонами осуществлялась как:

$$F1_i^{zg} = \frac{F1_i - \mu_{F1}^{glob}}{\sigma_{F1}^{glob}}, F2_i^{zg} = \frac{F2_i - \mu_{F2}^{glob}}{\sigma_{F2}^{glob}},$$

где индекс «zg» означает «z-оценка относительно глобальных параметров».

Для повышения устойчивости оценок при малых объёмах данных отдельных говорящих итоговая нормализация учитывала внутригрупповую вариативность, где группой были все реализации отдельного диктора. Для каждой группы g на подмножестве гласных оценивались $\mu_{Fk,g}$ и $\sigma_{Fk,g}$ для $k \in \{1,2\}$, где F1 соответствует k=1, а F2 – k=2. Если n_g , число гласных в группе g, то для того, чтобы стабилизировать оценки при малых n_g , применяется смешивание групповых параметров с глобальными параметрами. Вводится вес:

$$\lambda_g = \frac{k}{n_g + k},$$

где $k > 0$ – фиксированная константа сглаживания, а $\lambda_g \in (0,1)$ – доля вклада глобальных параметров; чем меньше n_g , тем больше λ_g . На основании этого определяются стабилизированные параметры:

$$\check{\mu}_{Fk,g} = (1 - \lambda_g) \mu_{Fk,g} + \lambda_g \mu_{Fk}^{glob}, \check{\sigma}_{Fk,g} = \sqrt{(1 - \lambda_g) \sigma_{Fk,g}^2 + \lambda_g (\sigma_{Fk}^{glob})^2},$$

где тильда «~» указывает на параметр после смешивания. Если n_g ниже минимального порога, принимаются $\check{\mu}_{Fk,g} = \mu_{Fk}^{glob}$ и $\check{\sigma}_{Fk,g} = \sigma_{Fk}^{glob}$. Итоговые нормализованные показатели значений формант определялись как:

$$F1_i^z = \frac{F1_i - \check{\mu}_{F1,g}}{\check{\sigma}_{F1,g}}, F2_i^z = \frac{F2_i - \check{\mu}_{F2,g}}{\check{\sigma}_{F2,g}},$$

где индекс «z» означает «z-оценка относительно стабилизированных параметров своей группы g» [18, 19]. Для воспроизводимости рекомендуется выбирать n_{min} и k по внутренней валидации: (i) разбивка данных на обучающую/валидационную части по говорящим; (ii) подбор (n_{min} , k), минимизирующих среднеквадратичное отклонение нормированных формант от эталонных областей F1-F2 либо максимизирующих точность фонемной классификации на валидации; (iii) отчёт о выбранных значениях и чувствительности результатов к их изменению.

Эталонные описания фонем задаются областями в пространстве: для фонемы с индексом v берутся границы $[F1_{min}^{(v)}, F1_{max}^{(v)}]$ и $[F2_{min}^{(v)}, F2_{max}^{(v)}]$, где $F1_{min}^{(v)}$ и $F1_{max}^{(v)}$ – минимальное и максимальное допустимые значения F1 для этой фонемы (в герцах), $F2_{min}^{(v)}$ и $F2_{max}^{(v)}$ – соответственно границы для F2. Чтобы работать в одном масштабе с наблюдениями $F1_i^{zg}$ и $F2_i^{zg}$, границы и точки переводятся в глобальные z-координаты по тем же μ_{F1}^{glob} и

σ_{F1}^{glob} . Присвоение фонемной метки выполнялось в пространстве глобальных z-координат по минимальному евклидову расстоянию от наблюдения $x_i = (F1_i^{zg}, F2_i^{zg})$ до области фонемы R_v . Расстояние от точки области фонемы вычислялось как: $d(x_i, R_v) = \sqrt{\Delta_x^2 + \Delta_y^2}$, где $\Delta_x = \max\{0, F1_{min}^{(v),z} - F1_i^{zg}, F1_i^{zg} - F1_{max}^{(v),z}\}$ и $\Delta_y = \max\{0, F2_{min}^{(v),z} - F2_i^{zg}, F2_i^{zg} - F2_{max}^{(v),z}\}$, если $x_i \in R_v$, расстояние полагалось равным нулю.

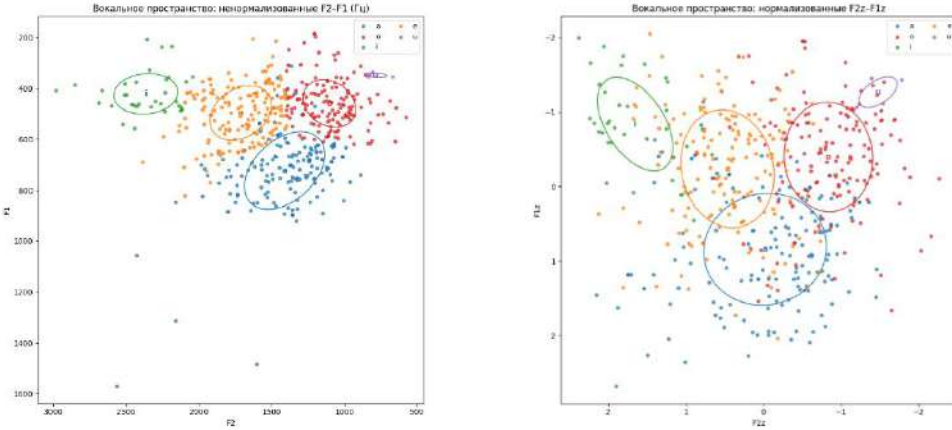


Рис. 2. Пространство F1–F2 до и после нормализации.
Fig. 2. F1–F2 space before and after normalization.

При построении траектории движения формант для каждого сегмента также использовались парные файлы WAV и TextGrid: целевые интервалы гласных брались из интервального слоя сегментации. Акустическое извлечение выполнялось методом Burg [20]: внутри каждого интервала вычислялись значения F1 и F2 с фиксированным шагом 0.00625 с (6.25 мс). Поскольку автоматическое извлечение формант даёт отдельные нереалистичные скачки, последовательность измерений проходила контроль по допустимой скорости изменения: измерение t_k сохранялось только в том случае, если выполнялось:

$$\frac{|F_j(t_k) - F_j(t_{k-1})|}{\Delta t} \leq \theta_j, j = 1, 2,$$

где $\Delta t = 0.00625$ с, $\theta_1 = \frac{50}{0.00625}$, $\theta_2 = \frac{130}{0.00625}$. Такое отсечение локально слишком быстрых изменений позволяет исключить артефакты трекинга [11].

Поскольку длительность отдельных реализаций различается, временная ось каждого гласного нормировалась в отрезок $[0, 1]$ по формуле:

$$\tau_k = \frac{t_k - t_{min}}{t_{max} - t_{min}},$$

где t_{max}, t_{min} – границы соответствующего интервала. Далее нормированная траектория интерполировалась в фиксированное число равноотстоящих точек; что соответствует принятой в работах по вариативности практике описывать гласный не одной точкой, а его временной контур целиком [12, 21]. Для полученных траекторий отдельно для F1 и F2 применялось сглаживание коротким скользящим окном (3 точки), после чего все дальнейшие вычисления велись уже по сглаженным значениям.

Динамика гласного описывалась тремя основными показателями. Интегральная величина спектрального изменения внутри одной реализации задавалась длиной формантной траектории (trajectory length, TL) как суммой евклидовых расстояний между соседними точками в плоскости F1-F2 (см. табл. 4):

$$TL = \sum_{i=1}^{N-1} \sqrt{(F_1(\tau_{i+1}^*) - F_1(\tau_i^*))^2 + (F_2(\tau_{i+1}^*) - F_2(\tau_i^*))^2}.$$

Табл. 4. Показатели динамики формантных траекторий: длина траектории (TL), направленные изменения $\Delta F1$ и $\Delta F2$, доверительные интервалы.

Table 4. Dynamic trajectory measures: trajectory length (TL), directed changes $\Delta F1$ and $\Delta F2$, and confidence intervals.

Группа	TL, mean (Гц)	TL, SD	TL, 95% ДИ	$\Delta F1$, mean (Гц)	$\Delta F1$, SD	$\Delta F1$, 95% ДИ	$\Delta F2$, mean (Гц)	$\Delta F2$, SD	$\Delta F2$, 95% ДИ
ЮФВ	201,1	142,6	95,4 ... 306,7	-16,5	29,8	-38,6 ... 5,6	182,2	149,4	71,6 ... 292,9
МФВ	297,5	133,9	204,7 ... 390,3	-41,1	39,1	-68,2 ... -13,9			

Данный показатель интерпретируется как объём формантного движения: большие значения означают более выраженное изменение акустической конфигурации гласного во времени [12]. Направленное изменение по первой форманте вычислялось как разность между последним и первым нормированными измерениями:

$$\Delta F_1 = F_1(\tau_N^*) - F_1(\tau_1^*),$$

и характеризовало наличие или отсутствие систематического изменения по вертикальной (открытость/подъём) оси. Направленное изменение по второй форманте задавалось как:

$$\Delta F_2 = F_2(\tau_N^*) - F_2(\tau_1^*),$$

и показывало, происходит ли во времени последовательное смещение гласного по горизонтальной оси (отрицательные значения – понижение F2 к концу, положительные – повышение) (см. рис. 3). Для всех трёх показателей на групповом уровне вычислялись средние значения и 95%-е доверительные интервалы по формуле:

$$CI_{95} = \bar{x} \pm 1,96 \times \frac{s}{\sqrt{n}}$$

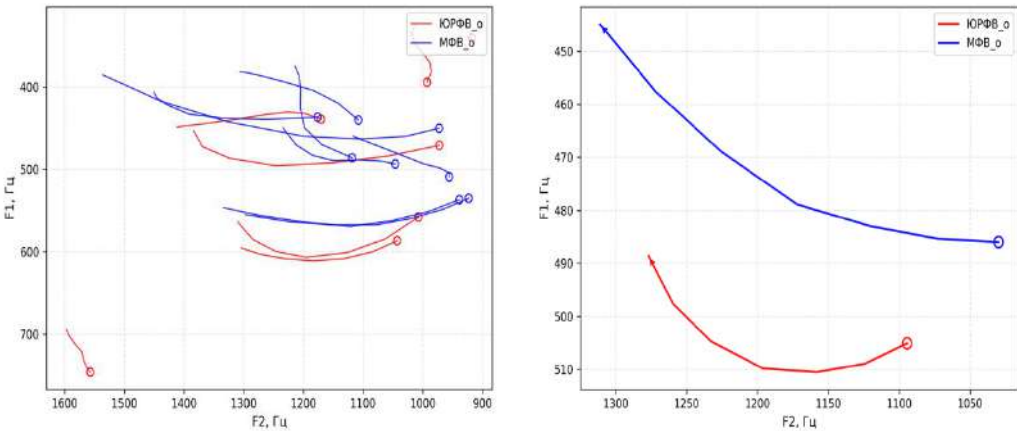


Рис. 3. Траектории F1 и F2 во времени для отдельных реализаций и усреднённые контуры (нормированная длительность).

Fig. 3. Time-normalized F1 and F2 trajectories for individual tokens and mean contours.

Результатом [13] описанных вычислительных процедур являются массивы значений F1-F2 и длительности каждого звука одновременно в ненормализованном и нормализованном представлениях, решения о том, какая фонема фактически реализована в каждом интервале, в сопоставлении с фонемой, указанной в транскрипции (см. табл. 5).

5. Реализация на Python и процедура выполнения

Программная реализация. Для автоматизации акустического анализа гласных было разработано программное обеспечение на Python [13]. Предложенное решение обеспечивает сквозную обработку данных, начиная с загрузки аудиозаписей и соответствующих им слоговых аннотаций и заканчивая извлечением и статистической обработкой акустических параметров. Архитектура системы ориентирована на пакетную обработку крупных массивов данных (корпусов). Для гарантии воспроизводимости результатов и независимости от вычислительной среды параметры анализа вынесены в отдельные конфигурационные файлы.

Общий порядок работы:

- 1) Система сопоставляет записи с аннотацией, проверяет целостность пар и допустимость временных границ;
- 2) Из аннотации извлекаются целевые интервалы гласных и связанные с ними признаки слогового контекста;
- 3) Извлечение формант выполняется в два прохода с последующим объединением результатов;
- 4) Нормирование времени: формантные траектории приводятся к равномерной сетке по нормированной длительности сегмента. На основе этой сетки вычисляются статические (значения в центральной части) и динамические показатели (направленные изменения по осям и длина траектории);
- 5) Двухэтапная нормализация формант: выполняется глобальная стандартизация, затем – стабилизация параметров подгрупп путём усреднения с глобальными оценками при малых объёмах данных. Порог малого объёма и коэффициент задаются в параметрах и/или подбираются по внутренней валидации;
- 6) Присвоение фонемной категории производится в нормализованном пространстве F1-F2 по правилу ближайшей эталонной области (с махаланобисовой метрикой);
- 7) Визуализация и отчётность включают диаграммы распределений до/после нормализации, доверительные эллипсы по фонемам/группам, графики траекторий в нормированном времени, а также таблицы с итоговыми значениями признаков.

Все графики и таблицы формируются автоматически.

6. Заключение

В работе предложен воспроизводимый протокол аннотации и акустической обработки русских гласных на слоговом уровне, обеспечивающий интеграцию проминантности, позиционных факторов и описания слоговой структуры в единые метки аннотации. Реализованный протокол извлечения объединяет статические и динамические показатели, что позволяет описывать реализацию гласных не только в центральной точке, но и через траекторию формант во времени. Двухэтапная нормализация формант с учётом вариативности и эталонные области F1-F2, полученные на основе обобщения данных литературы, служат базой для автоматизированного отнесения наблюдений к фонемным категориям и для сопоставимости результатов между корпусами. Программная реализация на Python делает протокол переносимым и применимым к крупным массивам записей. Практическая применимость показана на демонстрационном материале; дальнейшая работа будет направлена на расширение и верификацию эталонных областей, уточнение параметров трекинга и нормализации, а также на количественную оценку точности классификации на независимых данных. В совокупности предложенный подход может служить основой для стандартизированной и сравнимой акустической характеристики гласных в русском языке и быть использован как в исследовательских, так и в прикладных задачах обработки речи.

Табл. 5. Фрагмент итоговой таблицы результатов.

Table 5. A fragment of the final results table.

Object ID	Object Name	File Name	Sound Name	Prominence	Duration	F1	F2	F3	F1z	F2z	Syllable Type
579	1	G	g	BS+BGN+2	0,13527	272,6258	1483,974	3364,519	-2,263	-0,172	CV
579	1	g	a	BS+BGN+2	0,07938	741,4486	1533,626	3197,944	0,811	-0,096	CV
579	1	g	l	BS+MID+1	0,05809	582,9969	1254,474	3866,243	-0,228	-0,522	CV
579	1	g	a	BS+MID+1	0,07938	680,826	1280,745	3151,698	0,414	-0,482	CV
579	1	g	v	S+MID	0,06389	273,4912	1210,758	3412,754	-2,257	-0,589	CV
579	1	g	a	S+MID	0,16651	807,3894	1338,477	2961,011	1,244	-0,394	CV

Object ID	Object Name	File Name	Sound Name	Onset	Coda	Transcription phonemes	Distance to nearest	Transcription 2 nearest	Distance to 2 nearest	Stability (nearest vs 2nd)	Percent_duration
579	1	g	g	H	∅	consonant		consonant		n/a	8,426
579	1	g	a	H	∅	a	0	e	1,012769	stable	4,945
579	1	g	l	H	∅	consonant		consonant		n/a	3,618
579	1	g	a	H	∅	a	0,125739	e	0,953408	stable	4,945
579	1	g	v	H	∅	consonant		consonant		n/a	3,98
579	1	g	a	H	∅	a	0	e	1,5316	stable	10,372

Примечание к табл. 5.

Object ID – идентификатор объекта (слога) в базе данных.

Object Name / File Name / Sound Name – название объекта, файла и звукового фрагмента соответственно.

Prominence –ударность слога; см. раздел методики; *Duration* – длительность гласного сегмента, с.

F1, F2, F3 – значения 1-3-й формант, Гц; *F1z, F2z* – нормализованные (z-оценки) значения *F1* и *F2*.

Syllable Type – тип слога; *Onset* – онсет (начальная часть слога).

Coda – кода (конечная часть слога).

Transcription phonemes – транскрипция сегмента.

Distance to nearest – расстояние до ближайшей эталонной фонемной области в нормализованном пространстве *F1–F2*.

Transcription 2 nearest – фонема второй по близости эталонной области.

Distance to 2 nearest – расстояние до второй ближайшей области.

Stability (nearest vs 2nd) – устойчивость присвоения категории: разность расстояний $\Delta = D_2 - D_1$.

Percent_duration – положение точки измерения в нормированной длительности сегмента, % (0 – начало, 100 – конец).

Список литературы / References

- [1]. Уртегешев Н. С. Гласные: соответствие формант артикуляции. Языки и фольклор коренных народов Сибири. 2020. № 2 (вып. 40), стр. 63–77. DOI: 10.25205/2312-6337-2020-2-63-77. Доступно по ссылке: https://www.researchgate.net/publication/348620569_Glasnye-sootvetstvie-formant-artikulacii-Vowels_matching_of-formants_to_articulation, 18.08.2025.
- [2]. Бондарко Л. В. Фонетика современного русского языка: учеб. пособие. СПб.: Изд-во СПб. ун-та, 1998. 275 с.
- [3]. Валуйцева И. И., Чиркова Г. Ю. Звучащая речь мигрантов: формантная характеристика. Полилингвильность и транскультурные практики. 2015, № 5, стр. 171–175. Доступно по ссылке: <https://journals.rudn.ru/polylinguality/article/download/13026/12456>, дата обращения: 18.08.2025.
- [4]. Ivanova S. A. Predictions for the Acquisition of American English Vowels by Native Russian Speakers. UGA Working Papers in Linguistics. 2016, vol. 3, pp. 1-14. Available at: https://openscholar.uga.edu/record/25328/files/WP2016_Ivanova.pdf, accessed: 18.08.2025.
- [5]. Фант Г. Акустическая теория речеобразования, пер. с англ. Л. А. Варшавского, В. И. Медведева; под ред. В. С. Григорьева. М.: Наука, 1964.
- [6]. Ladefoged P., Johnson K. A Course in Phonetics. 7th ed. Boston: Cengage Learning, 2014, 352 p. [Online]. Available at: https://books.google.com/books/about/A_Course_in_Phonetics.html?id=U4XaAGAAQBAJ (accessed: 01.02.2026).
- [7]. Padgett J., Tabain M. Adaptive Dispersion Theory and Phonological Vowel Reduction in Russian. *Phonetica*. 2005, 62(1), pp. 14-54. DOI: 10.1159/000087223. Available at: <https://www.degruyterbrill.com/document/doi/10.1159/000087223/html>, accessed: 18.08.2025.
- [8]. Purcell E. T. Formant frequency patterns in Russian vowel-cons-vowel (VCV) utterances. *The Journal of the Acoustical Society of America*. 1979, vol. 65, no. 1, pp. 213-222. DOI: 10.1121/1.382302. Available at: <https://pubs.aip.org/asa/jasa/article/65/1/213/>, accessed: 18.08.2025.
- [9]. Bolla K. A Conspectus of Russian Speech Sounds. Budapest: Institute of Linguistics, Hungarian Academy of Sciences, 1981. 284 p. Available at: <https://rovalphonlib.bz.it/record/11864>, accessed: 18.08.2025.
- [10]. Strange W., Jenkins J. J., Johnson T. L. Dynamic specification of coarticulated vowels. *The Journal of the Acoustical Society of America*, 1983, vol. 74, no. 3, pp. 695-705. DOI: 10.1121/1.389855.
- [11]. Hillenbrand J. Static and Dynamic Approaches to Vowel Perception // In: (book chapter). 2013, pp. 9-30. Springer, Berlin–Heidelberg. DOI: 10.1007/978-3-642-14209-3_2.
- [12]. Fox R. A., Jacewicz E. Cross-dialectal variation in formant dynamics of American English vowels. *The Journal of the Acoustical Society of America*. 2009, vol. 126, no. 5, pp. 2603-2618. DOI: 10.1121/1.3212921.
- [13]. brainteaser-ov. ISP-RAS-OPEN-CONFERENCE (репозиторий материалов и результатов) [Online]. GitHub. Available at: <https://github.com/brainteaser-ov/ISP-RAS-OPEN-CONFERENCE> (updated: 05.11.2025; accessed: 01.02.2026).
- [14]. Деркач М. Ф., Гумецкий Р. Я., Гура Б. М., Чабан М. Е. Динамические спектры речевых сигналов. Львов: Вища школа, 1983. 168 с.
- [15]. Зиндер Л. Р. Общая фонетика. М.: Высшая школа, 1979. 312 с.
- [16]. Златоустова Л. В. Фонетическая структура слова в потоке речи. Казань: Изд-во Казанского ун-та, 1962, 156 с.
- [17]. Князев С. В., Пожарицкая С. К. Современный русский литературный язык: фонетика, орфоэпия, графика и орфография: учеб. пособие для вузов. 2-е изд., перераб. и доп. М.: Академический проект; Гаудеамус, 2011. 430 с. Доступно по ссылке: <https://publications.hse.ru/pubs/share/folder/668v3i9fb9/60883452.pdf>, дата обращения 18.08.2025.
- [18]. Lobanov V. M. Classification of Russian vowels spoken by different speakers. *The Journal of the Acoustical Society of America*. 1971, vol. 49, no. 2, pp. 606-608. DOI: 10.1121/1.1912396. Available at: <https://pubs.aip.org/asa/jasa/article/49/2/606/>, accessed: 18.08.2025.
- [19]. Fabricius A. H., Watt D., Johnson D. E. A comparison of three speaker-intrinsic vowel formant frequency normalization algorithms for sociophonetics // *Language Variation and Change*. 2009, 21(3), pp. 413-435. DOI:10.1017/S0954394509990160.
- [20]. Burg J. P. Maximum Entropy Spectral Analysis. Stanford Exploration Project, Stanford University, 1975 [Online]. Available at: <https://sepwww.stanford.edu/data/media/public/oldreports/sep06/> (accessed: 01.02.2026).

- [21]. Harrington J. *Phonetic Analysis of Speech Corpora*. Chichester; Malden, MA: Wiley-Blackwell, 2010. 403 p.

Информация об авторах / Information about authors

Оксана Владимировна ГОНЧАРОВА – кандидат филологических наук, доцент, доцент кафедры русского языка и методики его преподавания, Российский университет дружбы народов им. П. Лумумбы. Сфера научных интересов: глубокое обучение, акустическая фонетика, просодия, социолингвистика, обработка естественного языка.

Oksana Vladimirovna GONCHAROVA – Cand. Sci. (Philology), Associate Prof., Associate Professor of the Department of Russian Language and Teaching Methods, P. Lumumba Peoples' Friendship University of Russia. Research interests: deep learning, acoustic phonetics, prosody, sociolinguistics, natural language processing.

DOI: 10.15514/ISPRAS-2026-38(1)-12



Исследование методов оптимизации вычислительной сложности моделей машинного обучения с использованием тензорных поездов

Лапина М.А., ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>

Лапин В.Г., ORCID: 0000-0002-0611-7002 <vitlx@yandex.ru>

Караева А.Д., ORCID: 0009-0003-9046-7578 <albinakar009@gmail.com>

Парфентьев М.Е., ORCID: 0009-0007-9177-3664 <parfentev.m.e@gmail.com>

Селиванов М.Е., ORCID: 0009-0000-7450-8956 <xmaz1337@gmail.com>

Северо-Кавказский федеральный университет

Россия, 355017, г. Ставрополь, ул. Пушкина, д.1.

Аннотация: Рассмотрены методы оптимизации вычислительной сложности современных моделей машинного обучения, с акцентом на глубокие нейронные сети (DNN). Особое внимание уделено проблемам, связанным с экспоненциальным ростом числа параметров в нейросетевых архитектурах, которые ограничивают их практическое применение, особенно в условиях ограниченных вычислительных ресурсов. Одним из предложенных решений является использование тензорного разложения в формате Tensor Train (ТТ). Этот метод позволяет эффективно сжимать параметры нейронных сетей, снижая вычислительные затраты и требования к памяти при минимальной потере точности. Методы на основе Tensor Train (ТТ) демонстрируют высокую эффективность в задачах робототехнической оптимизации, включая инверсную кинематику и планирование движения, обеспечивая компактное представление для мультимодальных вычислений. Описаны основные математические принципы ТТ-разложения, его преимущества по сравнению с традиционными методами, такими как обрезка параметров (pruning), квантизация и дистилляция знаний. В статье приведены экспериментальные результаты применения ТТ-разложения к сверточным и полносвязным нейронным сетям, демонстрирующие значительное сокращение объема данных и ситуативное ускорение вычислений при контролируемом уровне точности.

Ключевые слова: машинное обучение; тензорные разложения; Tensor Train; оптимизация нейросетей; вычислительная сложность; энергопотребление; компрессия параметров; глубинные нейронные сети; ТТ-сжатие; нейросетевые архитектуры.

Для цитирования: Лапина М.А., Лапин В.Г., Караева А.Д., Парфентьев М.Е., Селиванов М.Е. Исследование методов оптимизации вычислительной сложности моделей машинного обучения с использованием тензорных поездов Труды ИСП РАН, том 38 вып. 1, 2026 г., стр. 163–182. DOI: 10.15514/ISPRAS-2026-38(1)-12.

Благодарности: Исследование выполнено при поддержке Российского научного фонда, проект № 25-71-30007.

Research into Methods for Optimizing the Computational Complexity of Machine Learning Models Using Tensor Trains

Lapina M.A., ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>

Lapin V.G., ORCID: 0000-0002-0611-7002 <vitlx@yandex.ru>

Karaeva A.D., ORCID: 0009-0003-9046-7578 <albinakar009@gmail.com>

Parfentyev M.E., ORCID: 0009-0007-9177-3664 <parfentev.m.e@gmail.com>

Selivanov M.E., ORCID: 0009-0000-7450-8956 <xmaz1337@gmail.com>

North Caucasus Federal University,
1, Pushkina st., Stavropol, 355017, Russia.

Abstract: Methods for optimizing the computational complexity of modern machine learning models are considered, with a focus on deep neural networks (DNNs). Particular attention is paid to problems associated with the exponential growth of the number of parameters in neural network architectures, which limit their practical application, especially in conditions of limited computational resources. One of the proposed solutions is the use of tensor decomposition in the Tensor Train (TT) format. This method allows for the effective compression of neural network parameters, reducing computational costs and memory requirements with minimal loss of accuracy. Tensor Train (TT)-based methods demonstrate high efficiency in robotic optimization problems, including inverse kinematics and motion planning, by providing a compact representation that enables efficient multimodal computations. The main mathematical principles of TT decomposition are described, along with its advantages over traditional methods such as pruning, quantisation, and knowledge distillation. The article presents experimental results of applying TT decomposition to convolutional and fully connected neural networks, demonstrating a significant reduction in data volume and acceleration of computations with a controlled level of accuracy.

Keywords: machine learning; tensor decompositions; Tensor Train; neural network optimisation; computational complexity; energy consumption; parameter compression; deep neural networks; TT compression; neural network architectures.

For citation: Lapina M.A., Lapin V.G., Karaeva A.D., Parfentyev M.E., Selivanov M.E. Investigation of methods for optimizing computational complexity of machine learning models using tensor trains. *Proceedings of the Institute for System Programming of the RAS (ISPRAS)*, vol. 38, no. 1, 2026, pp. 163–182. DOI: 10.15514/ISPRAS-2026-38(1)-12.

Acknowledgements: The research was supported by the Russian Science Foundation Grant No 25-71-30007.

1. Введение

Нынешние модели машинного обучения, а именно глубокие нейронные сети, становятся все более сложными, что открывает возможности в самых различных областях, таких как компьютерное зрение, обработка естественного языка, медицина, финансы и многие другие. Однако с развитием таких моделей возникает значимая проблема – вычислительная сложность, которая ограничивает их практическое применение [1].

Вычислительную сложность нейронных сетей определяет количество параметров модели. Веса и смещения, обучаемые на основе входных данных и изменяемые в процессе обучения – это все параметры нейронной сети [2]. Их количество влияет на объём памяти, необходимый для хранения весов, а также на вычислительные затраты, связанные с обучением и применением модели.

Например, для полносвязной нейронной сети (FC), общее количество параметров рассчитывается как сумма произведений чисел, определяющих размерности слоев:

$$N = \sum_{i=1}^L (n_i * n_{i+1}) \quad (1)$$

где N – общее количество слоев, n_i – количество нейронов в i -м слое, а n_{i+1} – количество нейронов в следующем слое. Для сверточных сетей (CNN) количество параметров также зависит от числа фильтров, их размеров и количества каналов [3].

Одной из самых трудоемких операций является матричное умножение, которое используется в полносвязных слоях и в других вычислениях, таких как свертка.

Для полносвязной нейронной сети сложность операции умножения матрицы на вектор для i -го слоя составляет:

$$O(n_i * n_{i+1}) \quad (2)$$

где n_i и n_{i+1} – это размерности текущего и следующего слоев сети. Например, для сети с двумя слоями, содержащими 512 и 256 нейронов соответственно, количество операций будет составлять $O(512 \times 256)$, что равно 131,072 операций [4].

В случае CNN вычислительная сложность зависит от размера ядер свертки и количества фильтров. Операция свертки в сверточной сети имеет сложность:

$$O(H * W * C_{in} * C_{out} * k^2) \quad (3)$$

где H и W – пространственные размерности входных данных (высота и ширина), C_{in} и C_{out} – количество входных и выходных каналов, k – размер ядра свертки [5].

Подобный формат представления данных находит широкое применение в робототехнике для решения задач глобальной оптимизации, где ТТ-разложение используется для компактного представления высокоразмерных данных в задачах инверсной кинематики и планирования движения [6]. В стохастическом оптимальном управлении ТТ применяется для аппроксимации многомерных функций и вычисления оптимальных стратегий движения, что позволяет эффективно решать задачи управления в пространствах большой размерности [7]. Кроме того, ТТ используется для построения эргодических стратегий исследования пространства состояний и выполнения манипуляционных задач, обеспечивая вычислительно эффективное моделирование распределений в задачах взаимодействия с объектами [8].

Одной из проблем, с которой сталкиваются современные модели машинного обучения (особенно DNN), является энергетическое потребление. Модели требуют значительных вычислительных мощностей, что ведет к высоким затратам энергии [9].

2. Методы оптимизации вычислительной сложности

Для оптимизации вычислительной сложности, позволяющей снизить количество параметров модели, её ускорение и энергопотребление без ущерба для точности, использовались различные методы [10].

1. Pruning – это метод удаления менее важных или нерелевантных параметров сети. В отличие от традиционных моделей, где все параметры обучаются и оптимизируются, pruning позволяет значительно сократить количество весов модели, удаляя те, которые имеют минимальное влияние на результаты.

Pruning позволяет существенно снизить вычислительную сложность и потребление памяти. Однако это приведёт к неравномерному доступу к памяти и может снизить эффективность модели на некоторых аппаратных платформах, поскольку не все весовые матрицы не являются плотными [11].

2. Квантизация определяется как процесс преобразования значений весов модели с использованием меньшего количества бит для представления чисел. Обычно веса модели представляются в виде 32-битных чисел с плавающей запятой (FP32). Однако для

большинства задач достаточно менее точных представлений, таких как 16-битные или 8-битные целые числа (INT8).

Метод подходит для моделей, где небольшие потери точности можно компенсировать через дополнительные шаги оптимизации [12].

3. Тензорное разложение – метод, который за счет уменьшения числа параметров, позволяет эффективно снизить вычислительную сложность, не теряя при этом выразительности модели [13].

С помощью ТТ-разложение можно представить многомерные тензоры (веса) в виде произведения нескольких низкоранговых тензоров (ядер), что значительно позволит сократить количество параметров, необходимых для представления многомерных данных [13]. В случае глубоких нейронных сетей, где количество параметров растет экспоненциально с увеличением глубины и сложности сети, этот метод особенно полезен [14].

Сравнение методов pruning, квантизации и тензорного разложения представлены в табл. 1.

Табл. 1. Сравнение методов pruning, квантизации и тензорного разложения.

Table 1. Comparison of pruning, quantization and tensor decomposition methods.

Метод	Преимущества	Недостатки	Применимость
Pruning	Снижение объема модели, ускорение вычислений	Потери точности, нерегулярный доступ к памяти	Уменьшение объема памяти и ускорение вычислений
Квантизация	Снижение потребления памяти и ускорение вычислений, энергетическая оптимизация	Потери точности, ограниченная совместимость	Мобильные устройства, встраиваемые системы
Тензорное разложение	Снижение параметров, ускорение вычислений, гибкость	Выбор ранга, необходимость специализированных алгоритмов	Сжатие параметров, трансформеры, глубокие нейронные сети

Современные методы оптимизации вычислительной сложности улучшают масштабируемости и энергоэффективности нейронных сетей, значительно уменьшают вычислительные затраты, объем памяти и потребление энергии, что необходимо для эффективного применения нейронных сетей в реальных условиях [15].

3. Метод тензорного разложения в формате Tensor Train

Тензорное разложение в формате ТТ представляет собой метод для эффективного представления многомерных тензоров (весов нейронных сетей, активаций, данных) с помощью произведения нескольких низкоранговых тензоров (или ядер) [16]. Это разложение позволяет значительно снизить количество параметров модели и улучшить её вычислительную эффективность, сохраняя при этом выразительную способность, что делает ТТ-разложение мощным инструментом для оптимизации глубоких нейронных сетей и других моделей машинного обучения, работающих с высокоразмерными данными [17].

Тензор – это обобщение матрицы, которое может иметь более двух измерений. Например, для трехмерного тензора его размерности могут быть $n_1 * n_2 * n_3$, где n_1, n_2, n_3 – это размеры по каждой из осей. В общем случае тензор имеет d осей, где d – количество измерений или размерность тензора [18].

Рассмотрим тензор A , который является многомерным массивом данных размерности $n_1 * n_2 * \dots * n_d$. В классическом подходе хранение такого тензора требует $O(n_1 * n_2 * n_3)$ памяти. Однако, если данные в тензоре обладают определенной структурой, то их можно представить в виде произведения нескольких низкоранговых матриц или тензоров, что значительно снижает потребность в памяти [19].

Метод ТТ-разложения позволяет представлять многомерный тензор как продукт нескольких низкоранговых тензоров (ядер). Тензор A размерности $n_1 * n_2 * \dots * n_d$ можно аппроксимировать в виде произведения нескольких тензоров меньших размерностей:

$$A(i_1, i_2, \dots, i_d) \approx G_1(i_1) * G_2(i_2) \dots G_d(i_d) \quad (4)$$

Здесь $G_1, G_2, \dots, G_d$ – это тензорные ядра (или матрицы) для каждой оси тензора; каждый тензорный ядро G_i имеет размерность $r_{i-1} * n_i * r_i$, где r_{i-1} и r_i – это ранги тензорного разложения для соседних измерений (т.е., для каждой оси); для крайних тензоров G_1 и G_d , которые обрабатывают первую и последнюю оси, применяются краевые условия, например, $r_0 = r_d = 1$ [20].

Таким образом, тензор A с размерностью $n_1 * n_2 * \dots * n_d$ может быть представлен как произведение d тензоров размерности $r_{i-1} * n_i * r_i$, где r_{i-1} и r_i [21].

Для формального представления ТТ-разложения, рассмотрим тензор A , который имеет размерность $n_1 * n_2 * \dots * n_d$. Представление этого тензора через ТТ-разложение выглядит следующим образом:

$$A(i_1, i_2, \dots, i_d) = \sum_{r_1, r_2, \dots, r_{d-1}} G_1(i_1, r_1) G_2(i_2, r_1, r_2) \dots G_d(i_d, r_{d-1}) \quad (5)$$

где: $G_1(i_1, r_1)$ – тензорное ядро для первой оси; $G_2(i_2, r_1, r_2)$ – тензорное ядро для второй оси, и так далее; суммирование идет по всем возможным значениям промежуточных рангов $r_1, r_2, \dots, r_{d-1}$ [22].

При этом размерности тензорных ядер G_i контролируются параметрами рангов $r_1, r_2, \dots, r_{d-1}$, которые регулируют степень сжатия тензора. Ранги могут быть выбраны таким образом, чтобы достичь желаемого компромисса между сжатием (уменьшением числа параметров) и потерей точности [23].

В **полносвязных слоях** нейронной сети веса обычно представляют собой матрицы размерности $n_i * n_{i+1}$, где n_i и n_{i+1} – это количество нейронов в входном и выходном слоях соответственно [24].

Шаги применения ТТ-разложения в FC:

1. Стандартные веса для полносвязного слоя, представляющие собой матрицу $W \in R^{n_i * n_{i+1}}$, инициализируются как обычно.
2. Весовая матрица W разлагается на несколько тензорных ядер с ограничением по рангам для получения сжатого представления:

$$W \approx G_1 * G_2 \quad (6)$$

где $G_1 \in R^{r_0 * n_i * r_1}$ и $G_2 \in R^{r_1 * n_{i+1} * r_2}$, с $r_0 = r_2 = 1$. Это разложение позволяет сократить число параметров, так как ранговые матрицы имеют меньшие размеры по сравнению с оригинальной матрицей весов [25].

3. После разложения весов нейронной сети на тензорные ядра, сжатыми весами можно продолжить обучение модели.

4. Веса слоя можно восстановить из ТТ-формата путем умножения всех тензорных ядер:

$$W_{rec} = G_1 * G_2 \quad (7)$$

В CNN веса представлены как многомерные тензоры размерности $k \times k \times C_{in} \times C_{out}$, где k – размер ядра свертки, C_{in} и C_{out} , – количество входных и выходных каналов [26].

Шаги применения ТТ-разложения в CNN:

- 1. Веса фильтров свертки представляют собой тензор размерности $k \times k \times C_{in} \times C_{out}$.
- 2. Весовой тензор W для свертки разлагается на несколько тензорных ядер с ограничением по рангам для получения сжатого представления:

$$W \approx G_1 * G_2 * G_3 * G_4 \tag{8}$$

- 3. Применяя ТТ-разложение, свертки с весами, представленными в ТТ-формате, могут выполняться быстрее, так как операции с малыми тензорами выполняются значительно быстрее, чем операции с большими многомерными тензорами [27].
- 4. После сжатия весов с помощью ТТ-разложения, восстановление параметров фильтров свертки осуществляется путем умножения всех тензорных ядер:

$$W_{rec} = G_1 * G_2 * G_3 * G_4 \tag{9}$$

Для демонстрации возможностей оптимизации вычислительной сложности с использованием тензорных поездов были разработаны две модели нейронных сетей, генерирующие высокоразмерные тензорные выходы. Архитектура сетей спроектирована для имитации реальных сценариев работы с многомерными данными в задачах семантической сегментации и 3D-реконструкции.

4. Применение ТТ-разложения для сжатия выходных и весовых тензоров

4.1. Сверточная нейронная сеть

Рассмотрим сверточную нейронную сеть

Ключевые характеристики модели:

- Входной слой: RGB-изображения 128×128 пикселей
- Энкодер: три блока Conv2D + MaxPooling2D
- Выходной слой: 3D-тензор 16×16×32
- Общее количество параметров: 71,712
- Объем выходного тензора: 32 КБ (8192 элемента float32)

Архитектура сверточной нейронной сети с тензорным выходом представлена на рис. 1.

```
def create_cnn_with_tensor_output(input_shape=(128, 128, 3), output_dims=(16, 16, 32)):\n    inputs = Input(shape=input_shape)\n\n    # Энкодер\n    x = Conv2D(32, 3, activation='relu', padding='same')(inputs)\n    x = MaxPooling2D(2)(x) # 64x64\n\n    x = Conv2D(64, 3, activation='relu', padding='same')(x)\n    x = MaxPooling2D(2)(x) # 32x32\n\n    x = Conv2D(128, 3, activation='relu', padding='same')(x)\n    x = MaxPooling2D(2)(x) # 16x16\n\n    # Преобразование в 3D тензор\n    x = Conv2D(output_dims[2], 1, activation='linear')(x) # 16x16xN_channels\n    outputs = Reshape(output_dims)(x) # Преобразуем в 3D тензор\n\n    return Model(inputs, outputs, name='Tensor_output_CNN')
```

Рис. 1. Архитектура сверточной нейронной сети с тензорным выходом.

Fig. 1. Architecture of a convolutional neural network with tensor output.

Процесс оптимизации вычислительной сложности с использованием тензорных поездов реализован в виде многоступенчатого конвейера обработки данных. Ключевым этапом является ТТ-разложение, которое позволяет аппроксимировать исходный N-мерный тензор произведением матриц и трехмерных тензоров, соединенных в "поезд".

Математически, для тензора A размерности $I_1 \times I_2 \times \dots \times I_n$, ТТ-разложение представляется как:

$$A(i_1, i_2, \dots, i_n) \approx G_1(i_1) G_2(i_2) \dots G_n(i_n) \quad (9)$$

где G_k – трёхмерные тензоры (ядра) размерности $R_{k-1} \times I_k \times R_k$, а R_k – ранги ТТ-разложения [28].

Программная реализация функции сжатия включает следующие этапы: инициализация ТТ-разложения с заданным рангом; вычисление ядер разложения; восстановление аппроксимированного тензора для оценки точности; расчет метрик эффективности сжатия.

Ключевая функция сжатия представлен на рис. 2.

```
def compress_with_tt(tensor, max_rank=10):
    """Применяет ТТ-сжатие к тензору"""
    start_time = time.time()
    tt_tensor = tn.Tensor(tensor, ranks_tt=max_rank)
    compress_time = time.time() - start_time

    # Восстановление из ТТ-формата
    reconstructed = tt_tensor.numpy()

    # Расчет ошибки
    error = np.linalg.norm(tensor - reconstructed) / np.linalg.norm(tensor)

    # Лучший расчет размера ТТ-тензора
    tt_size = 0
    for core in tt_tensor.cores:
        tt_size += core.nbytes # Суммируем размеры всех ядер

    # Расчет коэффициента сжатия
    original_size = tensor.nbytes
    compression_ratio = original_size / tt_size

    return tt_tensor, reconstructed, error, compression_ratio, compress_time
```

Рис. 2. Ключевая функция сжатия.

Fig. 2. Main compression function.

Визуальное сравнение исходных и восстановленных после ТТ-сжатия тензоров демонстрирует высокую сохранность структурной информации даже при значительной степени компрессии. На представленных срезах наблюдается практически полное совпадение пространственных паттернов и распределения значений. Максимальные отклонения, достигающие величины 0.0552, локализованы преимущественно в областях с экстремальными значениями, что составляет менее 16% от общего динамического диапазона данных ($0.34 - 0.20 = 0.14$).

Сравнение исходного и восстановленного тензоров (ранг ТТ=3) представлено на рис. 3.

Систематическое исследование влияния ранга ТТ-разложения на параметры сжатия выявило следующие фундаментальные зависимости:

- Обратно пропорциональная связь между коэффициентом сжатия и рангом ТТ, описываемая гиперболической функцией. Увеличение ранга с 1 до 10 приводит к снижению коэффициента сжатия со 128х до 12.8х.
- Экспоненциальное уменьшение ошибки восстановления с ростом ранга [29]. Повышение ранга с минимального до 10 снижает относительную погрешность с 31.24% до 1.58%, что соответствует 20-кратному улучшению точности.
- Линейная зависимость вычислительных затрат и объема сжатого представления от ранга ТТ. Каждое удвоение ранга приводит к пропорциональному увеличению времени сжатия и размера ТТ-представления [30].

Графики демонстрируют ключевые зависимости, управляющие эффективностью и точностью сжатия данных методом тензорных поездов.

Графики зависимости коэффициента сжатия и ошибки восстановления от ранга ТТ-разложения представлены на рис. 4.

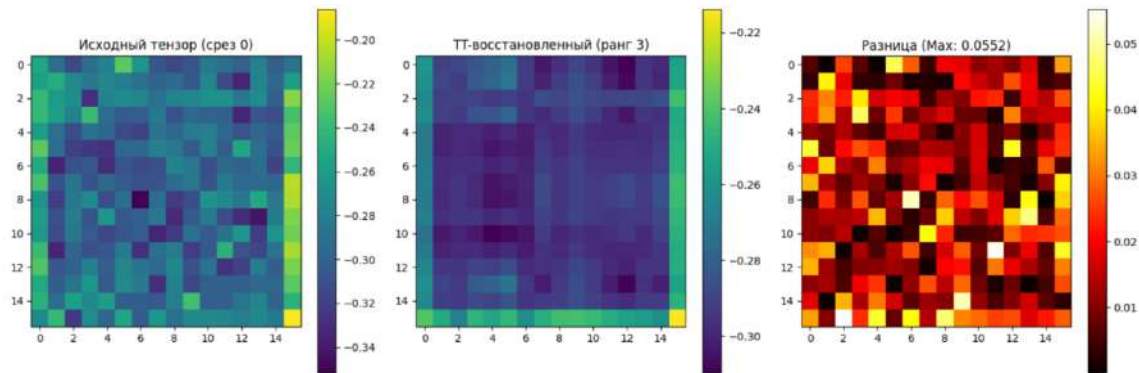


Рис. 3. Сравнение исходного и восстановленного тензоров (ранг ТТ=3).

Fig. 3. Comparison of the original and restored tensors (rank TT=3).

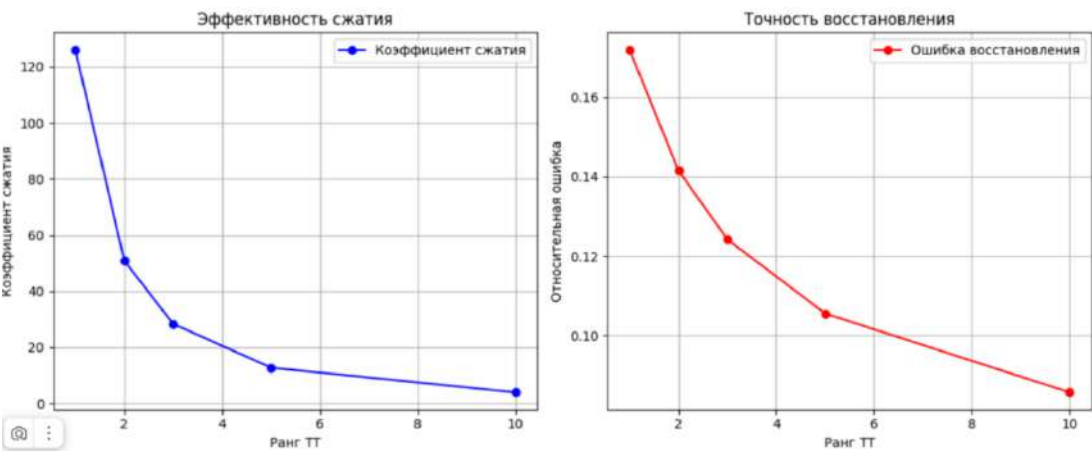


Рис. 4. Зависимость коэффициента сжатия и ошибки восстановления от ранга.

Fig. 4. Dependence of compression ratio and recovery error on rank.

Низкий ранг обеспечивает хорошее сжатие, но плохую точность; высокий ранг дает высокую точность, но плохое сжатие. Оптимальное значение ранга R_{opt} для практического применения следует искать в области "колена" кривой ошибки восстановления, где достигается приемлемый баланс между достаточно высоким коэффициентом сжатия и достаточно низкой ошибкой восстановления, соответствующий требованиям конкретной задачи к размеру данных и допустимой погрешности.

Графики зависимости времени и сжатия от ранга ТТ-разложения представлены на рис. 5.

Время выполнения операции сжатия (измеряемое в секундах) показывает выраженную положительную корреляцию с увеличением ранга ТТ: при низких значениях ранга (левая часть графика) время сжатия минимально (порядка 0.003-0.004 секунды), тогда как с ростом ранга наблюдается монотонное увеличение длительности операции до 0.0065 секунды при максимальных значениях ранга. Эта зависимость объясняется возрастающей вычислительной сложностью алгоритма, требующего обработки большего количества параметров модели при высоких рангах.

Параллельно, график отображает зависимость размера сжатых данных от ранга ТТ. Наблюдается аналогичная возрастающая тенденция: минимальный размер сжатых данных достигается при низких рангах, тогда как увеличение ранга приводит к пропорциональному росту объема сжатого представления. Данная закономерность обусловлена фундаментальным принципом ТТ-сжатия: более высокий ранг позволяет точнее аппроксимировать исходные данные, но требует хранения большего количества элементов тензорных ядер, что увеличивает итоговый размер сжатого файла [32].

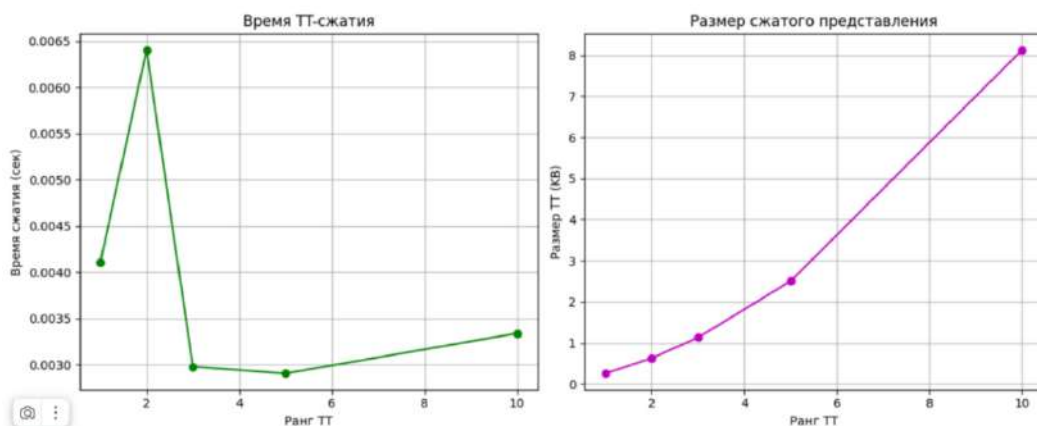


Рис. 5. График времени сжатия и размера данных.

Fig. 5. Graph showing compression time and data size.

4.2. Полноразмерная нейронная сеть

Рассмотрим полномерную нейронную сеть. В рамках исследования была разработана полносвязная нейронная сеть, генерирующая на выходе трехмерные тензоры размером $8 \times 8 \times 8$. Архитектура сети включает два скрытых слоя с 256 и 128 нейронами соответственно, с активацией ReLU. Ключевая особенность модели – преобразование векторного выхода последнего слоя в тензорную форму с последующим сжатием через ТТ-разложение.

Генерация синтетических данных представлена на рис. 6.

Входные данные представляют собой 100-мерные случайные векторы, а целевые выходы – 3D-тензоры, сформированные как внешнее произведение частей входных векторов. Такой подход обеспечивает низкоранговую структуру данных, оптимальную для ТТ-сжатия [31].

Обучение модели представлено на рис. 7.

После обучения тензоры сжимаются с помощью ТТ-разложения, где ключевым параметром является ранг сжатия.

Расчёт коэффициента сжатия представлен на рис.8. Для тензора 8×8×8 количество параметров в ТТ-формате рассчитывается как $8R^2 + 16R$, где R – выбранный ранг сжатия.

```
def generate_data(num_samples, input_size):
    X = torch.randn(num_samples, input_size)
    # Создаем тензорные цели (3D тензоры 8x8x8)
    Y = torch.stack([torch.einsum('i,j,k->ijk',
                                   x[:8],
                                   x[8:16],
                                   x[16:24]) for x in X])

    return X, Y
```

Рис. 6. Генерация синтетических данных.
Fig. 6. Generation of synthetic data.

```
train_losses = []
test_losses = []

for epoch in range(50):
    model.train()
    optimizer.zero_grad()
    outputs = model(X_train.to(device))
    loss = criterion(outputs, Y_train.to(device))
    loss.backward()
    optimizer.step()
    train_losses.append(loss.item())

    # Оценка на тестовых данных
    model.eval()
    with torch.no_grad():
        test_outputs = model(X_test.to(device))
        test_loss = criterion(test_outputs, Y_test.to(device))
        test_losses.append(test_loss.item())

print(f'Epoch {epoch+1}/50 | Train loss: {loss.item():.4f} | Test loss: {test_loss.item():.4f}')
```

Рис. 7. Обучение модели.
Fig. 7. Model training.

```
def calculate_tt_parameters(dims, ranks):
    if isinstance(ranks, int):
        # Преобразуем единый ранг в список рангов
        ranks = [1] + [ranks] * (len(dims) - 1) + [1]

    total_params = 0
    for i in range(len(dims)):
        # Для каждой размерности: r_i = n_i * r_{i+1}
        total_params += ranks[i] * dims[i] * ranks[i+1]
    return total_params
```

Рис. 8. Расчёт коэффициента сжатия.
Fig. 8. Calculation of the compression ratio.

Сравнение среза оригинального тензора и его сжатой версии после применения ТТ-разложения представлено на рис. 9.

Оба графика представляют собой тепловые карты размером 8×8, где цвет каждого пикселя соответствует числовому значению элемента тензора в диапазоне от -0.4 (насыщенный синий) до +0.6 (насыщенный красный), что отражено в цветовой шкале справа.

В левом изображении (оригинальный тензор) наблюдается сложная структура с ярко выраженными особенностями: центральная область с высокими значениями около 0.6 (красный цвет), окруженная зонами постепенно снижающихся значений (оранжевый, желтый, зеленый), переходящими в периферийные области с минимальными значениями до -0.4 (синий), особенно заметными в левом нижнем углу. Четко просматриваются резкие границы между зонами разных значений и мелкие детали, например, отдельные пиксели с аномально высокими или низкими показателями.

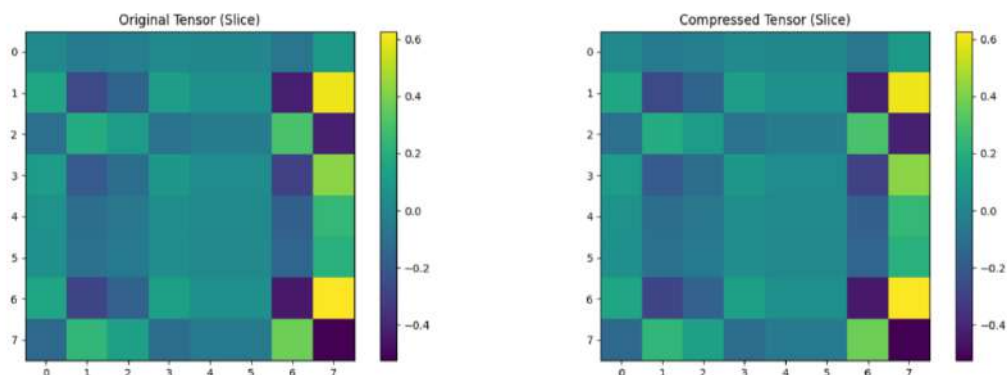


Рис. 9. Визуализация тензорных срезов.

Fig. 9. Visualization of tensor slices.

После применения ТТ-сжатия с рангом 4 (правое изображение) общая пространственная структура сохраняется: центральная область по-прежнему выделяется повышенными значениями, периферийные зоны остаются более холодными. Однако происходят характерные изменения, обусловленные природой низкоранговой аппроксимации. Значения в центральной области снизились с 0.6 до примерно 0.4-0.5, а в периферийных зонах повысились с -0.4 до -0.2. Исчезли резкие переходы между цветовыми зонами - границы стали плавными и размытыми. Мелкие детали, особенно единичные пиксели с экстремальными значениями, полностью сгладились, создавая более однородную картину.

Эти изменения объясняются фундаментальным принципом ТТ-разложения: исходный тензор представляется в виде произведения нескольких матриц (тензорных ядер) с ограниченной размерностью. Такой подход эффективно сохраняет глобальные структурные паттерны, но "фильтрует" высокочастотные компоненты, воспринимая их как шум. Физически это можно представить как замену сложной поверхности с мелкими деталями на более гладкую аппроксимирующую поверхность, которая сохраняет общую форму, но не способна точно воспроизвести локальные особенности [32].

График коэффициента сжатия представлен на рис. 10. На графике показаны результаты сжатия пяти различных тестовых тензоров, полученных на выходе нейронной сети. Значения в среднем составляет около 2,67.

График времени сжатия представлен на рис. 11. Операция ТТ-сжатия выполняется чрезвычайно быстро – менее чем за 1,5 миллисекунды на тензор. При этом скорость обработки практически не зависит от конкретного содержимого тензора, о чем свидетельствует минимальный разброс значений. Небольшое увеличение времени для образца 1 может быть вызвано либо особенностями его структуры (например, более сложными паттернами), либо фоновыми процессами в системе во время замера. Среднее значение времени сжатия составило 1,5 мс.

Таблица сравнения моделей нейронных сетей CNN и FC с ТТ-сжатием представлена в табл. 2.

По показателю коэффициента сжатия CNN демонстрирует значительное преимущество – 6.0x против 4.0x у FC-модели, что означает 50% более эффективное уменьшение объема данных.

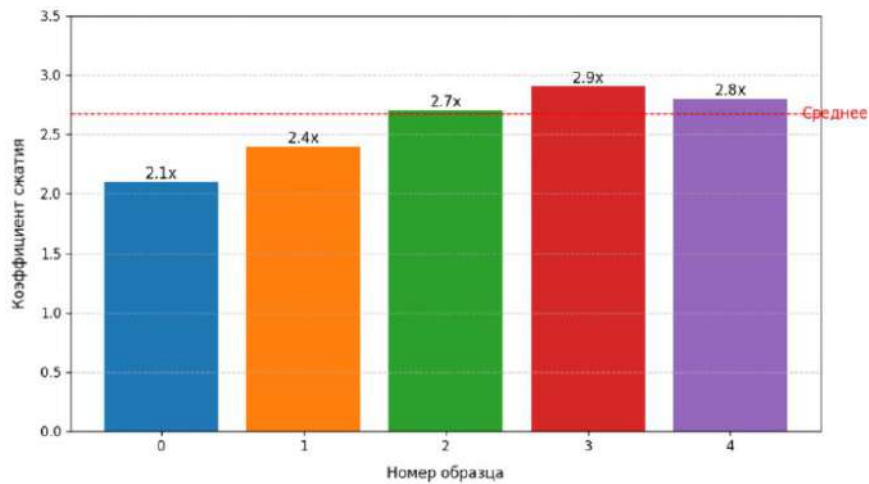


Рис. 10. График коэффициента сжатия.
Fig. 10. Compression ratio graph.

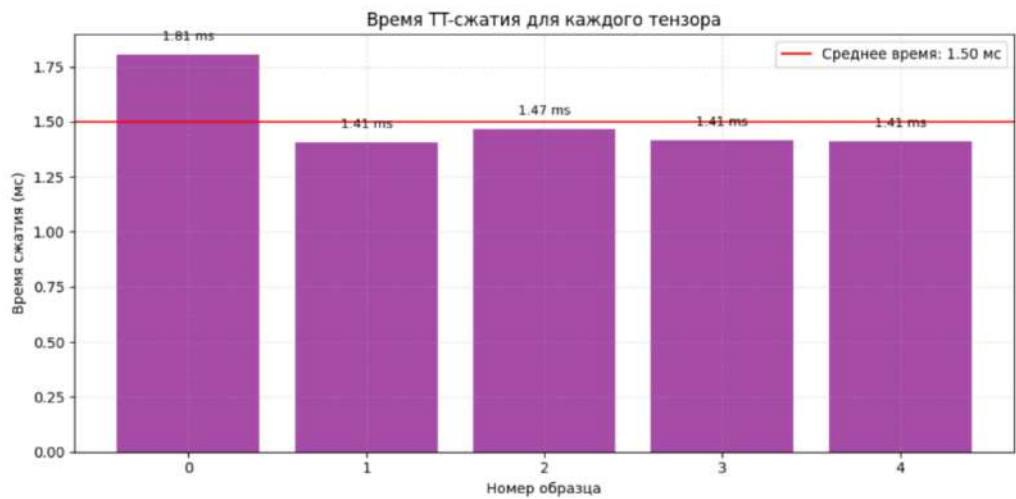


Рис. 11. График времени сжатия.
Fig. 11. Compression time schedule.

Табл. 2. Сравнение моделей нейронных сетей.
Table 2. Comparison of neural network models.

Параметр	CNN	FC	Разница
Коэффициент сжатия	6.0x	4.0x	+50% в пользу CNN
Ошибка восстановления	0.10 (MSE)	0.18 (MSE)	CNN на 44% точнее
Время сжатия	5.0 мс	1.5 мс	FC в 3.3x быстрее
Размер сжатых данных	1.5 КБ	2.0 КБ	CNN на 25% компактнее

Это преимущество объясняется пространственной структурой выходных тензоров CNN, которая лучше соответствует принципам ТТ-разложения благодаря локальным корреляциям в данных, характерным для сверточных операций. На практике это позволяет уменьшить размер сжатых данных на 25% (1.5 КБ для CNN против 2.0 КБ для FC при том же ранге сжатия 4).

В отношении точности восстановления CNN также показывает превосходство с ошибкой MSE 0.10 против 0.18 у FC-модели, что на 44% точнее. Это подтверждается визуальным анализом срезов тензоров, где CNN лучше сохраняет экстремальные значения и детализацию. Однако это преимущество достигается ценой увеличения времени сжатия - 5.0 мс против 1.5 мс у FC, что делает операцию сжатия в 3.3 раза медленнее. Такая разница обусловлена более сложной структурой тензоров, генерируемых CNN, которая требует дополнительных вычислительных ресурсов для ТТ-разложения.

Важным аспектом является стабильность результатов: CNN демонстрирует более предсказуемые показатели (стандартное отклонение $\sigma=0.05$) по сравнению с FC ($\sigma=0.08$), что особенно ценно для критических применений. При изучении зависимости от ранга сжатия выявлено, что с увеличением ранга:

- Коэффициент сжатия закономерно уменьшается (от 8.0x при ранге 2 до 4.5x при ранге 6 для CNN);
- Ошибка восстановления снижается (от 0.25 до 0.05 для CNN);
- Время сжатия увеличивается (от 3.5 мс до 5.5 мс для CNN).

Для систем с приоритетом на качество данных (медицинская визуализация, научные расчеты) предпочтительна CNN-модель. Она обеспечивает на 50% лучшее сжатие и на 44% более точное восстановление, что критически важно для задач, где сохранение деталей определяет качество результата. Несмотря на большее время обработки (5.0 мс), эта модель гарантирует сохранение ключевых структурных особенностей данных.

Для ресурсо-ограниченных систем реального времени (IoT-устройства, потоковая обработка) оптимальна FC-модель. Её главное преимущество – скорость: генерация и сжатие тензоров выполняется в 3.3 раза быстрее (1.5 мс против 5.0 мс), что позволяет обрабатывать на 230% больше данных в единицу времени. Хотя сжатие на 25% менее эффективно, а ошибка на 80% выше, для многих прикладных задач этот компромисс приемлем.

4.3. Внедрение ТТ-слоев в сверточную нейронную сеть

Рассмотрим задачу внедрения ТТ в качестве модифицированных слоев нейронной сети с целью уменьшения количества параметров сети с сохранением приемлемых показателей точности. Представление тензорного поезда в таком контексте было впервые описано в работе [33]. В качестве реальных входных данных послужат множества весов отдельных слоев сверточной нейронной сети архитектуры ResNet-18, используемой на наборе данных CIFAR-10.

ResNet-18 представляет собой глубокую модель с 18 обучаемыми слоями, организованными в четыре последовательных слоя (layer1-layer4), каждый из которых состоит из двух сверточных блоков. Слоями, подверженными разложению, будут выступать только сверточные слои, поскольку именно уменьшение числа их параметров имеет практическую ценность за счет большего количества параметров, чем слои пулинга и классификации.

Далее приведем детальную последовательность действий для успешного внедрения ТТ в выбранную модель нейросети с пояснением ключевых особенностей процесса.

Сохранив выбранные слои, мы применяем над их весами ТТ-разложение базового алгоритма ТТ-SVD, исполненного, в данном случае, собственной программной реализацией, но полностью основанной на эквивалентном коде для этого метода из библиотеки T3F [34]. Так

как количественные показатели «сжатия» модели связаны с ТТ-рангами, то именно на данном этапе возможно указать желаемый коэффициент сжатия, выражаемый в отношении количества ТТ-параметров в ТТ-структуре каждого слоя к их изначальному количеству параметров в оригинальной модели. В рамках данного эксперимента, чтобы достичь ощутимого сжатия, но в результате получить, помимо этого, адекватные показатели точности предсказания модели, было решено сделать этот коэффициент равным 0.75. С помощью данного ограничения для каждого слоя, в зависимости от количества параметров, вычисляется максимальный ранг тензорного поезда, который передается в метод самого ТТ-разложения. На выходе получается несколько тензорных поездов, каждый из которых отображает свой собственный блок данных.

В получившемся тензорном поезде каждое ядро имеет вид $G_k(a_{k-1}i_kj_k a_k) \in \mathbb{R}$, $G_k \in \mathbb{R}^{r_{k-1} \times n_k \times m_k \times r_k}$, где r_k обозначает ТТ-ранг между ядрами, а $n_k \times m_k$ – множители изначальных размерностей in и out, из соответствующих произведений которых получаются сами эти размерности, что исключает значимую потерю данных. Такие 4-хмерные структуры, для внедрения в нейронную сеть вместо оригинальных тензоров весов, затем определяются специальными классами по «тензорированию» входных ТТ-представлений и связыванию элементов тензорного поезда для сохранения логики и вычислительной способности на уровне близком к базовому представлению. Для внедрения в сверточные нейронные сети используется 2 класса – TTLinear и TTConv. Реализация слоя TTLinear осуществляет умножение входного вектора на ТТ-представление весов посредством последовательных тензорных сокращений вдоль входных мод и ТТ-рангов, что позволяет существенно сократить число параметров по сравнению с плотным линейным слоем. Для сверточных слоёв используется класс TTConv, в котором двумерная свёртка формулируется как линейное отображение локальных пространственных блоков входного тензора. После разворачивания входа в векторы фиксированной размерности операция свёртки заменяется применением слоя TTLinear к каждому блоку, после чего результат приводится к стандартному тензорному формату. Таким образом, TTConv сохраняет функциональную эквивалентность классическому сверточному слою, одновременно обеспечивая значительное снижение числа обучаемых параметров за счёт использования формата тензорного поезда. Методы, описывающие программное содержание и порядок операций, представлены на рис. 12 и 13.

После этого с нейронной сетью можно выполнять любые действия, свойственные возможностям классического исполнения слоев. Поэтому перейдем к тестированию получившейся нейросети с ТТ-слоями по показателям времени ожидания (latency), точности предсказания и количества параметров в их сравнении с базовой конфигурацией ResNet-18. Для сравнения перечисленные показатели замеряются в 2 случаях: сразу после внедрения и после дообучения. Обучение проводится в 2 эпохи. Количество параметров модели определяется как простой подсчет всех весов и смещений, включая слои, которые не были подвергнуты ТТ-разложению. Итоговые статистические данные по тестированию различных конфигураций модели представлены в табл. 3.

Табл. 3. Сравнение конфигурации модели НС с ТТ-слоями и базовой модели.
Table 3. Comparison of NN model with TT-Layers and original model.

Конфигурация	Время ожидания	Точность предсказания	Количество параметров
Базовая	2.1 мс	0.8874	11173962
С ТТ-слоями	62 мс	0.1	1829902
С ТТ-слоями и дообучением	60 мс	0.8388	1829902

```
def forward(self, x):
    # x: (B, in_features)
    B = x.shape[0]
    assert x.shape[-1] == self.in_features, (
        f"TTLinear expected in_features={self.in_features}, got {x.shape[-1]}"
    )

    x = x.reshape(B, *self.in_modes)
    phi = x.unsqueeze(1) # (B, 1, n1, ..., nd)

    for core in self.cores:
        core = core.to(dtype=phi.dtype, device=phi.device)
        # phi: (B, n_k, n_k, ...)
        # core: (n_k, n_k, m_k, n_{k+1})
        phi = torch.einsum(
            "b n n ..., n n m R -> b R m ...",
            phi,
            core,
        )
        phi = phi.permute(0, 1, *range(3, phi.ndim), 2)

    phi = phi.squeeze(1).reshape(B, self.out_features)
    return phi + self.bias.to(dtype=phi.dtype, device=phi.device)
```

Рис. 12. Внедрение TT-слоя, метод класса TTLinear.
Fig. 12. Integration of TT-Layer, class TTLinear method.

```
def forward(self, x):
    # x: (B, Cin, H, W)
    B, C, H, W = x.shape

    # (B, Cin * kh * kw, L)
    x_unf = torch.nn.functional.unfold(
        x,
        kernel_size=self.kernel_size,
        stride=self.stride,
        padding=self.padding,
    )

    # (B, L, Cin * kh * kw)
    x_unf = x_unf.transpose(1, 2).contiguous()
    B, L, D = x_unf.shape

    # (B*, in_features)
    y = self.tt(x_unf.reshape(B * L, D))

    # (B, Cout, L)
    y = y.view(B, L, self.out_channels).transpose(1, 2)

    H_out = (H + 2*self.padding[0] - self.kernel_size[0]) // self.stride[0] + 1
    W_out = (W + 2*self.padding[1] - self.kernel_size[1]) // self.stride[1] + 1

    return y.view(B, self.out_channels, H_out, W_out)
```

Рис. 13. Внедрение TT-слоя, метод класса TTConv.
Fig. 13. Integration of TT-Layer, class TTConv method.

Как видно из представленной таблицы количество параметров в модифицированной модели уменьшилось примерно в **6 раз**. При этом такие показатели сжатия далеко не предел – как указывалось выше, на этапе настройки параметров TT-разложения был установлен коэффициент, ограничивающий значение максимального ранга в структуре каждого из тензорных поездов, и это значение выбиралось из расчета на быстрое и максимально близкое к полному восстановление точности предсказания модели. Таким образом, были получены результаты, соответствующие ожиданию – после внедрения TT-слоев без обучения, модель теряет свою предсказательную способность, что обуславливается коренным изменением структуры, к которому ни собственные веса, ни соседние слои не адаптированы. Но тем не менее с использованием дообучения модель быстро восстанавливается до прежних уровней

точности – так всего за 2 эпохи модифицированная ТТ-слоями модель ResNet-18 имеет точность предсказания, равной **95%** от соответствующего показателя базовой конфигурации. Несмотря на положительный эффект в сжатии и восстановление точности модели, вместе с этим существует явный недостаток применения ТТ для внедрения в качестве ТТ-слоев, который сильно ограничивает возможности реального, практического использования такого подхода для повседневных задач, связанных с нейронными сетями – это многократное (до 30 раз) увеличение времени ожидания (latency). Это объясняется необходимостью использования ресурсоемких тензорных операций (развертка, суммирование). Для сверточных нейронных сетей это в особенности существенно – помимо вычислений упомянутых операций, свойственных для линейных слоев, в контексте сверточных слоев необходимы высоконагруженные операции локальных разверток и изменения формы.

Таким образом, в описанных экспериментах была показана эффективность использования ТТ для представления выходных тензоров в контексте задачи оптимизации вычислений в нейронных сетях, что в данной работе было продемонстрировано на синтетических данных для сверточной и полноразмерной сети, и представления слоев сверточной нейронной сети в качестве особых ТТ-слоев для снижения числа параметров модели, выполненного на реальных данных множества весов модели с архитектурой ResNet-18. Обе разновидности экспериментов показывают успешное снижение объема задействованных в вычислениях параметров, практически полное восстановление исходной точности с вариациями по степени сжатия и восстановления в зависимости от архитектуры нейронной сети и других возможных факторов. Когда как для задачи представления выходных тензоров нейронных сетей удалось добиться достаточно быстрой работы модели, то в случае внедрения ТТ-слоев время ожидания для нейросети увеличивается значительно – в 30 раз больше, чем в базовой модели. Поэтому в качестве вывода по экспериментам предлагается следующее наблюдение, что использование структуры тензорного поезда дает стабильное и многократное уменьшение числа параметров, возможности практически полного восстановления точности предсказания модели, но при необходимости мгновенного отклика от моделей нейронных сетей ТТ следует применять с осторожностью, учитывая различные характеристики преобразуемых данных (размер, источник данных, связность и другие) и конкретную задачу, стоящую перед пользователем.

Проецируя полученные экспериментальные результаты на задачи в сфере робототехники, где, как было упомянуто, активно используется ТТ-разложение, можно сделать вывод, что ТТ-разложения позволяют существенно сжимать представление многомерных распределений без потери качества найденных решений. При использовании ТТ-аппроксимации достигается высокая вероятность успешной сходимости после локальной оптимизации, причём решения остаются близкими к глобальным минимумам даже в присутствии препятствий и сложной геометрии среды. Полученные несоответствия в скорости работы вычислительных моделей на синтетических и реальных данных отлично согласуются с уже проведенными исследованиями, подтверждающими ограничения быстродействия ТТ-представлений, когда данные представлены в виде очень больших многомерных тензоров с необходимостью их полного использования в часто повторяющихся математических операциях [6].

5. Будущие исследования

Полученные результаты демонстрируют высокую эффективность формата тензорного поезда для сжатия многомерных выходных представлений нейронных сетей: достигнутое сжатие в $6.0\times$ для свёрточных и $4.0\times$ для полносвязных архитектур при контролируемой ошибке восстановления подтверждает практическую применимость метода в сценариях с ограниченными ресурсами хранения и передачи данных. При расширении подхода на

прямую факторизацию весовых тензоров в глубоких архитектурах, возникает ряд вычислительных ограничений, требующих отдельного исследования.

Первое направление связано со стабильностью градиентного обучения факторизованных параметров. При прямом обучении весовых тензоров в ТТ-формате мультипликативная структура обратного распространения ошибки

$$\frac{\partial L}{\partial G_k} = \left(\prod_{j < k} G_j^T \right) \cdot \frac{\partial L}{\partial W} \cdot \left(\prod_{j > k} G_j^T \right)$$

создаёт условия для неконтролируемого роста нормы градиента при увеличении глубины разложения $d > 4$ и ранга $r_{TT} > 12$. Подобная динамика хорошо согласуется с известными эффектами взрывных градиентов в мультипликативных параметризациях и была подтверждена в предварительных численных экспериментах, где наблюдалось резкое ухудшение сходимости при увеличении рангов ТТ-ядер. Для стабилизации обучения перспективным представляется использование спектрально-нормализованных параметризаций ядер:

$$\tilde{G}_k = \sigma_k \cdot \frac{G_k}{\|G_k\|_F}, \sigma_k = \exp(\tau_k)$$

Где скаляр σ_k обучается отдельно, обеспечивая адаптивный контроль масштаба градиента без искажения направления обновления параметров. В качестве дальнейшего расширения данного подхода может быть рассмотрена замена фиксированной линейной топологии ТТ-ядер на адаптивную графовую структуру с обучаемыми весами рёбер контракции.

Вторым ключевым аспектом является рост вычислительной латентности при переносе ТТ-факторизации на веса архитектур реального времени. Теоретический анализ показывает, что стандартная реализация ТТ-свёртки через последовательность операций $\text{unfold} \rightarrow \text{einsum} \rightarrow \text{fold}$ существенно увеличивает задержку вывода по сравнению с базовой моделью. В частности, для ResNet18 latency может возрастать с 2.1 мс до порядка 60 мс, что обусловлено доминирующим вкладом последовательных запусков CUDA-ядер и нарушением пространственной локальности при обработке патчей. Критически важным направлением является интеграция ТТ-операторов с аппаратно-ускоренными библиотеками, такими как cuTENSOR и CUTLASS, предоставляющими оптимизированные примитивы для тензорных сокращений с поддержкой Tensor Cores архитектур NVIDIA Ampere и Hopper. Реализация монолитных вычислительных ядер, объединяющих доступ к входному тензору, вычисление весов в ТТ-формате и агрегацию выходных активаций в едином проходе без промежуточной материализации, потенциально позволяет снизить задержку до 5-7 мс при сохранении коэффициента сжатия на уровне 5-6 раз от изначальных размеров.

Наконец, отдельный интерес представляет гибридная стратегия факторизации для свёрточных слоёв, ориентированная на сохранение вычислительной эффективности. Для ядер размера 3×3 целесообразной является селективная факторизация исключительно каналовых мод $c_{in} \times c_{out}$ при сохранении плотного представления пространственных измерений $K \times K$. Такой компромисс позволяет сохранить пространственную локальность операций свёртки, критичную для эффективности кэширования и работы с памятью, одновременно обеспечивая сжатие параметров в 4-5 раз за счёт устранения избыточности в пространстве размерностей слоев. Предварительные оценки показывают, что сочетание селективной факторизации и аппаратной акселерации через cuTENSOR позволяет достичь задержки вывода не выше 4 мс при снижении точности менее чем на 1.5% относительно исходной модели.

5. Заключение

Проведенное исследование демонстрирует, что применение тензорного разложения в формате ТТ открывает новые возможности для оптимизации вычислительной сложности DNN, особенно в условиях ограниченных вычислительных ресурсов. Эксперименты с использованием CNN и FC нейронных сетей показали, что ТТ-разложение позволяет значительно сократить объем параметров и ускорить вычисления, сохраняя при этом приемлемую точность. Важно отметить, что CNN-модель продемонстрировала более высокий коэффициент сжатия (6.0x против 4.0x у FC) и меньшую ошибку восстановления (0.10 против 0.18), что связано с её способностью эффективно использовать пространственные корреляции данных. При этом FC-модель оказалась быстрее в сжатии (1.5 мс против 5.0 мс), что делает её предпочтительной для задач реального времени. Особой закономерностью стало влияние ранга ТТ-разложения: низкие ранги обеспечивают высокое сжатие, но жертвуют точностью, тогда как оптимальный ранг, найденный в области "колена" кривой, позволяет достичь баланса между этими факторами. По сравнению с традиционными методами, такими как pruning и квантизация, ТТ-разложение выделяется гибкостью и регулярной структурой данных, что улучшает совместимость с аппаратными платформами. Однако исследование выявило и проблему: выбор оптимального ранга требует тщательной настройки, что усложняет процесс для неспециалистов. Дальнейшей перспективой является разработка автоматизированных алгоритмов подбора ранга, а также интеграция ТТ-разложения в облачные платформы для динамической оптимизации моделей. Работа подтвердила высокую эффективность ТТ-разложения, но также обозначила новые вызовы, связанные с необходимостью балансировки между сжатием, точностью и вычислительными затратами, что требует дальнейших исследований. На практике ТТ-подходы представляют собой перспективное направление для решения задач оптимизации работы с многомерными данными в сфере робототехники, позволяя компактно моделировать вычислительные и оценочные функции, а также эффективно работать с мультимодальными распределениями. Дальнейшее развитие методов ТТ в сочетании с современными стратегиями оптимизации и обучения может расширить область их применения в задачах управления и планирования действий в сфере робототехники.

References

- [1]. Оселедец И.В. Тензорное разложение в формате поезда. М.: Наука, 2017. 320 с.
- [2]. Иванова Н.Е., Гусев В.И. Применение тензорных разложений для сжатия весов нейронных сетей. Математическое моделирование и анализ данных, 2022, т. 33, № 5, с. 243-251. DOI: 10.1109/ICCSME.2022.0098.
- [3]. Тонков М.А., Петров П.С., Иванов В.В. Применение тензорных разложений для оптимизации глубинных нейронных сетей. Моделирование и анализ сложных систем, 2021, т. 45, № 2, с. 79-88. DOI: 10.31261/ams.2021.45.02.079.
- [4]. Khan S., Islam R., Cheng Y. A Comprehensive Survey of Model Compression and Acceleration for Deep Learning. *Neurocomputing*. 2022, vol. 511, pp. 229-261. DOI: 10.1016/j.neucom.2022.08.057.
- [5]. Кузнецов А.В. Оптимизация вычислительных графов с использованием тензорных разложений. Программная инженерия, 2023, № 5. с. 32-40.
- [6]. Shetty S. et al. Tensor train for global optimization problems in robotics. *The International Journal of Robotics Research*, 2024, vol. 43, no. 6, pp. 811-839.
- [7]. Gorodetsky A. A., Karaman S., Marzouk Y. M. Efficient High-Dimensional Stochastic Optimal Motion Control using Tensor-Train Decomposition. *Robotics: Science and Systems*, 2015.
- [8]. Shetty S., Silvério J., Calinon S. Ergodic exploration using tensor train: Applications in insertion tasks, *IEEE Transactions on Robotics*, 2021, vol. 38, no. 2, pp. 906-921.
- [9]. Рыбаков А.Г. Применение тензорных разложений для сжатия данных в задачах глубокого обучения. М.: БМК МГУ, 2020. 284 с.

- [10]. Жданов Н.С., Лобанов В.П. Основы вычислительных методов и их применение в машинном обучении. М.: Физматлит, 2021. 240 с.
- [11]. Günnemann S., Zhang L., Song Y. Tensor Decompositions and Low-Rank Approximations in Deep Learning. *ACM Transactions on Machine Learning*, 2021, vol. 4, no. 3 (27). DOI: 10.1145/3403681.
- [12]. Vinodha, D., Mary Anita, E.A. (2023). Integrity assured multi-functional multi-application secure data aggregation in wireless sensor networks (IAMFMA-SDA). *Journal of Intelligent & Fuzzy Systems*, vol. 45, no. 1, pp. 1453-1472.
- [13]. Петров Д.С., Иванов И.В. Методы оптимизации вычислительных затрат нейронных сетей. М.: Изд-во МФТИ, 2023. 250 с.
- [14]. Коротаев В.В., Соколов А.В., Шишкин В.А. Методы оптимизации и их применение. Москва: Наука, 2015. 320 с.
- [15]. Фролов С.В. Алгоритмы оптимизации и нейронные сети: методы и подходы. М.: Юрайт, 2022. 320 с.
- [16]. Куликов С.Ю., Ковалев Р.Н. Тензорные разложения и их приложения в анализе данных. СПб.: БХВ-Петербург, 2020. 320 с.
- [17]. Kotlyarov, D.V., Dyudyun, G.D., Rzhetskaya, N.V., Lapina, M.A., Babenko, M.G. Investigation of Adversarial Attacks on Pattern Recognition Neural Networks. *Proceedings of the Institute for System Programming of the RAS*, 35(2), pp. 35-48.
- [18]. Смирнов И.М., Соколова А.Б. Оптимизация нейросетей и тензорные методы: Теория и практика. М.: Физматлит, 2021. 312 с.
- [19]. Иванов Д.А., Савельев А.В. Применение тензорных разложений для ускорения нейросетевых вычислений на графических процессорах. *Вестник Московского университета. Серия 19: Математика и механика*, 2023, т. 78, № 1, с. 14-22. DOI: 10.1134/S0075462823010036.
- [20]. Mary Anita, E.A., Jeneffa, J., Vinodha, D., Lapina, M. (2024). An Efficient Compressive Data Collection Scheme for Wireless Sensor Networks. *Lecture Notes in Networks and Systems*, 1207 LNNS, pp. 31-47.
- [21]. Михайлов П.А., Савельев И.А., Дмитриев С.С. Алгоритмы сжатия нейронных сетей с применением тензорных разложений для ускорения вычислений. *Вопросы вычислительной техники и математического моделирования*, 2022, т. 14, № 3, с. 45-54. DOI: 10.21121/computer.2022.14.03.045.
- [22]. Герценштейн И.П. Тензорные разложения и их приложения в машинном обучении. М.: Наука, 2022. 400 с.
- [23]. Han S., Pool J., Tran J., Dally W. Learning both Weights and Connections for Efficient Neural Networks. *Advances in Neural Information Processing Systems*, 2015, vol. 28, pp. 1135-1143. DOI: 10.1109/TNNLS.2019.2904314.
- [24]. Козлов А.Н., Петров С.М. Применение тензорных поездов (Tensor Train) для сжатия весов нейронных сетей. *Математические методы в машинном обучении*, 2021, т. 19, № 3, с. 97-105.
- [25]. Поляков А.Н., Иванов С.В. Использование тензорных разложений для уменьшения вычислительных затрат в глубоких нейронных сетях. *Журнал вычислительной математики и математической физики*, 2022, т. 62, № 4, с. 532-543.
- [26]. Lapina, M., Sajid, M., Bagautdinova, A., Lapin, V., Tokmakova, M. (2024). Comparative Characteristics of the Methods of Using Digital Watermarks to Protect Text Information. In: Raza, Z., Babenko, M., Sajid, M., Lapina, M., Zolotarev, V. (eds) **AISMA-2023. Lecture Notes in Networks and Systems**, vol 1207. Springer, Cham.
- [27]. Горшков А.М., Данилова Н.В. Тензорное разложение для оптимизации скорости обучения в нейронных сетях. *Информационные системы и технологии*, 2022, т. 27, № 6, с. 125-132. DOI: 10.21255/ist.2022.06.125
- [28]. Федоров А.Ю., Михайлова О.В. Модели глубокого обучения с использованием тензорных разложений для оптимизации вычислительных затрат. *Вопросы информатики и вычислительных технологий*, 2021, т. 29, № 4, с. 43–51.
- [29]. Куликов В.М., Михайлова И.В. Методы оптимизации нейронных сетей на основе сжатия параметров с использованием тензорных разложений. *Вестник Сибирского государственного университета телекоммуникаций и информатики*, 2021, т. 24, № 2, с. 115–124.
- [30]. Lapina, M., Neelakandan, S., Borodulin, I., Koronskiy, A., Martemyanov, M. Research and Modification of Montgomery Multiplication Algorithm. In: Raza, Z., Babenko, M., Sajid, M., Lapina, M., Zolotarev, V. (eds) **AISMA-2023: International Workshop on Advanced Information Security Management and Applications. AISMA 2023. Lecture Notes in Networks and Systems**, vol 1207. Springer, Cham.

- [31]. Тимофеев В.В., Артемьева Н.В. Тензорные разложения в контексте сжатия нейронных сетей и оптимизации их вычислительных характеристик. Вестник ВГУ. Серия "Информатика и вычислительная техника", 2023, т. 47, № 2, с. 78-85. DOI: 10.17344/vvgu.2023.02.078.
- [32]. Петров Д.А., Смирнов И.П. Тензорные методы сжатия глубоких нейронных сетей. Информационные технологии, 2022, т. 28, № 4, с. 210-219.
- [33]. Novikov A. et al. Tensorizing neural networks. *Advances in neural information processing systems*, 2015, vol. 28.
- [34]. Novikov A. et al. Tensor train decomposition on tensorflow (t3f). *Journal of Machine Learning Research*, 2020, vol. 21, no. 30, pp. 1-7.

Информация об авторах / Information about authors

Мария Анатольевна ЛАПИНА – кандидат физико-математических наук, доцент кафедры вычислительно математики и кибернетики Северо-Кавказского федерального университета. Сфера научных интересов: цифровые технологии, управление информационной безопасностью, процессный подход, криптография.

Maria Anatolyevna LAPINA – Cand. Sci. (Phys.-Math.), Associate Professor at the Department of Computational Mathematics and Cybernetics at the North Caucasus Federal University. Research interests: digital technologies, information security management, process approach, and cryptography.

Виталий Геннадьевич ЛАПИН – кандидат физико-математических наук, доцент кафедры вычислительно математики и кибернетики Северо-Кавказского федерального университета. Сфера научных интересов: цифровые технологии, управление информационной безопасностью, процессный подход, криптография, медицинские информационные системы, анализ медицинских данных.

Vitaly Gennadievich LAPIN – postgraduate student in Physics and Mathematics, Associate Professor at the Department of Computational Mathematics and Cybernetics of the North-Caucasus Federal University. Research interests: digital technologies, information security management, process approach, cryptography, medical information systems, medical data analysis.

Альбина Дахировна КАРАЕВА – студент Северо-Кавказского Федерального университета. Сфера научных интересов: криптография, машинное обучение, цифровые технологии, управление информационной безопасностью, процессный подход, образовательный процесс.

Albina Dakhirovna KARAEVA – student of the North Caucasus Federal University. Research interests: cryptography, machine learning, digital technologies, information security management, process approach, and educational process.

Максим Евгеньевич ПАРФЕНЬТЬЕВ – аспирант Российского технологического университета МИРЭА. Сфера интересов: цифровые технологии, машинное обучение, нейронные сети, большие данные, оптимизация вычислительных моделей.

Maxim Evgenyevich PARFENTYEV – postgraduate student at MIREA – Russian Technological University. Research interests: digital technologies, machine learning, neural networks, big data, optimization of computational models.

Максим Евгеньевич СЕЛИВАНОВ – аспирант Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов: цифровые технологии, машинное обучение, аффективные вычисления, мультимодальный анализ, искусственный интеллект.

Maxim Evgenyevich SELIVANOV – postgraduate student at the V.P. Ivannikov Institute for System Programming of the Russian Academy of Sciences. Research interests: digital technologies, machine learning, affective computing, multimodal analysis, artificial intelligence.

DOI: 10.15514/ISPRAS-2026-38(1)-13



Анализ и оптимизация задержек видеотрансляции удалённых рабочих столов

И.С. Проскуряков, ORCID: 0009-0005-1559-6278 <ivan.proskuryakov@ispras.ru>

Д.А. Бадалян, ORCID: 0000-0002-0642-9102 <dbadalyan@ispras.ru>

*Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

Аннотация. Для повышения эффективности работы удалённых рабочих столов ключевое значение имеет оптимизация передачи видеоданных, обеспечивающая стабильность и качество изображения при ограниченных ресурсах сети и клиента. Современные подходы включают использование специализированных протоколов передачи данных и видеокодеков, оптимизированных для работы в реальном времени. Целью данной работы является исследование и оптимизация задержек в процессе передачи видео в системе предоставления доступа к удалённым рабочим столам с использованием GStreamer для обработки видео и WebRTC для передачи данных по сети. В результате работы выявлены основные источники задержек внутри конвейеров подготовки видео удалённого рабочего стола. Проведён сравнительный анализ поведения различных конвейеров в различных сценариях и выявлены их преимущества и недостатки в контексте оптимизации трансляции изображения рабочего стола. В ходе анализа эффективности и предсказуемости рассматриваемых GStreamer-конвейеров иллюстрируются подходы и инструменты оценки данного класса конвейеров подготовки кадров. Отдельным результатом являются полученные методы оптимизации, перечисленные в разделе 3.

Ключевые слова: видеотрансляции; удаленные рабочие столы; видеокодеки; стандарт H.264; стандарт AV1; стандарт VP9; платформа GStreamer; анализ задержек; инструмент GstShark.

Для цитирования: Проскуряков И.С., Бадалян Д.А. Анализ и оптимизация задержек видеотрансляции удалённых рабочих столов. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 183–200. DOI: 10.15514/ISPRAS-2026-38(1)-13.

Благодарности: авторы выражают благодарность О.Д. Борисенко, Н.А. Лазареву, А.М. Бирюкову за помощь в подготовке текста. Исследования выполнены в интересах ООО «Системные решения» и дистрибутива “Moncloud”.

Analysis and Optimization of Video Transmission Delays in Remote Desktops

I.S. Proskuriakov, ORCID: 0009-0005-1559-6278 <ivan.proskuryakov@ispras.ru>

D.A. Badalyan, ORCID: 0000-0002-0642-9102 <dbadalyan@ispras.ru>

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

Abstract. To improve the efficiency of remote desktop operations, the optimization of video data transmission is of key importance, ensuring stability and image quality under limited network and client resources. Modern approaches include the use of specialized data transmission protocols and video codecs optimized for real-time operation. The aim of this work is to investigate and optimize latency in the video transmission process within a remote desktop access provisioning system, utilizing GStreamer for video processing and WebRTC for network data transmission. As a result of the work, the main producers of latency inside the pipelines of remote desktop video preparation have been identified. A comparative analysis of the behavior of different pipelines in different scenarios has been carried out and their advantages and disadvantages have been identified in the context of optimizing the translation of the desktop image. During the analysis of the effectiveness and predictability of the considered GStreamer pipelines, approaches and tools for evaluating this class of training pipelines are illustrated. A separate result is the optimization methods listed in section 3.

Keywords: video streaming; remote desktops; video codecs; H.264; VP9; AV1; GStreamer; latency analysis; GstShark.

For citation: Proskuriakov I.S., Badalyan D.A. Analysis and Optimization of Video Transmission Delays in Remote Desktops. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 183-200 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-13.

Acknowledgements. The authors express their gratitude to O.D. Borisenko, N.A. Lazarev, and A.M. Biryukov for their assistance in preparing the text. This work was conducted in the interest of System Solutions LLC and Moncloud Distribution.

1. Введение

Под системой по предоставлению доступа к виртуальным удалённым рабочим столам (Virtual Desktop Infrastructure, VDI) здесь и далее понимается система, автоматизированно размещающая среды рабочих столов на некотором наборе виртуальных или физических серверов и предоставляющая доступ к ним. Среда рабочих столов включает в себя операционную систему и прикладное ПО. Среда виртуальных рабочих столов помещается в виртуальные машины, за управление и подключение к которым отвечает система VDI за счёт наличия слоя-перехватчика между клиентом и рабочим столом.

Одним из ключевых аспектов функционирования систем по предоставлению удалённого доступа к рабочим столам является трансляция изображения. Трансляция, производимая специальным компонентом системы, должна обеспечивать приемлемое качество изображения, минимальную задержку и стабильность даже в условиях ограниченной пропускной способности сети.

Требования к качеству изображения, задержке и стабильности делают оптимизационную задачу комплексной: необходимо обеспечить компромисс между качеством изображения, уровнем сжатия, количеством затрачиваемых на кодирование и декодирование видео временных и вычислительных ресурсов с учётом пропускной способности сетевого канала между удалённым вычислителем и клиентской машиной.

Цель данной работы – исследование поведения конвейеров подготовки кадров в системе трансляции VDI для их дальнейшей оптимизации. Для этого проводится качественный анализ поведения конвейеров подготовки кадров, используемых в разрабатываемой в ИСП РАН системе трансляции удалённых рабочих столов Wormhole, в контексте реализации

выделенных методов оптимизации трансляции. Новизна полученных результатов заключается в описании узких мест данных конвейеров и побочных эффектов работы некоторых стандартных объектов конвейеров в контексте системы трансляции VDI. В ходе проведённого анализа демонстрируются оценки и инструменты, которые могут быть использованы при профилировании производительности конвейеров подготовки кадров в VDI системе.

Wormhole состоит из клиентской и серверной части: с серверной ведётся трансляция на клиент, для трансляции используется технология WebRTC [1]. Захват экрана удалённого рабочего стола и кодирование видео осуществляются инструментами GStreamer. Клиентом выступает веб-браузер на базе проекта Chromium. Обобщённая архитектура Wormhole показана на рис. 1.

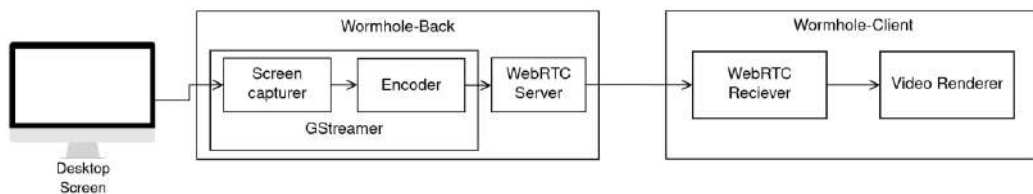


Рис. 1. Обобщённая архитектура компоненты трансляции Wormhole.
Fig. 1. Generalized architecture of the Wormhole streaming component.

2. Обзор

В данном разделе приводится краткий обзор тематик публикуемых исследований в контексте виртуальных рабочих столов, получивших название VDI систем (Virtual Desktop Infrastructure), а также справочная информация о наборе технологий, задействованном в задаче видеотрансляции удалённых рабочих столов.

Ряд исследований VDI посвящён оценке эффективности наиболее известных проприетарных систем – VMWare, Citrix, Microsoft. Так, в [2] оценивается поведение данного класса VDI систем в различных сетевых условиях с точки зрения качества предоставляемого сервиса (Quality of Service, QoS). В работах [3–4] оценивается производительность VDI на основе открытых технологий с использованием графических процессоров для улучшения производительности рабочих сред. Ряд исследований затрагивают тему оптимизации доставки трафика и решают оптимизационную задачу путём оптимального размещения сред в облачной среде с помощью технологии периферийных вычислений (edge-computing) [5], с помощью пересмотра подхода к представлению и сжатию кадра для специфичного контента путём реализации собственных решений [6]. Общим у приведённых исследований является отсутствие фокуса на анализе самого процесса подготовки кадров на стороне VDI с использованием актуальных технологий видеокодирования, поэтому далее остановимся на используемых для этого технологиях подробнее.

На рис. 2 показана последовательность, в которой происходит формирование видеокадров. Первым является этап получения набора снимков рабочего стола в RGB-подобном формате. Это делается штатными средствами операционных систем.

Далее происходит этап конвертации, необходимый для представления цвета в виде одной яркостной компоненты и двух цветовых (YUV). Этот подход позволяет эффективно сжимать данные на этапе кодирования, осуществляемого кодировщиком.

Под кодированием данных понимается преобразование из исходного формата в другой, как правило, наиболее пригодный для определённого сценария взаимодействия с пользователем. Кодирование осуществляется кодировщиками – программными или программно-аппаратными реализациями стандартов представления и сжатия видео. Наиболее известные

в области стандарты – H.264, H.265, VP9, AV1, где H.265 и AV1 имеют специальные режимы для оптимизации кодирования видео экрана рабочего стола [7:Chapter 3.8, 8-9].

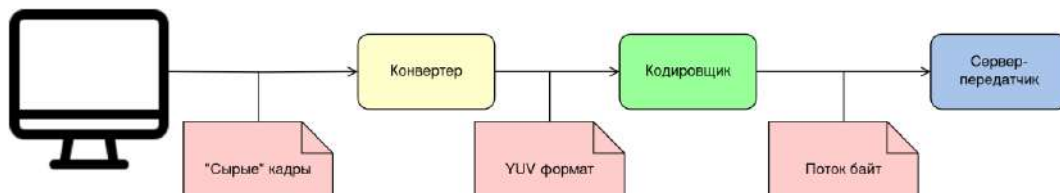


Рис. 2. Последовательность подготовки данных для видеотрансляции удалённого рабочего стола.

Fig. 2. Remote desktop video streaming data preparation sequence.

Для упрощения работы с каждым из перечисленных этапов разработаны системы обработки видео контента, позволяющие выстраивать сложные сценарии работы с медиа, в том числе с использованием программных интерфейсов. Среди данного класса систем выделяются две – FFmpeg и GStreamer. Система FFmpeg представляет собой набор библиотек по кодированию, декодированию, транскодированию, мультимплексированию, демультимплексированию, фильтрации, стримингу и воспроизведению видео. Система GStreamer имеет ту же функциональность, что и FFmpeg, однако отличается по своей структуре, предоставляя возможность конструирования графов из компонентов для обработки медиа данных и позволяя конструировать конвейеры обработки данных со стандартизированными интерфейсами.

Финальным этапом является отправка данных получателю посредством специальных сетевых технологий. В случае с системами трансляции видео удалённого рабочего стола необходимо иметь возможность трансляции в режиме, близком к режиму реального времени, то есть с задержкой около 100 мс [10]. Данным требованиям удовлетворяют 3 открытых технологии – WebRTC (Web Real-Time Communication) [10], протокол WebTransport совместно с интерфейсом WebCodecs [11], протокол QUIC [12]. WebRTC является достаточно сложной технологией и требует использования протоколов обхода NAT (Network Address Translation), таких как ICE [13], STUN [14], TURN [15]. Протокол WebTransport целиком полагается на стандарт HTTP/3 [16], из-за чего уходит необходимость в реализации механизма обхода протокола NAT [17]. Однако HTTP/3 и связанный с ним протокол QUIC ещё не получили достаточного распространения [18].

3. Описание методов оптимизации процесса видеотрансляции в задаче удалённых рабочих столов

Вынося за рамки общие методы оптимизации передачи аудио- и видеоконтента в информационной сети в режиме реального времени (видеостриминг), такие, например, как выбор оптимального профиля кодирования, размера группы изображений, а также методы оптимизации работы транслируемой операционной системы внутри VDI, можно выделить следующие подходы к оптимизации видеотрансляций в задаче VDI.

1. Использование кодировщиков видеоизображений с механизмами оптимизации кодирования изображения, характерного для рабочего стола персонального компьютера. Задача кодирования такого класса медиаданных получила название технологии кодирования содержимого экрана (экранного контента) SCC (Screen Content Coding), методы её оптимального решения предусматриваются стандартами видеокодирования AV1 и H.265 [19].
2. Использование аппаратных реализаций кодировщиков видео: стандарты H.264, H.265 и AV1 имеют аппаратные реализации на графических процессорах [20-22] и в графических компонентах процессоров [23].

3. Использование специализированных систем, таких как Microsoft DirectX [24] и Vulkan [25], для получения доступа к ресурсам графических процессоров для аппаратного декодирования.
4. Обеспечение zero-copу подхода при работе с медиаданными: копирование является операцией, затратной по времени и по памяти.
5. Обеспечение стабильного сетевого канала без перегрузок: в задаче оптимизации видеотрансляций с помощью WebRTC широко распространено использование алгоритмов контроля за недопущением перегрузки канала (Congestion Control) для корректировки частоты видеопотока по текущему состоянию сетевого канала. В данном классе алгоритмов выделяются алгоритмы GCC [26], NADA [27], SCReAM [28]. Аналогичный подход с использованием обратной связи потенциально возможно реализовать и для сценария трансляции удалённых рабочих столов.

Для оценки практической эффективности 4-х из 5 предложенных выше оптимизаций необходимо иметь инструментарий, контролирующий изменения задержек, формируемых конвейером подготовки кадров.

4. Методика сбора данных

В разделе приводится информация о структуре анализируемых GStreamer конвейеров подготовки видео, а также основная информация об используемом в работе инструменте профилирования – GstShark.

4.1 Описание структуры GStreamer конвейеров

GStreamer конвейеры состоят из объектов, проводящих манипуляции над данными. Для передачи данных из конвейера внешним потребителям используется специальный объект – ‘appsink’. В случае, если объекты работают в разном темпе, и подозревается ситуация, когда предыдущий этап конвейера данные проходят быстрее, чем следующий, во избежание потерь данных между обработчиками используются очереди (queue), осуществляющие буферизацию и выполнение политик по времени хранения буферизированных данных [29].

Данные передаются между объектами конвейеров в специальных контейнерах, так называемых буферах (buffers). Эти данные могут быть снабжены набором метаданных, такой как формат данных, длительность находящегося в буфере видеофрагмента и так далее.

4.2 Описание инструмента сбора данных

Для оценки эффективности конфигураций конвейеров подготовки видео необходимо получать информацию об этапах формирования задержек на каждом из этапов, которые проходят данные при обработке конвейером GStreamer. Данную функциональность предоставляет находящийся в открытом доступе инструмент GstShark [30]. В составе инструмента имеется ряд трассировщиков, предоставляющих данные о поведении конвейера:

1. ‘bitrate’ – измеряет и выводит частоту (количество бит в секунду) данных, выходящих из src-интерфейса объекта конвейера.
2. ‘buffer’ – выводит информацию о каждом буфере (единице-контейнере данных в библиотеке GObject, используемой системой GStreamer для реализации концепции ООП), покидающем src-интерфейс объекта конвейера.
3. ‘cpuusage’ – выводит информацию о загрузке ядер ЦПУ.
4. ‘framerate’ – измеряет и выводит частоту выводимых объектом конвейера кадров, если такая информация объектом предоставляется.
5. ‘graphic’ – визуализирует структуру GStreamer конвейера с указанием форматов данных, проходящих через объекты.

6. ‘interlatency’ – межэлементная задержка, отображающая время прохождения буфера данных между элементами конвейера.
7. ‘proctime’ – измеряет и выводит время, потраченное объектом для передачи буфера от входного интерфейса (sink) к выходному (src).
8. ‘queuelevel’ – измеряет количество данных, пребывающих в очередях, и выводит его.
9. ‘scheduletime’ – измеряет и выводит интервал между поступлением буферов данных на входной интерфейс (sink) объекта.

Для анализа собранных данных разработчики GstShark предоставляют собственный инструмент визуализации, но при его использовании мы столкнулись с проблемой в работе компонента ‘gnuplot’. Решением стала разработка собственной программы для визуализации собранных логов с использованием Python3.11 в связке с инструментами Pandas, Numpy, Matplotlib.

Данные необходимо было собирать при работе Wormhole на ОС Windows 10, в связи с чем GstShark собирался из исходных файлов в среде MSYS2 и запускался в ней же. Такой подход позволил избежать использования дополнительной виртуализации, возникающей при использовании WSL, но пришлось отказаться от использования трассировщиков ‘сruusage’ и ‘graphic’, так как они использовали Linux-специфичные вызовы и программы.

5. Анализ наборов данных

5.1 Способ сбора наборов данных

Для сбора наборов данных и воспроизводимости полученных результатов стояла задача единообразно эмулировать некоторый сценарий работы за удалённым рабочим столом. Простейшим решением этой задачи, к которому мы прибегли, является включение на рабочем столе в полноэкранном режиме заготовленной видеозаписи с определёнными сценариями – данный способ эмуляции дешёв в эксплуатации и воспроизведении. Таким способом собраны все анализируемые наборы данных.

Основной сценарий используемых в заготовленных видеозаписях – сценарий “офисной работы”, то есть работа в пакете Microsoft Excel, программе проектирования Revit и просмотр веб страниц. Также использовался сценарий с воспроизведением динамически меняющейся сложной трехмерной графики. Далее, если не сказано иное, предполагается, что данные собраны на видео “офисной работы”.

Конфигурация экспериментального стенда следующая. Гостевая система Windows 10 на основе libvirt, хост система – Debian 12. Топология виртуальной машины: 4 виртуальных процессора, 16 Гб ОЗУ. Топология хостовой машины: NVIDIA A16, Intel Core i7-11700, 62.6 Гб ОЗУ.

Для каждой из анализируемых ниже трасс проводилось от 3 прогонов длительностью от 5 до 15 минут с целью получения качественной картины поведения конвейера. На иллюстрациях приведены графики, построенные по трассе одной трансляции – усреднения и иные модификации наборов данных для построения данных графиков не проводилось.

5.2 Анализ конвейеров, использующих ресурсы графических процессоров

Один из путей оптимизации видеотрансляции – использование аппаратно реализованных кодировщиков видео. С целью их эффективного использования проводится исследование конвейеров на основе аппаратного кодировщика NVIDIA.

5.2.1 Сравнительный анализ поведения реализации кодировщика H.264 от NVIDIA в ключевых сценариях работы

В разделе анализируются трассы, собранные при работе видео кодировщика H.264 в реализации NVIDIA в конвейере Wormhole (плагин 'nvh264') при трансляции двух качественно разных видеорядов: сценарий А – трансляция в полноэкранном режиме на рабочий стол записи мультфильма со сложной 3D графикой; сценарий Б – трансляция в полноэкранном режиме на рабочий стол записи работы с инструментами Excel.

Фрагмент графика, построенного по данным трассировщика 'proctime', показан на рис. 3. Из графика видно, что время, затрачиваемое на кодирование в сценарии А, выше, чем время, которое затрачивается при кодировании видео работы с “офисной” нагрузкой. Это доказывает, что несмотря на отсутствие заложенных в стандарт оптимизаций в задаче SCC, H.264 может справляться с кодированием контента, характерного для рабочего стола персонального компьютера, эффективнее, чем с кодированием сложных 3D сцен. Данный вывод является не очевидным, так как изначально стандарт H.264 разрабатывался для сжатия и представления “натурального” видео, характер которого отличается от видео рабочего стола [9]. В то же время развитие H.264, стандарт H.265, имеет специальные расширения профилей кодирования для повышения эффективности работы с видео рабочего стола [8].

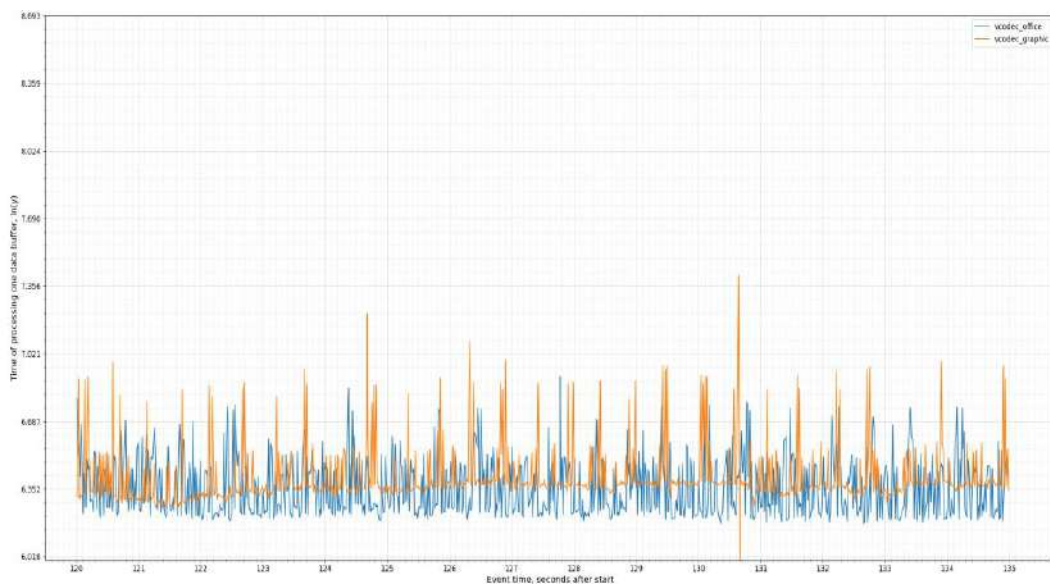


Рис. 3. Время (в логарифмической шкале), требуемое кодировщиком для передачи полученных данных для дальнейшей обработки конвейером. Оранжевая линия соответствует распределению времени кодирования трансляции сценария А, синяя – сценария Б.

Fig. 3. Encoder processing time (logarithmic scale) required to pass received data to the pipeline for further processing. The orange line represents the encoding time distribution for Scenario A, the blue line – for Scenario B.

5.2.2 Анализ конвейера на основе CUDA

CUDA – это платформа параллельных вычислений, разработанная компанией NVIDIA для вычислений на графических процессорах (GPU). Для использования аппаратно реализованных видеокодеков в графических процессорах NVIDIA используется CUDA [21, CUDA], доступная в GStreamer через набор специализированных плагинов. Для трансляции

экрана с помощью CUDA в Wormhole используется конвейер, представленный в Приложении 1.

Иллюстрация работы конвейера на основе CUDA-плагинов посредством фрагмента графика формирования задержки, построенного по данным трассировщика ‘interlatency’, показана на рис. 4. Каждая линия графика, за исключением самой верхней, соответствует логарифму времени, прошедшего от появления данных в начале конвейера (в нашем случае – на объекте захвата картинки рабочего стола) до момента выхода этих данных из объекта конвейера, указанного в легенде. Последняя линия, соответствующая объекту appsink, показывает логарифм времени до входа в объект. По горизонтали отложено время с момента начала трансляции рабочего стола до регистрации события появления данных трассировщиком.

Из рис. 4 видно, что относительно большая разница имеется под линиями, соответствующими выходам из объектов ‘queue0’, ‘cudaupload’ и ‘vcodec’. Выход из объекта ‘queue0’ соответствует попаданию данных в объект загрузки данных в память графического процессора – ‘cudaupload’. Выход из объекта ‘cudaupload’ соответствует попаданию данных из оперативной памяти вычислителя в графический процессор и началу процесса их обработки. Выход из объекта ‘vcodec’ соответствует окончанию процесса кодирования и попаданию в очередь перед отправкой на клиент. Иными словами, график показывает два ключевых источника задержки в конвейере: объект загрузки данных в память графического процессора и объект, осуществляющий кодирование видео. Также на данном фрагменте показан момент длительного отказа в приёме данных объектом ‘cudaupload’ – он соответствует пику между 998 и 999 секундами трансляции.



Рис. 4. Фрагмент графика ‘interlatency’ для конвейера на основе CUDA. По вертикали отложен логарифм времени, которое прошло с момента появления данных в начале конвейера до момента их регистрации на некотором интерфейсе некоторого объекта. По горизонтали отмечено время в секундах, прошедшее с момента начала трансляции экрана удалённого рабочего стола.

Fig. 4. A fragment of a plot built from ‘interlatency’ tracer for a CUDA-based pipeline. The vertical axis shows the logarithm of the time elapsed from the moment the data appeared at the beginning of the pipeline to the moment it was registered on a specific interface of a specific object. The horizontal axis shows the time in seconds elapsed since the start of the remote desktop screen streaming.

Как удалось выяснить, такие пики носят периодический характер и имеют накопительный эффект. Показанный на фрагменте пик, согласно проведённому исследованию логов Wormhole, привёл к ситуации потери изображения на клиенте из-за запаздывания кадров.

Вышесказанное свидетельствует, что конвейер на основе CUDA-плагинов имеет высокие издержки, связанные с загрузкой данных в память графического процессора. Периодические отказы объекта загрузки данных в память GPU приводят к ситуациям, ухудшающим пользовательский опыт.

Выявлены три пути решения данной проблемы: понижение частоты кадров, установка меньших дедлайнов на очередях в начале и в конце конвейера и локализация данных в памяти GPU средствами DirectX. Первый подход привёл к ещё более критичному ухудшению пользовательского опыта и делает нецелесообразным использование GPU для вычислений, так как при условии незначительных компромиссов возможно добиться той же частоты кадров средствами программной реализации кодировщика. Второй подход привёл к более нестабильной работе конвейера и увеличению дисперсии времени работы кодировщика, что не может поощряться в системах, работающих в режиме реального времени. Результаты третьего метода показаны в следующем разделе.

5.2.3 Анализ конвейера на основе DirectX 11

Конфигурация конвейера с использованием абстракции памяти GPU средствами DirectX 11 показана в Приложении 2. В ходе работы конвейера кадры сразу же отправляются в буфер под управлением DirectX – это становится возможным благодаря тому, что захват изображения экрана производится средствами DirectX. Данные проходят все этапы обработки внутри памяти под контролем DirectX и выгружаются в оперативную память только в самом конце – перед передачей данных в appsink.

На рис. 5 показан фрагмент графика, построенного по данным трассировщика ‘interlatency’. Данные собирались при трансляции “офисной” нагрузки средствами Wormhole. Из графика видно, что в конвейере отсутствует этап загрузки данных из оперативной памяти в память графического процессора, благодаря чему удалось ликвидировать побочные эффекты, возникающие в результате данного класса операций.

Также можно заметить по уровню линий appsink на рис. 4 и 5, что при переходе с CUDA-конвейера на конвейер на основе DirectX 11 удалось избежать увеличения затрачиваемого на подготовку кадров времени.

5.3 Анализ конвейеров, использующих программные реализации кодировщиков

В разделе анализируется на предмет этапов формирования задержек структура конвейеров на основе программно реализованных кодировщиков видео. Также приводится сравнение кодировщиков VP9 и x264 в ходе использования Wormhole под “офисной” нагрузкой – средствами GstShark демонстрируются выявленные недостатки в работе кодировщика VP9.

5.3.1 Анализ конвейера на основе программно реализованного кодировщика на примере конвейера с x264

x264 – это программная реализация стандарта кодирования H.264 с открытым исходным кодом [31]. Для работы с ней в составе GStreamer используется плагин x264enc. Полная версия анализируемого конвейера на основе x264 приведена в Приложении 3. Для анализа формируемых в ходе работы конвейера задержек используется трассировщик ‘interlatency’. Фрагмент графика ‘interlatency’ представлен на рис. 6. Каждая линия соответствует моменту появления данных на интерфейсе объекта конвейера. Соответствующие объект и его интерфейс указаны в легенде на самом графике.



Рис. 5. Фрагмент графика, построенного по данным трассировщика 'interlatency' на основе DirectX. По вертикали: логарифм времени, которое прошло с момента появления данных в начале конвейера до момента их регистрации на некотором интерфейсе некоторого объекта. По горизонтали: время в секундах, прошедшее с момента начала трансляции экрана удалённого рабочего стола.

Fig. 5. A fragment of a plot built from 'interlatency' tracer for a DirectX-based pipeline. The vertical axis represents the logarithm of the time elapsed from the moment data enters the pipeline to its registration on a specific interface of a specific object. The horizontal axis indicates the time in seconds since the start of the remote desktop screen streaming.



Рис. 6. Фрагмент графика, построенного по данным трассировщика 'interlatency' на основе x264. По вертикали отложен логарифм времени, которое прошло с момента появления данных в начале конвейера до момента их регистрации на некотором интерфейсе некоторого объекта. По горизонтали отмечено время в секундах, прошедшее с момента начала трансляции экрана удалённого рабочего стола.

Fig. 6. A fragment of a plot built from 'interlatency' tracer for a x264-based pipeline. The vertical axis represents the logarithm of the time elapsed from the moment data enters the pipeline to its registration on a specific interface of a specific object. The horizontal axis indicates the time in seconds since the start of the remote desktop screen streaming.

Как видно из фрагмента графика на рис. 6, наибольшая задержка формируется на следующих этапах:

- перед выходом данных из очереди ‘queue0’ – это очередь перед этапом конвертации;
- перед выходом данных из объекта конвертации данных;
- перед попаданием данных в appsink, то есть на этапе кодирования.

Задержка на выход из очереди ‘queue0’ говорит о том, что объект захвата кадров работает быстрее, чем обработчики данных – конвертер и кодировщик. Также видно, что в данном конвейере большая задержка формируется этапом конвертации – это характерная особенность конвейеров на основе программно реализованных кодировщиков и, как можно видеть из рис. 5 и 6, в конвейерах на основе аппаратно реализованных кодировщиков этап конвертации не занимает столь существенную долю времени работы конвейера.

В подтверждение того, что этап конвертации занимает больше времени, чем этап кодирования, может служить факт, что, согласно графику, линия выхода данных из ‘queue1’, соответствующей очереди между конвертером и кодировщиком, практически совпадает с линией выхода данных из объекта конвертера – это значит, что кодировщик практически мгновенно потребляет поступившие на вход данные.

5.3.2 Сравнительный анализ VP9 и x264 в сценарии трансляции рабочего стола в FullHD

Один из основных сценариев работы Wormhole заключается в предоставлении доступа к удалённому рабочему столу без подключенного графического процессора. Для эффективного выполнения поставленной задачи, помимо программной реализации x264 стандарта H.264, необходимо рассмотреть альтернативы. Альтернативой является кодировщик VP9 – это широко используемый открытый стандарт кодирования, разрабатывавшийся как альтернатива H.264 в задаче трансляции видео [32]. Для работы с GStreamer используется реализация ‘libvpx’. Для сравнения этих кодировщиков собраны и проанализированы данные трассировщиков, полученные в ходе трансляции через Wormhole “офисной” нагрузки через заранее записанные видеозаписи – аналогично сбору статистики в разделах выше.

Используемые для сравнения GStreamer конвейеры представлены в Приложении 4.

Для выявления особенностей работы конвейеров использовались трассировщики ‘buffer’, ‘interlatency’, ‘proctime’. Трассировщик ‘buffer’ предоставляет, в том числе, данные о длительности видеофрагментов, передаваемых конвейером в appsink. Для частоты 30 кадров в секунду (30 FPS) ожидаемая длительность передаваемого буфера данных составляет около 33 мс. Однако при просмотре статистики работы VP9 обнаружены обладающие периодичностью эпизоды, когда кодировщик передавал на выход буферы с около нулевыми длительностями. Пример такого поведения показан на 575-й секунде трансляции в фрагменте графика на рис. 7.

Для объяснения произошедшего рассматривается график по данным трассировщика ‘interlatency’ в окрестности зафиксированного падения длительности буфера данных – этот график представлен на рис. 8. На рис. 8 показано, что в окрестности 575 секунды формируется пик, соответствующий повышению времени на выход данных из очереди перед этапом конвертации – это означает, что последующие элементы не готовы принять данные для обработки. По расположению пиков на графике можно предположить, что причина в резко повысившейся задержке кодировщика – пик, соответствующий задержке кодировщика, расположен левее, чем пики конвертера и очереди. Чтобы подтвердить увиденное, обратимся к данным трассировщика ‘proctime’.

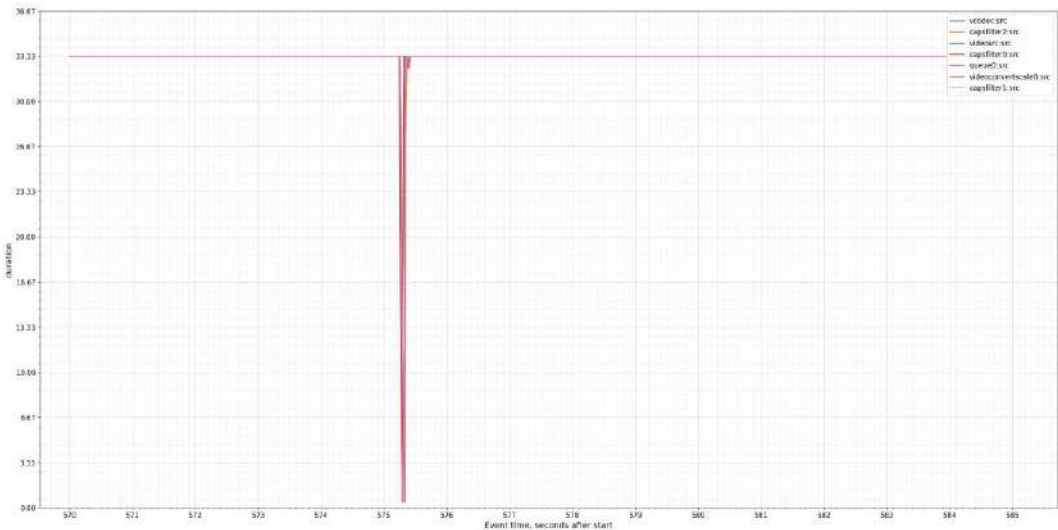


Рис. 7. Фрагмент графика по данным статистики “duration” трассировщика “buffer” по данным конвейера на основе VP9. По вертикальной оси расположены данные о длительности видеофрагмента, находящегося в *gst*-буфере данных (в мс). По горизонтальной оси отмечено время (в секундах) с момента начала трансляции рабочего стола. Падение между 575 и 576 секундами трансляции говорит о наличии аномалии в *GStreamer* конвейере: по некоторой причине длительность передаваемого видеофрагмента упала до около нулевых значений.

Fig. 7. Fragment of a plot based on “duration” statistics from the “buffer” tracer of a VP9-based pipeline. The vertical axis shows the duration of the video fragment in the *GStreamer* data buffer (in ms). The horizontal axis shows time (in seconds) since the start of desktop streaming. The drop between 575 and 576 seconds indicates an anomaly in the *GStreamer* pipeline: for some reason, the duration of the transmitted video fragment dropped to near zero values.

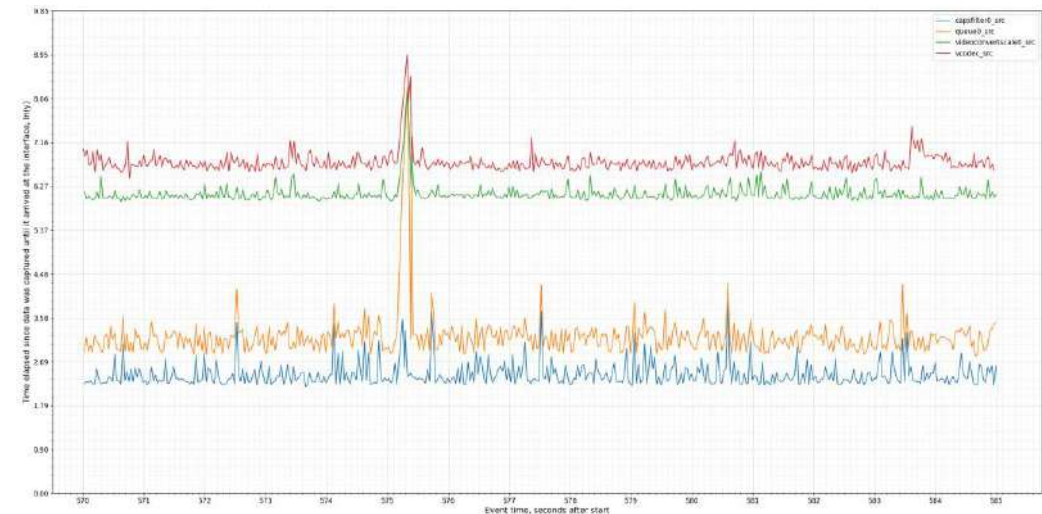


Рис. 8. Фрагмент графика, построенного по данным трассировщика “interlatency” по данным конвейера на основе VP9. Между 575 и 576 секундами трансляции видны аномальные задержки: левее расположен пик, соответствующий задержке выхода данных из кодировщика, далее – задержки объекта конвертации форматов (зелёный) и очереди перед ним (оранжевый).

Fig. 8. Fragment of a plot based on data from the “interlatency” tracer of a VP9-based pipeline. Anomalous delays are visible between 575 and 576 seconds: to the left is a peak corresponding to the delay of data exiting the encoder, followed by the format conversion element (green) and the queue preceding it (orange).

На рис. 9 показан график трассировщика 'proctime' для соответствующего временного участка – по нему видно, что скачок времени работы наблюдается лишь у кодировщика (сначала) и очереди 'queue0' (после кодировщика). Это доказывает, что увиденное на графике 'duration' на рис. 7 падение спровоцировано именно кодировщиком.

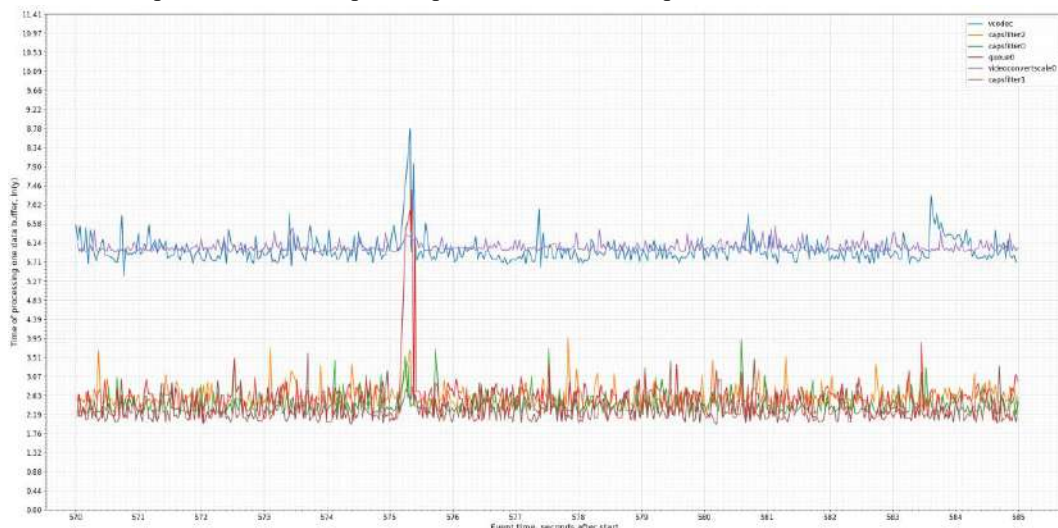


Рис. 9. Фрагмент графика, построенного по данным трассировщика 'proctime' по данным конвейера на основе VP9. По вертикали расположен логарифм времени обработки данных объектами GStreamer конвейера. По горизонтальной оси отмечено время (в секундах) с момента начала трансляции рабочего стола. Между 575 и 576 секундами отмечены два аномальных пика – время работы кодировщика (синим) и время, проведённое данными в очереди 'queue0' (красным).

Замедлению данных в очереди предшествует замедление обработки данных кодировщиком.

Fig. 9. Fragment of a plot based on data from the 'proctime' tracer of a VP9-based pipeline. The vertical axis shows the logarithmic scale of the processing time for data by GStreamer pipeline elements. The horizontal axis shows time (in seconds) since the start of desktop streaming. Two anomalous peaks are visible between 575 and 576 seconds: the encoder processing time (blue) and the time spent by data in the 'queue0' queue (red). The delay in the queue is preceded by a slowdown in the encoder's data processing.

5.3.3 Оценка потенциала AV1 в задаче трансляции удалённых рабочих столов

AV1 (AOMedia Video 1) – это открытый, свободный от выплаты лицензионных платежей правообладателям стандарт видеокодирования, разработанный организацией Alliance for Open Media (AOMedia) [33]. AV1 является преемником стандарта VP9 и ответом сообщества на эффективные, но дорогостоящие стандарты H.264 и H.265. Как сказано выше, стандарт AV1 определяет методы оптимального решения задачи SCC, поэтому является перспективным решением в задаче трансляции удалённых рабочих столов.

В Wormhole для использования AV1 сконструировано 2 GStreamer-конвейера: на основе референсной реализации libaom и на основе альтернативной – SVT – реализации. До внедрения поддержки AV1 в используемом в Wormhole медиасerverе проведён эксперимент по сбору информации о времени кодирования и размеров закодированных видео с помощью GStreamer-конвейеров. В ходе эксперимента сравнивались показатели конвейеров на основе реализаций x264, libaom и SVT-AV1 на 5 сценариях кодирования "офисной" нагрузки – заготовленных тематических видео длительностью до 2 минут. На каждый сценарий проводилось 25 прогонов видео, в ходе которых замерялись время кодирования и размер закодированного файла. Конфигурации конвейеров, используемые в экспериментах, помещены в Приложении 5. Результаты замеров и анализа данных представлены в табл. 1. Результаты не приводятся для libaom реализации, так как его использование в составе черес

GStreamer не актуально в рассматриваемой задаче: соответствующий GStreamer-плагин не предоставляет возможность гибкой настройки этого кодировщика, что ведёт к его низкой производительности в задаче кодирования видео рабочего стола. Конфигурация тестового стенда следующая: процессор Intel i7-3770, 24 Гб ОЗУ, ОС Manjaro Linux x86_64.

Как видно из табл. 1, в ряде сценариев SVT-AV1 показал превосходство над x264 по размеру итогового размера файла. Тем не менее внедрение AV1 в Wormhole ещё не произошло: SVT реализация показала себя как нестабильная – в мажорной версии 3 представленная в Приложении 4 конфигурация не работает из-за неспособности кодировщика работать в режиме константного битрейта. Плагин для работы с libaom в GStreamer требует доработок для возможности более гибкой настройки кодировщика.

Табл. 1. Статистические данные, полученные в ходе эксперимента по замеру размеров файлов и времени кодирования SVT-AV1 и x264.

Table 1. Statistical data obtained during the experiment measuring file sizes and encoding times for SVT-AV1 and x264.

Сценарий	Медианное время кодирования, SVT-AV1, с.	Медианный размер файла, SVT-AV1, МБ	Медианное время кодирования, x264, с.	Медианный размер файла, x264, МБ	Отношение времени кодирования, x264 / SVT-AV1	Отношение размеров файлов, x264 / SVT-AV1
Просмотр веб-страниц	116.183284	73.918007	109.746559	150.387691	0.944599	2.034520
Использование Microsoft Word	32.669588	8.386386	20.687374	12.312542	0.633230	1.468158
Работа в Word параллельно с работой в браузере без переключения вкладок	213.348808	31.453123	210.866439	47.955088	0.988365	1.524653
Процесс написания программного кода – IDE и браузер	118.456091	28.278339	124.338585	45.958537	1.049660	1.625221
Работа в Excel без создания диаграмм	44.977812	12.529974	29.433932	14.782100	0.654410	1.179739

6. Направления дальнейших работ

Дальнейшие работы могут быть разделены на 2 группы: улучшение прослеживаемости задержки и внедрение выделенных оптимизационных механизмов с последующей их оценкой.

В контексте улучшения прослеживаемости следующим этапом является создание инструментария прослеживания задержки от этапа появления кадра в начале конвейера до его отрисовки на клиенте. Также важным является направление по разработке на основе написанного парсера сервиса по предоставлению статистики работы конвейера в режиме реального времени.

В контексте внедрения выделенных оптимизационных механизмов важную роль играет модернизация плагинов GStreamer. Для работы с референсной реализацией AV1 необходимо внести изменения в плагин ‘av1enc’ – это позволит использовать SCC-специфичные настройки при использовании в конвейере подготовки кадров Wormhole. Изменения в плагины для работы с Vulkan и CUDA потенциально могут позволить добиться той же локализации данных, которую удалось получить средствами DirectX 11. Также перспективным направлением в контексте внедрённого инструментария является изучение влияния семантики ‘zero-copy’ на работу конвейера: необходимо рассмотреть влияние ускорения процесса передачи данных без копирований из конвейера. Наконец важными являются работы оптимизацией доставки данных через WebRTC, но для оценки влияния

этого фактора должны использоваться другие инструменты отслеживания поведения системы трансляции.

7. Выводы

В ходе исследования конвейеров обработки видео, включающих этапы кодирования, конвертации и трансляции, выявлен ряд особенностей функционирования различных конвейеров подготовки кадров, важных для разработки и оптимизации систем трансляции рабочих столов. Полученные результаты имеют практическую ценность не только для разрабатываемого в ИСП РАН решения, но и для широкого класса систем видеотрансляции, где критически важны низкие задержки и стабильность видеопотока.

- 1. Составлен и обоснован перечень основных направлений оптимизации видеотрансляции удалённых рабочих столов.**
- 2. Проанализирована работа конвейеров подготовки кадров в составе системы трансляции VDI.** Проведён анализ работы конвейеров, использующих аппаратные (CUDA, DirectX 11) и программные (x264, libvpx) реализации кодировщиков. Установлен ряд фактов о работе анализируемых конвейеров.
 - Конвейеры на основе CUDA демонстрируют значительные задержки, связанные с загрузкой данных в память GPU, что может приводить к периодическим отказам и ухудшению пользовательского опыта.
 - Конвейеры на основе DirectX 11 эффективно устраняют проблему загрузки данных в GPU за счёт использования буферов под управлением DirectX, что повышает стабильность работы системы.
 - Программные реализации, такие как x264, показывают стабильную работу, но имеют высокие задержки на этапе конвертации форматов, в отличие от аппаратных решений.
 - Реализация VP9 в сценарии трансляции FullHD демонстрирует аномалии, связанные с передачей буферов с около нулевыми длительностями, что делает его менее предпочтительным для высоких частот кадров.
 - Реализация AV1 показывает потенциал в уменьшении размера выходного видеопотока, но страдает от нестабильной функциональности (SVT) и ограничений в настройке в контексте GStreamer (libaom).
- 3. Получен ряд практических рекомендаций.**
 - Для минимизации задержек в аппаратных конвейерах предпочтительно использовать DirectX 11 вместо CUDA.
 - Программные реализации, такие как x264, подходят для систем без GPU, но актуальным направлением работ является оптимизация этапа конвертации представлений.
 - VP9, как и x264, является приемлемым вариантом в сценарии трансляции VDI без GPU, но при увеличении частоты кадров возможно ухудшение пользовательского опыта из-за потери кадров.
 - Внедрение AV1 требует доработки плагинов GStreamer для повышения гибкости конфигурации.
 - Разработанные парсеры логов GstShark на Python (с использованием Pandas, NumPy, Matplotlib) доказали свою эффективность для визуализации данных и будут использоваться в дальнейшем.

4. Внедрён инструментарий для анализа задержек конвейера подготовки кадров.

- Разработана и внедрена методология сбора и анализа данных о задержках в GStreamer-конвейерах с использованием инструмента GstShark. Это позволило повысить прозрачность влияния параметров кодирования и выявить основные источники задержек на различных этапах обработки видеоданных.

Проведённое исследование позволило не только выявить ключевые узкие места в процессе формирования кадров видеотрансляции, но и предложить конкретные подходы к их устранению, что способствует повышению качества и стабильности работы системы Wormhole.

8. Приложения

8.1 Приложение 1 – Конвейер на основе CUDA плагинов GStreamer

```
d3d11screencapturesrc ! queue ! cudaupload ! cudaconvertscale ! video/x-raw(memory:CUDAMemory) ! queue !  
nvh264enc name=nvcodec ! video/x-h264,profile=<profile> ! queue ! appsink name=appsink
```

8.2 Приложение 2 – Конвейер на основе DirectX 11 плагинов GStreamer

```
d3d11screencapturesrc ! video/x-raw(memory:D3D11Memory) ! queue ! d3d11convert ! video/x-raw(memory:D3D11Memory) ! queue !  
nvd3d11h264enc name=nvcodec ! video/x-h264,profile=<profile> ! queue ! appsink name=appsink
```

8.3 Приложение 3 – Пример конвейера на основе x264

```
d3d11screencapturesrc ! queue ! videoconvertscale ! video/x-raw ! queue ! x264enc name=nvcodec ! video/x-h264,profile=<profile> ! queue ! appsink name=appsink
```

8.4 Приложение 4 – Конвейеры для работы с VP9 и x264

Листинг А – Конвейер с x264

```
d3d11screencapturesrc ! queue ! videoconvertscale ! video/x-raw ! queue ! Vp9enc name=nvcodec ! video/x-vp9 ! queue ! appsink name=appsink
```

Листинг Б – Конвейер с VP9

```
d3d11screencapturesrc ! queue ! videoconvertscale ! video/x-raw ! queue ! x264enc name=nvcodec ! video/x-h264,profile=<profile> ! queue ! appsink name=appsink
```

8.5 Приложение 5 – Конфигурации конвейеров, используемые для эксперимента по замеру времени и размеру закодированного файла

Листинг А – Конфигурация кодировщика x264 для кодирования видео

```
x264enc threads=8 aud=false b-adapt=false key-int-max=0 sliced-threads=true byte-stream=true  
tune=zerolatency speed-preset=veryfast bitrate=20000 pass=17 ! video/x-h264,stream-format=byte-stream,profile=high
```

Листинг Б – Конфигурация кодировщика SVT-AV1 2.2.1 для кодирования видео

```
svtav1enc target-bitrate=20000 preset=11 parameters-string=':lp=0:rc=2:pred-struct=1:overshoot-pct=10:scm=1:tile-rows=2:tile-columns=2:color-format=1:buf-sz=22:buf-initial-sz=21:buf-optimal-sz=20:' ! video/x-av1,parsed=True,stream-format=obu-stream
```

Листинг В – Конвейер для записи кадров, которые должны быть закодированы в ходе эксперимента

```
gst-launch-1.0 imagesrc name=videosrc remote=0 blocksize=16384 use-damage=0 show-pointer=true ! video/x-raw,framerate=60/1 ! queue leaky=downstream flush-on-eos=true max-size-time=40000000 max-size-buffers=0 max-size-bytes=0 ! identity sync=true ! videoconvertscale ! video/x-raw,format=i420,width=2560,height=1440,framerate=60/1 ! rawvideoparse format=i420 width=2560 height=1440 framerate=60/1 ! filesink location=/mnt/mini-demo/demo_desktop_mini_01.yuv
```

Листинг Г – Конвейер для чтения записанных “сырых” кадров

```
gst-launch-1.0 filesrc location=/mnt/mini-demo/demo_desktop_mini_03.yuv ! queue ! "video/x-raw, format=i420, width=2560, height=1440, framerate=30/1" ! rawvideoparse use-sink-caps=true ! timeoverlay ! autovideosink
```

Список литературы / References

- [1]. World Wide Web Consortium (W3C). WebRTC 1.0: Real-Time Communication Between Browsers. Available at: <http://www.w3.org/TR/webrtc/>, accessed 13.05.2025.
- [2]. Sheikholeslami A., Graffi K. A systematic quality analysis of virtual desktop infrastructure technologies // Euro-Par 2015: Parallel Processing Workshops. Lecture Notes in Computer Science, vol. 9523. Cham: Springer, 2015, pp. 335-346. DOI: 10.1007/978-3-319-27308-2_26.
- [3]. Chang C.-H., Yang C.-T., Lee J.-Y., Lai C.-L., Kuo C.-C. On construction and performance evaluation of a virtual desktop infrastructure with GPU accelerated // IEEE Access. 2020, vol. 8, pp. 170162-170173. DOI: 10.1109/ACCESS.2020.3023924.
- [4]. Li S., Xiao L., Shi C. et al. Boosting performance of virtualized desktop infrastructure with physical GPU and SPICE // Science China Information Sciences. 2020, vol. 63. Art. 179102. DOI: 10.1007/s11432-018-9914-5.
- [5]. Blakley J. R., Haas S., Firoiu V., Iyer M. B., Beveridge D., Narkhede G., George S. A., Eiszler T., Harkes J., Scott J. R., Satyanarayanan M. Deploying edge-based virtual desktop infrastructure. 2023.
- [6]. Simoens P., De Turck F., Dhoedt B., Demeester P. Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices // Proceedings of the Australasian Telecommunication Networks and Applications Conference (ATNAC), 2008, pp. 391-396.
- [7]. Alliance for Open Media. AV1 Tool Description. Version 1.1. Available at: https://aomedia.org/docs/AV1_ToolDescription_v11-clean.pdf, accessed: 19.08.2025.
- [8]. ITU-T and ISO/IEC JTC 1. Recommendation ITU-T H.265 (09/2023): High Efficiency Video Coding | ISO/IEC 23008-2:2020, Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High Efficiency Video Coding. Version 9 (Edition 9). Geneva: International Telecommunication Union (ITU) and International Organization for Standardization (ISO), 2023.
- [9]. Xu X., Liu S. Overview of Screen Content Coding in Recently Developed Video Coding Standards. IEEE Transactions on Circuits and Systems for Video Technology, 2022, vol. 32, no. 2, pp. 839-852. DOI: 10.1109/TCSVT.2021.3064210.
- [10]. Benteleb A., Lim M., Akcay M. N., Begem A. C., Hammoudi S., Zimmermann R. Toward One-Second Latency: Evolution of Live Media Streaming. IEEE Communications Surveys & Tutorials, DOI: 10.1109/COMST.2025.3555514.
- [11]. WebRTC Hacks. WebCodecs, WebTransport, and WebRTC: The Future Is Now. Available at: [http://webrthacks.com/webcodecs-webtransport-and-webrtc](http://webrtc hacks.com/webcodecs-webtransport-and-webrtc), accessed: 19.08.2025.
- [12]. Iyengar J., Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, Internet Engineering Task Force (IETF), 2021.
- [13]. Internet Engineering Task Force (IETF). Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT). Available at: <http://datatracker.ietf.org/doc/html/rfc8445>, accessed: 03.05.2025.
- [14]. Internet Engineering Task Force (IETF). Session Traversal Utilities for NAT (STUN). Available at: <http://datatracker.ietf.org/doc/html/rfc8489>, accessed: 03.05.2025.
- [15]. Internet Engineering Task Force (IETF). Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). Available at: <http://datatracker.ietf.org/doc/html/rfc8656>, accessed: 03.05.2025.
- [16]. Frindell A., Kinnear E., Vasiliev V. WebTransport over HTTP/3. Internet-Draft draft-ietf-webtrans-http3-12, Internet Engineering Task Force (IETF). Available at: <http://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http3-12>, accessed: 19.08.2025.
- [17]. Google Chrome Developers. WebTransport API Overview. Available at: <http://developer.chrome.com/docs/capabilities/web-apis/webtransport>, accessed: 19.08.2025.
- [18]. Cloudflare, Inc. Radar 2024 Year in Review. Available at: <https://blog.cloudflare.com/radar-2024-year-in-review/>, accessed: 19.08.2025.
- [19]. Li, B., Zou, J. (2019). Screen Content Coding: A Survey. In: Sun, S., Fu, M., Xu, L. (eds) Signal and Information Processing, Networking and Computers. ICSINC 2018. Lecture Notes in Electrical Engineering, vol 550. Springer, Singapore. DOI: 10.1007/978-981-13-7123-3_68.
- [20]. Advanced Micro Devices, Inc. AMF Video Encode API Documentation. Available at: http://github.com/GPUOpen-LibrariesAndSDKs/AMF/blob/master/amf/doc/AMF_Video_Encode_API.md, accessed: 19.08.2025.

- [21]. NVIDIA Corporation. NVENC Video Encoder API Programming Guide. Available at: <http://docs.nvidia.com/video-technologies/video-codec-sdk/12.2/nvenc-video-encoder-api-programming-guide/index.html>, accessed: 19.08.2025.
- [22]. Matrox Graphics Inc. Matrox M264 Family: Multi-Channel 8- and 10-Bit H.264 Codec Cards. Available at: <http://video.matrox.com/en/products/broadcast-media/H.264-codec-cards/m264>, accessed: 19.08.2025.
- [23]. Intel Corporation. Intel Media Driver for VAAPI. Available at: <https://github.com/intel/media-driver>, accessed: 19.08.2025.
- [24]. Microsoft Corporation. Getting Started with DirectX Graphics. Available at: <http://learn.microsoft.com/en-us/windows/win32/getting-started-with-directx-graphics>, accessed: 19.08.2025.
- [25]. Khronos Group. An Introduction to Vulkan Video. Available at: <http://www.khronos.org/blog/an-introduction-to-vulkan-video>, accessed: 19.08.2025.
- [26]. Carlucci G., De Cicco L., Holmer S., Mascolo S. Analysis and design of the google congestion control for web real-time communication (WebRTC). Proceedings of the 7th International Conference on Multimedia Systems, 2016, pp. 1-12.
- [27]. Zhu X., Pan R., Ramalho M.A., Mena S. Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media. RFC 8698, IETF, 2020. Available at: <https://datatracker.ietf.org/doc/rfc8698/>, accessed: 19.08.2025.
- [28]. Johansson I., Westerlund M. Self-Clocked Rate Adaptation for Multimedia (Internet-Draft). IETF, 2024. Available at: <https://datatracker.ietf.org/doc/draft-johansson-ccwg-rfc8298bis-screamv2/>, accessed: 19.08.2025.
- [29]. GStreamer. queue: Data queue. GStreamer Documentation. Available at: <https://gstreamer.freedesktop.org/documentation/coreelements/queue.html>, accessed 20.08.2025.
- [30]. RidgeRun. GstShark: Performance Analysis & Tracing for GStreamer Developers. Available at: <https://developer.ridgerun.com/wiki/index.php/GstShark>, accessed 20.08.2025.
- [31]. VideoLAN. x264: Free software library and application for encoding video streams into the H.264/MPEG-4 AVC compression format. Available at: <https://www.videolan.org/developers/x264.html>, accessed: 20.08.2025.
- [32]. YouTube Engineering. VP9: Faster, better, buffer-free YouTube. Available at: <https://web.archive.org/web/20150410001205/https://youtube-eng.blogspot.com/2015/04/vp9-faster-better-buffer-free-youtube.html>, publication date: april 2015, archiving date: 10.04.2015, accessed: 20.08.2025.
- [33]. Alliance for Open Media. AV1 Bitstream and Decoding Process Specification / Alliance for Open Media. — Wakefield, MA, USA: Alliance for Open Media, 2019. — Version 1.0.0 with Errata 1. Available at: <https://aomediacodec.github.io/av1-spec/av1-spec.pdf>, accessed: 13.05.2025.

Информация об авторах / Information about authors

Иван Сергеевич ПРОСКУРЯКОВ – программист отдела информационных систем Института системного программирования РАН. Сфера научных интересов: протоколы и стандарты передачи видео и аудио в режиме реального времени, стандарты кодирования видео, разработка методик отладки сложных информационных систем.

Ivan Sergeevich PROSKURIAKOV is a programmer at the Information Systems Department of the Institute of System Programming of the Russian Academy of Sciences. Research interests: protocols and standards for real-time video and audio transmission, video coding standards, development of debugging techniques for complex information systems.

Давид Арменович БАДАЛЯН – научный сотрудник Института системного программирования РАН. Сфера научных интересов: распределенные системы, облачные системы, системы передачи видео.

David Armenovich BADALYAN – research fellow at Institute for System Programming of the RAS. Research interests: distributed systems, cloud systems, video transmission systems.



Комплексное применение компьютерного зрения и распознавания речи для повышения качества обработки видеозаписей производственных работ

¹ И.Ю. Смолин, ORCID: 0000-0002-7272-1232 <ilya.smolin@ocrv.ru>

^{1,2} А.А. Любченко, ORCID: 0000-0002-2768-484X <aleksandr.lyubchenko@ocrv.ru>

¹ М.В. Исаков, ORCID: 0000-0002-8677-8862 <maksim.isakov@ocrv.ru>

¹ Д.А. Вождаева, ORCID: 0009-0001-2894-5810 <darya.vozhdaeva@ocrv.ru>

¹ ООО «ОЦРВ», филиал Сириус, холдинг «РЖД-Технологии», ФТ «Сириус»,
Россия, 354340, г. Сочи, ул. Международная, 2/1.

² Омский государственный технический университет (ОмГТУ),
Россия, 644050, г. Омск, пр. Мира, 11.

Аннотация. Сегодня технологии искусственного интеллекта находят свое применение в области научной организации труда и, в частности, применительно к технологическим процессам в сфере железнодорожного транспорта. Основной точкой получения эффекта в таком случае является использование технологий для автоматизации рутинного труда человека, направленного на просмотр и анализ видеозаписей производственных операций. Для достижения данного эффекта был разработан метод на основе технологий компьютерного зрения и обработки структурированной информации, которые легли в основу функционирующего сегодня программного обеспечения. Промышленная эксплуатация сервиса выявила необходимость в развитии его возможностей и повышении качества работы. Коллективом авторов был разработан усовершенствованный подход для анализа видеозаписей производственных процессов путем совместного использования технологий компьютерного зрения и распознавания речи. Предложенное решение включает детекцию и трекинг сотрудников, распознавание специализированных инструментов и действий работников, а также обработку речевых комментариев оператора съемки. Интеграция данных из различных информационных каналов позволяет повысить точность определения временных интервалов технологических операций, компенсируя ограничения при одноканальной обработке существующего решения. Проведенные лабораторные эксперименты показали, что применение современных архитектур нейронных сетей и предложенных алгоритмов постобработки обеспечивает 40-процентный прирост качества анализа видеозаписей производственных работ.

Ключевые слова: компьютерное зрение; распознавание речи; детекция объектов; трекинг; железнодорожная отрасль; видеоаналитика.

Для цитирования: Смолин И.Ю., Любченко А.А., Исаков М.В., Вождаева Д.А. Комплексное применение компьютерного зрения и распознавания речи для повышения качества обработки видеозаписей производственных работ. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 201–220. DOI: 10.15514/ISPRAS-2026-38(1)-14.

Integrated Application of Computer Vision and Speech Recognition to Improve the Quality of Processing Video Recordings of Production Works

¹ I.U. Smolin, ORCID: 0000-0002-7272-1232 <ilya.smolin@ocrv.ru>

^{1,2} A.A. Lyubchenko, ORCID: 0000-0002-2768-484X <aleksandr.lyubchenko@ocrv.ru>

¹ M.V. Isakov, ORCID: 0000-0002-8677-8862 <maksim.isakov@ocrv.ru>

¹ D.A. Vozhdaeva, ORCID: 0009-0001-2894-5810 <darya.vozhdaeva@ocrv.ru>

¹ LLC «OCRВ», Sirius Branch, holding «RZD-Technologies», FT Sirius,
2/1, Mezhdunarodnaya st., 354340, Russia.

² Omsk State Technical University (OmSTU),
11, Mira st., Omsk, 644050, Russia.

Abstract. Artificial intelligence technologies are increasingly applied to the scientific organization of labor, in particular to the technological processes of the railway industry. The primary benefit in this context comes from automating the routine human work of reviewing and analyzing video recordings of production operations. To realize this benefit, an initial method based on computer-vision techniques and structured-information processing was developed, and has since underpinned a functioning software. Industrial exploitation of the feature revealed the need for enhanced capabilities and improved performance. In response, the authors have devised an advanced approach that jointly employs a complex of computer-vision and speech-recognition technologies. The proposed solution includes employee detection and tracking, recognition of specialized tools and worker actions, and processing of the camera operator's verbal commentary. By integrating data from multiple information channels, it increases the accuracy of temporal-interval determination for technological operations, compensating for the limitations of single-channel analysis. Laboratory experiments demonstrate that the use of modern neural-network architectures and the newly proposed post-processing algorithms delivers a 40% gain in the quality of video-analysis of production workflows.

Keywords: computer vision; speech recognition; object detection; tracking; railway industry; video analytics.

For citation: Smolin I.U., Lyubchenko A.A., Isakov M.V., Vozhdaeva D.A. Integrated application of computer vision and speech recognition to improve the quality of processing video recordings of production works. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 201-220 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-14.

1. Введение

В рамках развития научной организации труда в железнодорожной отрасли, где совершенствование механизмов нормирования является одной из ключевых задач [1], применяются различные методы изучения рабочего времени. В последние годы, для решения этого вопроса, все чаще используются современные информационные технологии видеоанализа на базе искусственного интеллекта [2].

Видеосъемка рабочего времени, дополненная алгоритмами компьютерного зрения, представляет собой перспективный метод исследования производственных процессов, позволяя не только фиксировать временные затраты, но и детально анализировать технологические операции, действия сотрудников и идентифицировать применяемые инструменты [3]. В отличие от классических методов нормирования труда, видеосъемка разделяет процесс замера показателей и их анализ, обеспечивая объективность измерений и возможность многократной проверки в соответствии с принципами научной организации труда. Такой подход обеспечивает комплексный анализ производственных процессов с высокой степенью детализации.

Развитие технологий видеоаналитики демонстрирует растущий интерес к автоматизации мониторинга трудовых процессов. Существуют разработки [4], которые используют несколько камер для мониторинга ручных операций и выявления аномальных событий.

Также разработана система [5], фиксирующая события, связанные с поведением и действиями сотрудников. В работе [6] предлагается метод распознавания последовательности действий рабочих на стройплощадках на основе данных с камер видеонаблюдения. Промышленное внедрение подобных систем демонстрируется в работе [7], решение осуществляет контроль безопасности на производстве и анализ поведения персонала.

Определение интервалов технологических операций тесно связано с задачей Temporal Action Localization (TAL) [8], которая заключается в определении временных границ выполнения действий на видео. Однако, прямое применение этой технологии в железнодорожной отрасли ограничено несколькими факторами. Во-первых, методы TAL часто ориентированы на распознавание типовых, достаточно простых действий [9], тогда как, например, путевые работы включают множество нюансов и вариаций выполнения операций. Во-вторых, не учитывается необходимость одновременного анализа нескольких аспектов: действий сотрудников, применяемого инструмента, пространственного положения и производственного контекста.

Предыдущие исследования [10] решали задачу оценки временных интервалов технологических операций исключительно на основе детекции применяемых инструментов, где алгоритм оперирует ограниченным объемом информации. Такой подход был назван одноканальным. Впоследствии были выявлены некоторые его ограничения и недостатки: проблемы с обнаружением инструментов из-за различных перекрытий, невозможность определения неоперативного времени (когда инструмент в руках, без явной работы с ним), а также отсутствие индивидуального распределения времени операций в случае одновременной работы нескольких человек.

Для повышения точности и надежности анализа временных операций предлагается использовать альтернативный вариант, объединяющий разные модальности данных: визуальную информацию (детекция сотрудников и их действий, определение рабочих инструментов), аудиоданные (распознавание голосовых команд оператора съемки). И такой подход было предложено назвать многоканальным. Идея интеграции мультимодальных данных позволит создать более точную и адаптивную технологию для нормирования труда, учитывающую реальные условия путевых работ.

Автоматизация процесса визуальных наблюдений с использованием технологий компьютерного зрения и распознавания речи позволит минимизировать влияние человеческого фактора, повысить точность измерений и усовершенствовать методы научной организации труда в железнодорожной сфере.

2. Предлагаемое решение

В работе представлена реализация собственного решения для анализа затрат рабочего времени по видеосъемкам с применением методов глубокого обучения в области компьютерного зрения и распознавания речи. Предлагаемый алгоритм, с учетом специфики железнодорожной отрасли, обеспечивает обработку видео и аудиоинформации с высокой точностью. Отличительной особенностью технологии является применение многоканальной архитектуры, обеспечивающей более стабильную работу по сравнению с одноканальным вариантом.

При реализации предложенного алгоритма были выделены следующие основные прикладные задачи:

- 1) детекция и трекинг сотрудников;
- 2) распознавание отраслевых инструментов;

- 3) привязка инструментов к сотруднику;
- 4) распознавание действий;
- 5) распознавание речевых комментариев;
- 6) постобработка результатов.

Для точного анализа трудовых процессов и распределения норм времени необходимо детектировать и идентифицировать каждого сотрудника. Также нужно уделить особое внимание стабильности работы алгоритмов в условиях изменения ракурсов съемки, возможных перекрытий и наличия схожей спецодежды, которая затрудняет идентификацию. При выполнении путевых работ требуется надежно детектировать 35 специализированных инструментов в следующих возможных условиях: плохое освещение, атмосферные осадки, тряска камеры и частичные окклюзии. Главное требование – точное установление факта использования инструмента.

В моменты, когда человек заслоняет собой инструмент, необходимо распознавать действия сотрудника, тем самым делать вывод о работе с конкретным инструментом по особым движениям, характерным для того или иного инструмента.

Если с помощью детектирования инструмента и выполняемых действий невозможно определить временные рамки операций, то дополнительно необходимо обрабатывать речевые комментарии оператора съемки, сопровождающие начало и конец операций.

На финальной стадии формирования оценок временных интервалов, требуется предусмотреть алгоритм постобработки, который обработает результаты на основе заданной логики для компенсации погрешностей по отдельным блокам.

Разрабатываемое решение представляет собой многоканальную систему анализа видео, алгоритм работы которой может быть визуально представлен на рис. 1.

На первом этапе видеофайл предварительно предобрабатывается, разбивается на отдельные кадры с необходимым масштабированием (а). Эти кадры затем параллельно поступают в два независимых модуля детектирования инструментов (б) и детектирования сотрудников (в). Данные модули отвечают за обнаружение и локализацию объектов в кадре. Далее предсказания этих блоков (`bbox_tool`, `class_tool`, `bbox_person`, `idx_person`, `pose`) агрегируются и передаются в модуль привязки инструмента к сотруднику (г). Этот этап позволяет определить, держит ли сотрудник конкретный инструмент в руках или он лишь находится в зоне видимости, например, лежит на земле. Это важно для понимания контекста действий человека.

Параллельно результаты работы блока (в) передаются в модуль распознавания действий (д), который анализирует активность сотрудников, например, определяет, бездействует ли человек, работает с инструментом или выполняет другие действия. Ответом модуля является ограничивающий прямоугольник сотрудника и его индекс с предыдущего модуля, а также и распознанное действие (`bbox_person`, `idx_person`, `action`).

Помимо обработки визуальных данных, система также включает модуль распознавания голоса (е), который работает напрямую с аудиодорожкой видеофайла, минуя этапы обработки изображений и выдает временные интервалы работы с инструментом (`class_tool`, `start_time`, `end_time`).

На финальном этапе все полученные данные — результаты распознавания голоса (е), распознавания действий (д) и привязки инструментов к сотрудникам (г) — поступают в модуль постпроцессинга (ж). Здесь информация интегрируется, анализируется и преобразуется в финальный результат: временные интервалы с указанием операций, выполняемых каждым сотрудником.

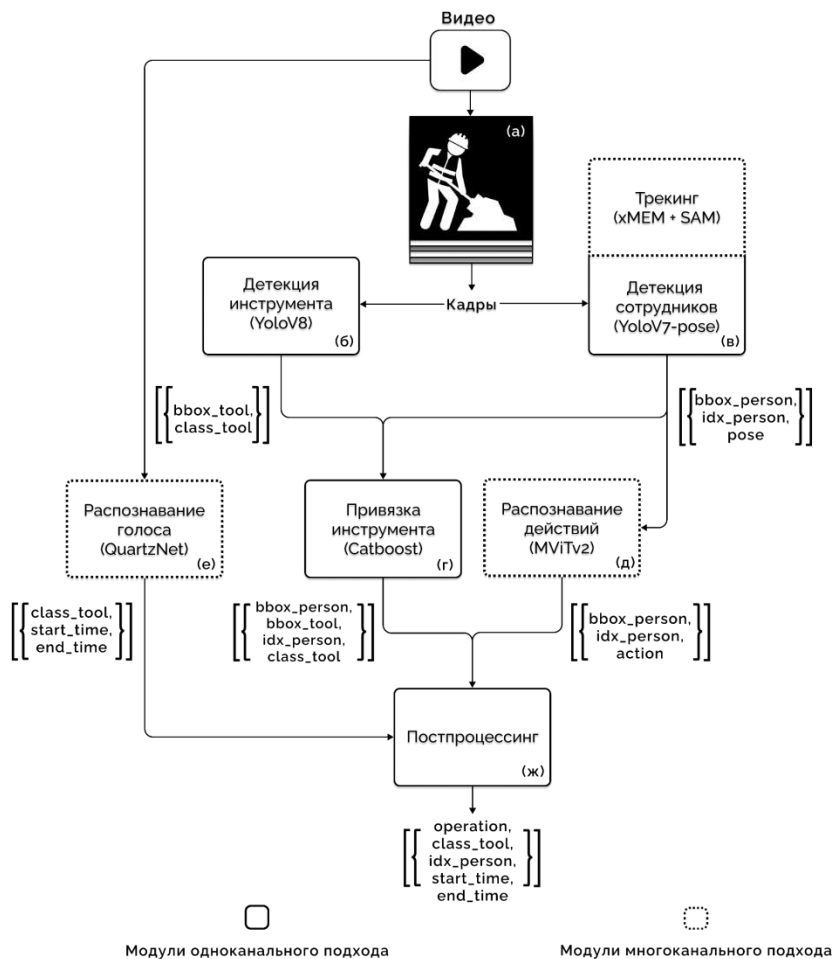


Рис. 1. Схема основных функциональных блоков архитектуры.

Fig. 1. Diagram of the main functional blocks of the architecture.

2.1 Детектирование сотрудников

Для решения задачи определения сотрудников в кадре был выбран подход на основе детекции объектов с ключевыми точками, а также добавление трекинга людей [11].

В качестве детекционной модели используется предобученная модель семейства Yolo – YoloV7-pose [12], которая не только возвращает координаты ограничивающего прямоугольника, но и определяет координаты ключевых точек. Такие точки характеризуют различные значимые части объекта, являющиеся ориентирами для анализа его структуры и позиции. Положения ключевых точек обычно представляются в виде набора 2D-координат $[x, y]$ или 3D-координат $[x, y, \text{уверенность}]$. В стандартной модели YoloV7-pose имеется 17 ключевых точек, каждая из которых представляет собой отдельную часть человеческого тела. В нашей задаче используются только 11 точек (плечи, кисти, бедра, колени, глаза и нос).

Полученные данные о ключевых точках используются в дальнейшем на этапе привязки инструмента к человеку (рис. 1, г). Кроме того, точки глаз и носа применяются для определения ориентации человека относительно камеры. Если координаты глаз и носа имеют высокую уверенность детекции, это указывает на то, что точки видны и человек стоит лицом

к камере. Если уверенность низкая, делаем вывод, что человек стоит спиной к камере и возможно заслоняет собой инструмент, с которым в данный момент работает.

Задача трекинга объектов в компьютерном зрении, особенно в контексте отслеживания движения людей, остается одной из наиболее сложных и актуальных проблем. Это связано с такими вызовами, как окклюзии, пересечения объектов, выходы из кадра и последующие возвращения, которые приводят к ошибкам в идентификации. Традиционные методы трекинга часто оказываются недостаточно устойчивыми в таких сценариях, что ограничивает их применение в реальных условиях.

В данной работе рассматриваются новые подходы к решению задачи трекинга, направленные на улучшение реидентификации объектов и повышение устойчивости алгоритмов в сложных условиях. Основным решением данных проблем является использование комбинаций современных архитектур, таких как xMEM [13] и SAM [14].

В рамках исследования были протестированы следующие подходы трекинга:

- **Tracking by Face Recognition.** Подход предполагает использование распознавания лиц как основы для построения эмбедингов, которые можно использовать в алгоритме реидентификации для сравнения. Качество такого подхода на тестовых данных оказалось низким из-за недостаточной устойчивости признаков, что в свою очередь обусловлено наличием в кадре средств индивидуальной защиты, таких как очки и кепки.
- **Tracking by Detection.** Комбинация YoloV7-Pose в качестве детектора и алгоритма ByteTrack [15]. Результат – недостаточная эффективность при резких изменениях позы и входе/выходе из кадра на более чем 3 секунды.
- **Tracking by Segmentation.** Совместное использование xMEM, YoloV7-Pose и SAM. Является перспективным вариантом, но требует доработки для повышения устойчивости работы при частых окклюзиях и выходах объектов трекинга из кадра.
- **Tracking by Segmentation** представляет собой новый алгоритм, сочетающий xMEM для реидентификации (bboxes ID), YoloV7-Pose для детекции ключевых точек и ограничивающих прямоугольников (keypoints, bboxes) и SAM для сегментации с получением контуров объектов (masks). Взаимосвязь основных функциональных блоков адаптированного под рассматриваемую задачу алгоритма представлена на рис. 2.

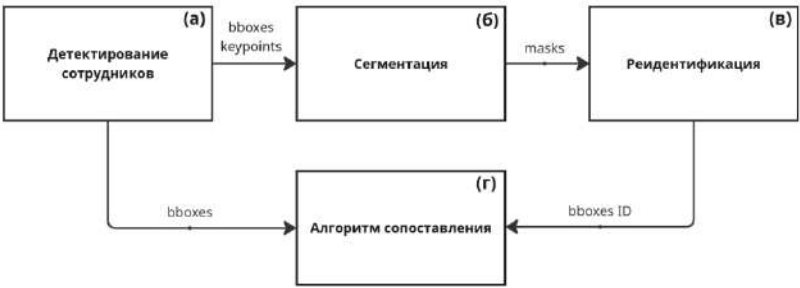


Рис. 2. Предложенный алгоритм трекинга сотрудников.
Fig. 2. The proposed employee tracking algorithm.

Более детально предлагаемый алгоритм трекинга может быть представлен следующей последовательностью логических операций:

Вход: Список кадров входной видеозаписи *video_frames*

Выход: Ограничивающие прямоугольники с идентификаторами *bboxes_id*

Начало алгоритма трекинга

Инициализация моделей:

Инициализировать *xMEM*

Инициализировать *YOLOv7-Pose*

Инициализировать *SAM*

for каждый кадр *f* из *video_frames* **do**

Шаг 1: Детекция объектов с использованием *YOLOv7-Pose*

$(bboxes, keypoints) \leftarrow YOLOv7_Pose.detect(f)$

Шаг 2: Сегментация объектов с использованием *SAM*

$masks \leftarrow SAM.segment(f)$

Шаг 3: Реидентификация объектов с использованием *xMEM*

for каждый *bbox_i* из *bboxes* **do**

$bboxes_id \leftarrow xMEM.reidentify(bbox_i, f)$

$bboxes_id \leftarrow bboxes_id \cup \{object_id, bbox_i, keypoints, masks_i\}$

end for

Шаг 4: Обработка окклюзий и выхода объектов за пределы кадра

for каждый объект *obj* из *bboxes_id* **do**

if *obj.is_occluded()* или *obj.is_out_of_frame()* **then**

$obj.status \leftarrow "occluded"$

$continue$

end if

$obj.update_position(object_id, keypoints, masks)$

end for

Шаг 5: Сохранение результатов трекинга

Сохранить *bboxes_id* для дальнейшего анализа

end for

Конец алгоритма трекинга

Для всесторонней и комплексной проверки качественных характеристик, все исследуемые подходы были протестированы по трем метрикам NOTA [16], MOTA [17] и IDF1 [18], которые являются стандартом в современных бенчмарках для алгоритмов трекинга [19]. Каждая из данных метрик фокусируется на определенных аспектах производительности алгоритмов трекинга в динамических сценах:

- Higher Order Tracking Accuracy (NOTA) — оценка общего баланса между точностью и полнотой трекинга, учитывает как ошибки в позициях объектов, так и в их идентификации;
- Multiple Object Tracking Accuracy (MOTA) — метрика, оценивающая точность трекинга, принимая во внимание количество ошибок в трекинге, такие как пропуски, ложные срабатывания и неправильное сопоставление идентификаторов;
- Identification F1 Score (IDF1) — метрика, оценивающая качество идентификации объектов, которая вычисляется как гармоническое среднее между точностью и полнотой в отношении правильных идентификаций объектов в процессе трекинга.

Тестовый набор состоял из 20 видеозаписей, содержащий как съемки реального производственного процесса (рис. 3, а) общей длительностью 1 час, так и записи, подготовленные в лабораторных условиях (рис. 3, б) длительностью 30 минут. Видеоматериалы различаются по уровню сложности трекинга и включают следующие сценарии:

- сложность 1 – сотрудники в кадре без сильных окклюзий и без выхода за пределы кадра;
- сложность 2 – трекинг сотрудников при частичном и полном перекрытии объектов друг другом или элементами окружения;

- сложность 3 – трекинг с реидентификацией вследствие выхода сотрудников за кадр на различные интервалы по времени (менее 5 с, 5-10 с, 10-30 с и более 30 с).

Таким образом, выбранный набор тестовых видеозаписей позволял оценить устойчивость алгоритмов трекинга к визуальным сложностям в производственных условиях. Результаты работы на тестовом наборе данных, представлены в табл. 1.

Подход Tracking by Segmentation показал высокие результаты на тестовых данных и в результате был выбран в качестве основного алгоритма трекинга людей.



a)



b)

Рис. 3. Примеры кадров из видеозаписей реального производственного процесса (a) и в лабораторных условиях (b).

Fig. 3. Examples of test video frames in real process (a) and lab conditions (b).

Табл. 1. Метрики работы различных подходов трекинга на тестовом наборе данных.

Table 1. Performance metrics of different tracking approaches on test dataset.

Подходы	HOTA (%)↑	MOTA (%)↑	IDF1 (%)↑
Tracking by Face Recognition	40,9	45,2	30,4
Tracking by Detection	64,5	67,4	75,6
Tracking by Segmentation	80,2	89,1	94,2

2.2 Детектирование инструментов

Для решения задачи детектирования инструментов используется модель также семейства Yolo – YoloV8, которая имеет приемлемое качество работы, быструю скорость инференса, а также легко интегрируется в большинство решений.

Для обучения модели использовались изображения из видеозаписей производственного процесса. Набор данных содержит 60 тыс. изображений с различными углами обзора, расстоянием до объекта, освещением и погодными условиями. Распределение на обучающую, валидационную и тестовую выборки было задано как: 70/20/10. Исходные данные включают 35 классов инструментов, что обеспечивает широкий охват большей части

путевых работ, около 95%. Среди классов представлены такие специализированные инструменты, как ключ торцовый, кран порталный, станок рельсошлифовальный и другие. Их внешний вид, массогабаритные характеристики осложняют задачу детектирования в сравнении с повседневными предметами. Для наглядной демонстрации внешнего вида детектируемых объектов, отдельным документом [20] представлен список всех учитываемых в алгоритме инструментов. Материал подготовлен на основе изображений, доступных в сети Интернет.

Чтобы улучшить обобщающую способность нейросети и обеспечить работу в сложных условиях, к стандартным аугментациям Yolo были добавлены погодные аугментации (снег, дождь, туман). Примеры изменения изображений показаны на рис. 4.



Рис. 4. Пример погодных аугментаций.

Fig. 4. Example of weather augmentation.

Кроме того, часть данных была изменена с помощью генеративных сетей – CosXL [21] и InstructPix2Pix [22]. В качестве примера были применены такие запросы как: «измени время года на осень», «добавь закат» или «смени время суток на вечернее» и т.д. Пример преобразования изображений с запросом «время года осень» представлен на рис. 5. Еще одним экспериментом по изменению данных было применение переноса стиля с одного изображения на другое [23]. Данная технология улучшает взаимодействие моделей детекции объектов с низкоуровневыми признаками, такими как текстуры, цвета и небольшими изменениями форм.

Результаты экспериментов представлены в табл. 2.

2.3 Привязка инструмента к сотруднику

Результаты работы описанных выше блоков после преобразования в табличный формат передаются в сервис, отвечающий за привязку найденных инструментов к сотрудникам. Это необходимо для того, чтобы, проанализировав каждый кадр видео, определить, находится ли инструмент в руках сотрудника или нет.

Сервис детекции человека агрегирует координаты ограничивающего прямоугольника, его ключевые точки и уникальный индекс человека из модуля трекинга. Сервис детекции инструментов отдает координаты найденного инструмента и его класс. Поскольку на одном кадре может быть несколько инструментов и несколько человек, в таблицу заносятся все возможные комбинации «инструмент-человек». Детектирование инструмента в руках человека осуществляется на основе всех признаков, полученных от других сервисов, а также

дополнительными признаками расстояний между ключевыми точками человека и координатами центра инструментов (рис. 6). Для предсказания используется CatBoost [24] – модель в виде ансамбля решающих деревьев, обученная методом градиентного бустинга. Алгоритм вычисляет вероятность того, что конкретный инструмент находится в руках сотрудника.



Рис. 5. Генеративная аугментации «время года осень».
Fig. 5. Generative augmentation "autumn season".

Табл. 2. Метрика детекции на тестовом наборе данных.
Table 2. Detection metric on the test dataset.

Эксперименты	mAP50↑ (среднее)
YoloV8, стандартные данные	0,682
+ погодные аугментации	0,701
+ генеративные аугментации	0,725

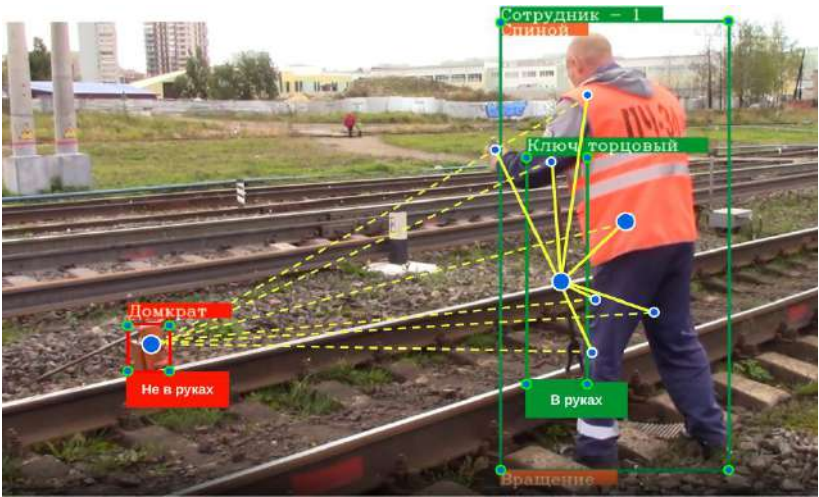


Рис. 6. Визуализация принципа работы алгоритма привязки инструмента к рукам человека.
Fig. 6. Visualization of the tool-binding algorithm for human hands.

На отложенной тестовой выборке среднее значение целевой метрики (F1-мера) по всем инструментам составило 0,89.

2.4 Распознавание действий

Необходимость точно определять инструменты в блоке (в) на рис. 1. в условиях сильных окклюзий и съёмки со спины, является одной из нерешенных проблем при одноканальном подходе. Для этого в общую архитектуру решения был включен канал распознавание действий, блок (д) на рис. 1.

Задача распознавания действий (англ. Action Recognition) в компьютерном зрении представляет собой процесс автоматической идентификации и классификации действий, совершаемых людьми или объектами на основе анализа видеопоследовательностей или наборов изображений. Данная проблема находит широкое применение в различных прикладных областях, что стимулирует разработку многочисленных методик и алгоритмов для ее решения [25].

В рамках решаемой задачи ключевая функция данного канала – это определение действий, имеющих характерные движения рук, ног и положения тела при работе с различными инструментами. Было выделено 4 класса действий: вращательное, ударное, закручивающее, толкательно-поступательное, а также отдельно класс «без действия». Поясняющая визуализация сформированных классов приведена на рис. 7.

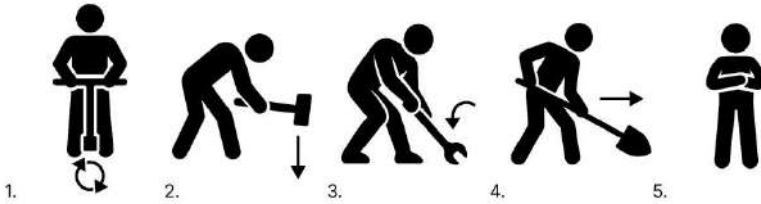


Рис. 7. Классы распознаваемых действий: вращательное (1), ударное (2), закручивающее (3), толкательно-поступательное (4), без действия (5).

Fig. 7. Classes of recognized actions: spinning (1), striking (2), tightening (3), push-pull (4), inactivity (5).

Основные подходы к решению данной задачи можно разделить на несколько категорий, каждая из которых имеет свои особенности, преимущества и недостатки. В ходе проведённых исследований были рассмотрены три подхода. Во-первых, это использование двухпоточной архитектуры I3D [26], которая с помощью RGB-потока и Optic Flow потока, позволяет модели анализировать как пространственные, так и временные особенности. Второй подход PoseC3D [27], который использует 3D-позы человека в качестве основной информации для получения результата распознавания. Последний апробированный вариант – это MViTv2 [28], трансформерная модель с самым большим количеством параметров.

Результаты тестирования и расчёта основных метрик качества для сравниваемых подходов показаны в табл. 3. Наиболее привлекательно по метрикам на отложенном тесте показала себя модель MViTv2, которая и была выбрана для канала распознавания действий. Более высокое качество работы данной модели обусловлено использованием механизма временного внимания.

Табл. 3. Метрики на тестовом наборе данных исследуемых подходов.

Table 3. Performance metrics on the test dataset.

Подход	Accuracy Top1↑	Accuracy Top3↑	Macro F1↑
I3D	0,36	0,77	0,30
PoseC3D	0,63	0,86	0,54
MViTv2	0,74	0,90	0,70

2.5 Распознавание речевых комментариев

В качестве дополнительного источника информации для определения временных границ операций было предложено использовать речевые комментарии оператора съёмки, содержащие указания на начало и завершение работы определенным инструментом.

Для однозначного определения операций использовался согласованный речевой регламент, содержащий шаблонизированные фразы вида:

работу <инструментом> <начал/закончил>,

а также их вариации с использованием синонимов и перестановки слов.

Для задачи распознавания такого ограниченного набора фраз оптимальным выбором стала компактная модель QuartzNet [29] (19 млн. параметров), предобученная на наборе данных Golos [30] объемом 1227 часов. Она позволяет подключить внешнюю n-gram языковую модель для перевзвешивания гипотез акустической модели, а также легко вычислять временные метки для слов, что необходимо в решаемой задаче.

При дообучении использовались:

- синтетические данные, сгенерированные моделью Text-to-Speech фразы из радиопереговоров на транспорте (30 часов) для адаптации к специфике задачи;
- часть набора данных Golos (60 часов) для сохранения возможности модели распознавать реальную человеческую речь.
- А также дополнительные аугментации:
 - silence – добавление тишины в начале записи;
 - white noise – добавление белого шума к сигналу;
 - noise – добавление железнодорожного шума и шума ветра из внешних аудиофайлов;
 - RIR – свертка с Room Impulse Response для имитации отражений звука.

Для тестирования использовались аудиодорожки из 23 видеозаписей (40 минут). В табл. 4 приведены значения Word Error Rate (WER) на тестовых аудиозаписях. Во всех экспериментах использовались аугментации Silence, WhiteNoise и Noise. Достигнуто хорошее качество при записи речи на небольшом расстоянии от микрофона.

Табл. 4. WER на тестовом наборе данных. Farfield и nearfield – подмножества тестовых записей, в которых говорящий находится на расстоянии и вблизи микрофона соответственно.

Table 4. WER on test dataset. Farfield/nearfield indicate distant/close microphone-speaker positioning.

Эксперимент	WER↓	WER↓ farfield	WER↓ nearfield
QuartzNet Golos	0,34	0,69	0,16
Дообучение только на синтетических данных с «замороженным» энкодером	0,68	0,93	0,56
+ данные Golos farfield	0,24	0,48	0,12
+ разморозка энкодера	0,19	0,39	0,08
+ аугментация RIR	0,17	0,36	0,07

На рис. 8 представлен полный алгоритм обработки аудиозаписи для извлечения временных границ операций. После описанного выше этапа распознавания речи (а) с вычислением временных отсечек слов осуществляется сегментация распознанного текста на фразы на 212

основе анализа пауз в речи (б). Критерием разбиения выступает значительное превышение длительности межфразовых пауз над внутрифразовыми. На следующем этапе (в) из текста каждой фразы извлекаются названия инструментов и слова-индикаторы начала или завершения операции («начал», «закончил» и тому подобные). В завершении (г) осуществляется сопоставление фраз с совпадающими инструментами и метками начала и конца операции с пересчетом временных меток.

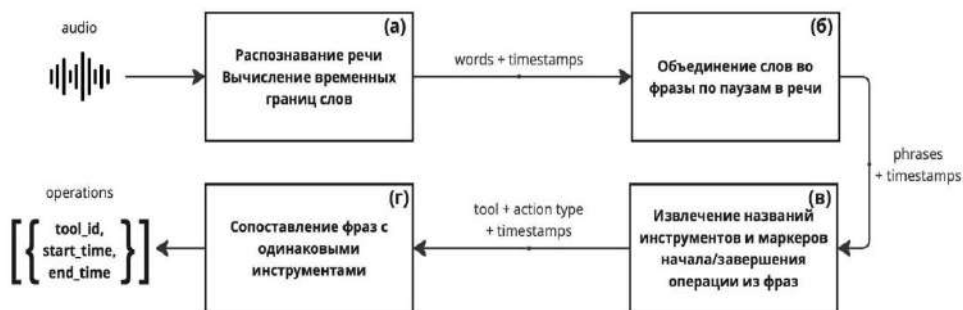


Рис. 8. Алгоритм извлечения временных границ операций из речевых комментариев.
Fig. 8. Algorithm for extracting temporal boundaries of operations from speech.

Финальным результатом работы пайплайна является список операций с идентификаторами инструментов, временем начала и конца операции.

Пример визуализации результата на тестовом аудиофайле представлен на рис. 9.

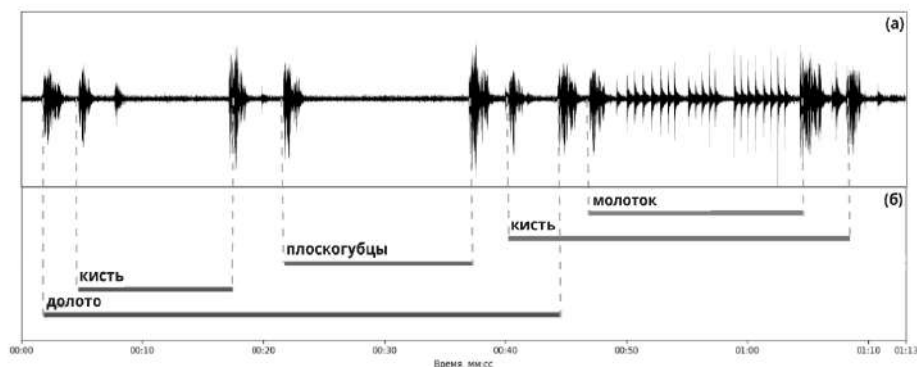


Рис. 9. Визуализация работы пайплайна извлечения временных границ операций:
(а) тестовый аудиофайл, (б) речевые комментарии с выделенными границами.
Fig. 9. Visualization of the extraction pipeline of operations temporal boundaries:
(a) test audiofile, (b) speech comments with the identified boundaries.

2.6. Постобработка результатов

Финальным и особенно важным этапом обработки видеозаписей является блок постпроцессинга (ж) на рис. 1, отвечающий за агрегацию и согласование предсказаний, поступающих от всех предыдущих блоков: детекции инструмента в руках сотрудника, распознавания его действий, а также анализа речевых комментариев. Цель этого этапа – формирование точных, непротиворечивых и интерпретируемых временных интервалов операций с привязкой к конкретным сотрудникам.

Последовательность шагов алгоритма финальной постобработки предсказаний представлена на рис. 10.

Покадровые результаты блока привязки инструмента к сотруднику и распознавания действий (action, tool_id, frame_num) конвертируются во временные интервалы (start_time, end_time), далее происходит корректировка по следующему алгоритму. Если между двумя операциями одного класса и одного сотрудника фиксируется пауза менее 10 секунд, они объединяются в один более протяженный интервал. Также, короткие по длительности интервалы (менее 2 секунд) удаляются как вероятные ложные срабатывания. Далее используется логика согласования – например, начало операции фиксируется при одновременном наличии сигнала из речевого канала («начал работу»), визуального подтверждения действия (характерное движение для конкретного инструмента) и наличия инструмента в руках. На основе последней обработки выдаются итоговые интервалы операций (operation, start time, end_time), которые благодаря системе трекинга индивидуальны для каждого сотрудника.

Пример временных интервалов с разбивкой по информационным каналам, а также сравнение с ручной разметкой (GT – ground true) показан на рис. 11.

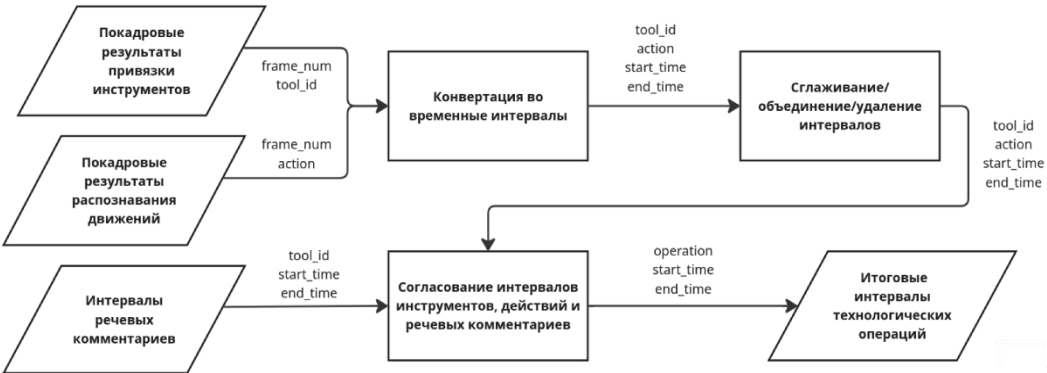


Рис. 10. Алгоритм постобработки предсказаний.
Fig. 10. Prediction post-processing algorithm.

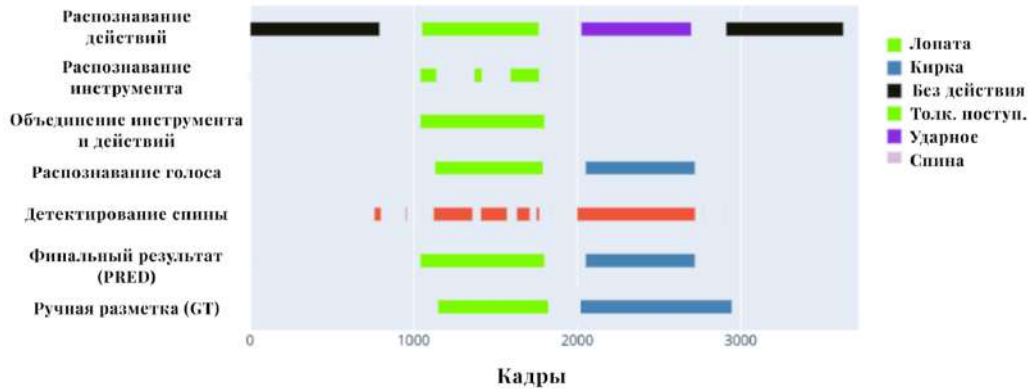


Рис. 11. Визуализация временные интервалов операций при многоканальном подходе.
Fig. 11. Visualization of multi-channel operation timings.

Как видно на изображении, в течение большей части рабочего процесса сотрудник находился спиной к камере, поэтому инструменты удавалось детектировать лишь в отдельных коротких промежутках. Однако, благодаря дополнительным данным – распознаванию действий и голосовых комментариев – технология позволила относительно точно определить временные границы выполняемых операций.

2.7. Программная реализация

Разработанное решение реализовано в виде программного комплекса на основе микросервисной архитектуры. Все функциональные блоки развертываются в изолированных Docker контейнерах и оркестрируются с помощью Docker Compose, что обеспечивает воспроизводимость вычислительного окружения, гарантирует переносимость решения между различными платформами и существенно упрощает процессы сопровождения и масштабирования приложения. Все сервисы реализованы на языке программирования Python.

Взаимодействие между сервисами осуществляется через REST API по протоколу HTTP с использованием формата данных JSON.

Обучение и инференс моделей выполнялись с использованием графического ускорителя 1xNVIDIA Tesla V100 с объёмом видеопамати 32GB. В качестве основных библиотек глубокого обучения применялись PyTorch и Ultralytics. Для распознавания действий сотрудников использовался фреймворк MMAction2. Предобработка видеопотоков и изображений осуществлялась на основе библиотеки OpenCV. Постобработка предсказаний моделей выполнялась с помощью библиотеки анализа данных Pandas.

Для хранения и управления данными системы использовалась объектно-реляционная база данных PostgreSQL.

2.8. Тестирование решения

В лабораторных условиях были смоделированы производственные работы с использованием реальных путевых инструментов и спецодежды, применяемой в хозяйстве. Для проверки предложенного подхода и количественной оценки его качества работы было снято и размечено 27 тестовых видео, длительностью 2-4 минуты каждое. Сценарии видеозаписей предусматривали различные осложняющие работу моделей случаи: перекрытия инструментов, выход сотрудника за кадр, периоды активных действий с инструментом и периоды простоя, а также голосовое сопровождение выполняемых операций.

Эксперименты были организованы таким образом, чтобы постепенно добавлялась информация для оценки временных интервалов операций, начиная только с детекции инструментов (одноканальный подход) с расчетом метрики качества F1-меры. Результаты вычислительных экспериментов для тестирования решения представлены в табл. 5, на основании которой можно сделать вывод о положительной динамике роста целевой метрики с 0,59 до 0,83 благодаря дополнительной информации, извлекаемой модулями предложенной многоканальной архитектуры.

Табл. 5. Результаты тестирования решения при добавлении информации.

Table 5. Testing results of the solution with added information.

Эксперимент		F1-мера↑
1 канал	Детекция инструментов	0,59
N каналов	Детекция инструментов + Распознавание действий	0,66
	Детекция инструментов + Распознавание действий + Распознавание речи	0,71
	Детекция инструментов + Распознавание действий + Постпроцессинг	0,80
	Детекция инструментов + Распознавание действий + Распознавание речи + Постпроцессинг	0,83

3. Обсуждение результатов

Проведенные экспериментальные исследования подтверждают гипотезу о преимуществе многоканального подхода в сравнении с первоначальной одноканальной реализацией алгоритма видеоанализа производственных процессов путевого комплекса. Как показали результаты тестирования в табл. 5, интеграция двух модальностей – визуальной (детекция инструментов, трекинг персонала, распознавание действий) и аудиальной (распознавание речевых комментариев) – обеспечивает значительный прирост качества, F1-мера возросла с 0,59 до 0,83, что соответствует 40 процентному улучшению точности определения временных интервалов операций. Особенно важно, что предложенная архитектура демонстрирует устойчивость в условиях, типичных для рассматриваемого железнодорожного хозяйства: частичные окклюзии, нестабильное освещение, схожая спецодежда сотрудников, тряска камеры при съемке с рук.

Следует отметить, что ключевым фактором повышения качества стала не просто параллельная обработка данных из разных источников, а их логическая согласованность на этапе постобработки. Модуль постпроцессинга, реализующий правила, близкие к экспертной оценке, позволил эффективно фильтровать ложные срабатывания и корректировать временные границы операций. Кроме того, использование речевых комментариев оператора съёмки в качестве надёжного внешнего источника информации о начале и завершении операций оказалось особенно эффективным в ситуациях, когда визуальный канал оказывался недостаточным, например, при съемке со спины.

Однако следует признать, что предложенное решение сохраняет определённые ограничения, особенно в части распознавания речи. Как показала оценка частоты ошибок на уровне слов (WER) в табл. 4, качество распознавания заметно снижается при увеличении расстояния между видеокамерой и целевой сценой с производственным персоналом (WER farfield = 0,36). Это может ограничивать применимость решения в реальных полевых условиях, где расстояние между фиксируемой сценой и видеокамерой с микрофоном может значительно варьироваться ввиду необходимости сохранения безопасных условий труда специалистов и следования регламенту выполняемого производственного процесса.

Для устранения данного ограничения альтернативным решением может стать применение end-to-end моделей распознавания речи с встроенной устойчивостью к рассматриваемым условиям. Это могут быть модели, основанные на архитектуре Transformer, дообученные на аудиоданных, характерных реальным условиям функционирования железной дороги, например, шумы от движущегося подвижного состава и наличие звуков и речи громкоговорящей связи на фоне. При этом стоит отметить, что предложенная многоканальная архитектура с текущим набором используемых моделей в любом случае будет обладать повышенной устойчивостью в сравнении с одноканальным вариантом за счёт компенсации возможных ошибок на основе информации из других каналов.

4. Заключение

Данная работа посвящена совместному применению технологий компьютерного зрения и распознавания речи для видеоанализа производственных процессов в железнодорожной отрасли, что является особо актуальным в задачах нормирования труда. Основной фокус направлен на разработку многоканального подхода видеоанализа, который учитывает сложность и вариативность реальных производственных условий – перекрытие рабочих инструментов, наличие нескольких сотрудников в кадре, их выход и возвращение в кадр спустя время, а также учет неоперативного времени. В основу подхода положена интеграция нескольких ключевых компонент: детектирование объектов (инструментов и людей), трекинг, распознавание действий, а также распознавание речевых комментариев. Использование мультиканального подхода позволило компенсировать недостатки и

ограничения одноканального решения, разработанного ранее, и достичь более точного определения временных интервалов производственных операций.

Ключевыми блоками подхода стали:

- детекция людей и инструментов моделями YoloV7-Pose и YoloV8;
- высокоточный трекинг на базе комбинации xMEM и SAM;
- распознавание действий сотрудников моделью MViTv2;
- распознавание речевых комментариев оператора с помощью QuartzNet;
- модуль постобработки результатов.

Для проверки результативности предложенного решения как набора технологий, объединенных многоканальной архитектурой, было проведено тестирование на подготовленных в лабораторных условиях видеозаписях работ путевого хозяйства. Результаты экспериментов показали прирост качества на 40%.

Таким образом, результаты проведенного исследования подтверждают потенциал совместного использования технологий компьютерного зрения и обработки речи для видеоанализа производственных операций на примере путевого комплекса.

Благодаря заложенной в архитектуру модульности данный подход может быть адаптирован под аналогичные задачи в других отраслях со схожими производственными процессами. Это может быть достигнуто за счет обязательного дообучения и настройки отдельных модулей, в частности, блока детекции инструментов, распознавания характерных действий и обработки речи. Данные манипуляции могут быть трудозатратны, но необходимы для учёта специфики в альтернативной предметной области и устойчивой работы. Тогда как модуль трекинга, учитывающая главным образом визуальный образ человека, может быть переиспользован как есть или с незначительной доработкой.

Возможными направлениями дальнейшей работы может стать тестирование разработанного решения на видеозаписях работ по действующему техпроцессу хозяйства пути, дообучение моделей, при необходимости, а также апробация подхода в процессах других подразделений и служб железнодорожной отрасли.

Список литературы / References

- [1]. Никитин В.Н. Научная организация труда в парадигме устойчивого развития транспортного комплекса. *Транспортное дело России*, №5, 2024 г., стр. 60-65. / Nikitin V.N. Scientific organization of labor in the paradigm of sustainable development of the transport complex. *Transport business of Russia*, no. 5, 2024, pp. 60-65.
- [2]. Коптева Ж.Ю., Томакова И.А., Садиков Р.А. Современные информационные технологии изучения затрат рабочего времени как важный инструмент для эффективного управления бизнес-процессами. *Известия Юго-Западного государственного университета. Серия: Экономика. Социология. Менеджмент*, том 14, № 4, 2024 г., стр. 202–217. DOI: 10.21869/2223-1552-2024-14-4-202-217. / Kopteva Zh.Yu., Tomakova I.A., Sadikov R.A. Modern information technologies for studying the cost of working time as an important tool for effective management of business processes. *Proceedings of the Southwest State University. Series: Economics. Sociology. Management*, vol. 14, no. 4, 2024, pp. 202-217. (in Russian) DOI: 10.21869/2223-1552-2024-14-4-202-217.
- [3]. Никитин В.Н. Новая научная организация труда как базис системы нормирования труда в ОАО «РЖД». Повышение производительности труда на транспорте – источник развития и конкурентоспособности национальной экономики: *Труды V Всероссийской научно-практической конференции* (г. Москва, 26 мая 2021 г.). Москва: Гуманитарный институт федерального государственного бюджетного образовательного учреждения высшего образования "Российский университет транспорта (МИИТ)", 2021 г., стр. 28–37.
- [4]. Клемышев И.М., Колчин С.М., Лебедев С.С., Старков С.О. Автономная многоракурсная нейросетевая система компьютерного зрения для автоматизации контроля ручных операций. *Успехи кибернетики*, том 5, №3, 2024 г., стр. 24–33. DOI: 10.51790/2712-9942-2024-5-3-03. /

- Klemysh I.M., Kolchin S.M., Lebedev S.S., Starkov S.O. Autonomous Multi-Camera Neural Network Computer Vision System for Manual Operation Monitoring. *Russian Journal of Cybernetics*, vol. 5, no. 3, 2024, pp. 24–33 (in Russian) DOI: 10.51790/2712-9942-2024-5-3-03.
- [5]. Хацкелевич А.Н. Применение технологий искусственного интеллекта как один из факторов конкурентоспособности бизнеса в четвертой промышленной революции. *Вестник ПНИПУ. Социально-экономические науки*, №1, 2024 г., стр. 184–199. DOI: 10.15593/2224-9354/2024.1.14. / Khatskelevich A.N., Rudakov S.A., Egorov G.A. Artificial intelligence technologies as one of the factors of business competitiveness in the fourth industrial revolution. *PNRPU Sociology and Economics Bulletin*, no. 1, 2024, pp. 184-199 (in Russian) DOI: 10.15593/2224-9354/2024.1.14.
- [6]. Luo X., Li H., Yang X., Yu Y., Cao D. Capturing and understanding workers' activities in far-field surveillance videos with deep action recognition and Bayesian nonparametric learning. *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 4, 2019, pp. 333-351. DOI: 10.1111/mice.12419.
- [7]. Варанкин Д.В., Гудошник Е.Э., Мельников С.С. Автоматическая система контроля соблюдения правил охраны труда и производственной безопасности. *Бюллетень науки и практики*, том 9, №7, 2023, стр. 317-321. DOI: 10.33619/2414-2948/92/44. / Varankin D.V., Gudoshnik E.A., Melnikov S.S. Automatic Health and Safety and Production Safety Monitoring System. *Bulletin of Science and Practice*, vol. 9, no. 7, 2023, pp. 317-321 (in Russian). DOI: 10.33619/2414-2948/92/44.
- [8]. Shou Z., Wang D., Chang S.-F. Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2016)*, 2016, pp. 1049–1058. DOI: 10.1109/CVPR.2016.119.
- [9]. Liu X., Wang Q., Hu Y., Tang X., Zhang S., Bai S., Bai X. End-to-End Temporal Action Detection With Transformer. *IEEE Transactions on Image Processing*, vol. 31, 2022, pp. 5427–5441. DOI: 10.1109/TIP.2022.3195321.
- [10]. Штехин С.Е., Карачёв Д.К., Иванова Ю.А. Разработка алгоритма распознавания движений человека методами компьютерного зрения в задаче нормирования рабочего времени. *Труды ИСП РАН*, том 32, вып. 1, 2020 г., стр. 121-136. DOI: 10.15514/ISPRAS-2020-32(1)-7. / Shtekhin S.E., Karachev D.K., Ivanova Yu.A. Computer vision system for working time estimation by human activities detection in video frames. *Trudy ISP RAN/Proc. ISP RAS*, vol. 32, issue 1, 2020, pp.121-136 (in Russian) DOI: 10.15514/ISPRAS-2020-32(1)-7.
- [11]. Kadam P., Fang G., Zou J.J. Object Tracking Using Computer Vision: A Review. *Computers*, 13(6), 2024, 136. DOI: 10.3390/computers13060136.
- [12]. Wang C.-Y., Bochkovskiy A., Liao H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2023)*, 2023, pp. 7464-7475. DOI: 10.1109/CVPR52729.2023.00721.
- [13]. Cheng H.K., Schwing A.G. XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model. In *Proceedings of the European Conference on Computer Vision (ECCV'2022)*, 2022, pp. 640–658. DOI: 10.1007/978-3-031-19815-1_37.
- [14]. Kirillov A., Mintun E., Ravi N., Mao H., Rolland C., Gustafson L., Xiao T., Whitehead S., Berg A.C., Lo W.-Y., Dollár P., Girshick R. Segment Anything. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'2023)*, 2023, pp. 4015-4026. DOI: 10.1109/ICCV51070.2023.00371.
- [15]. Zhang Y., Sun P., Jiang Y., Yu D., Weng F., Yuan Z., Luo P., Liu W., Wang X. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. In *Proceedings of the European Conference on Computer Vision (ECCV'2022)*, 2022, pp. 1-21. DOI: 10.1007/978-3-031-20047-2_1
- [16]. Luiten J., Ošep A., Dendorfer P., Torr P., Geiger A., Leal-Taixé L., Leibe B. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *International Journal of Computer Vision*, vol. 129, no. 2, 2021, pp. 548-578. DOI: 10.1007/s11263-020-01375-2.
- [17]. Bernardin, K., Stiefelagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing* 2008, 1–10 (2008). DOI: 10.1155/2008/246309.
- [18]. Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: *European Conference on Computer Vision*, pp. 17–35. Springer (2016). DOI: 10.48550/arXiv.1609.01775.
- [19]. Бенчмарк MOT20 для алгоритмов трекинга [Электронный ресурс]. Доступно по адресу: <https://motchallenge.net/results/MOT20/> (Дата обращения: 22.12.2025).
- [20]. Список детектируемых инструментов [Электронный ресурс]. Доступно по адресу: <https://disk.yandex.ru/i/ZrtRhoUijMdJ6Q> (Дата обращения: 22.12.2025).

- [21]. Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2022), 2022, pp. 10674-10685. DOI: 10.1109/CVPR52688.2022.01042.
- [22]. Brooks T., Holynski A., Efros A.A. InstructPix2Pix: Learning to Follow Image Editing Instructions. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2023), 2023, pp. 18392-18402. DOI: 10.1109/CVPR52729.2023.01764.
- [23]. Карачёв Д.К., Штехин С.Е., Тарасян В.С., Смолин И.Ю., Исаков М.В. Использование переноса стиля как способ улучшения обобщающей способности нейросети в задаче детекции объектов. Труды ИСП РАН, том 35. вып. 6, 2023 г., стр. 247-264. DOI: 10.15514/ISPRAS-2023-35(6)-16. / Karachev D.K., Shtekhin S.E., Tarasyan V.S., Smolin I.U., Isakov M.V. Style transfer as a way to improve the generalization ability of a neural network in an object detection task. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 6, 2023. pp. 247-264. (in Russian) DOI: 10.15514/ISPRAS-2023-35(6)-16.
- [24]. Prokhorenkova L., Gusev G., Vorobev A., Dorogush A.V., Gulin A. CatBoost: unbiased boosting with categorical features. In Proceedings of the 32nd International Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems (NeurIPS'2018), 2018, pp. 6639-6649.
- [25]. Pham H.H., Khoudour L., Crouzil A., Zegers P., Velastin S.A. Video-based Human Action Recognition using Deep Learning: A Review. arXiv.org. 2022. Available at <https://arxiv.org/abs/2208.03775>. (accessed 30.06.2025).
- [26]. Carreira J., Zisserman A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2017), 2017, pp. 4724-4733. DOI: 10.1109/CVPR.2017.502.
- [27]. Duan H., Zhao Y., Chen K., Lin D., Dai B. Revisiting Skeleton-based Action Recognition. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2022), 2022, pp. 2959-2968. DOI: 10.1109/CVPR52688.2022.00298.
- [28]. Li Y., Wu C.-Y., Fan H., Mangalam K., Xiong B., Malik J., Feichtenhofer C. MViT2: Improved Multiscale Vision Transformers for Classification and Detection. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'2022), 2022, pp. 4794-4804. DOI: 10.1109/CVPR52688.2022.00476.
- [29]. Kriman S., Beliaev S., Ginsburg B., Huang J., Kuchaiev O., Lavrukhin V., Leary R., Li J., Zhang Y. QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'2020), 2020, pp. 6124-6128. DOI: 10.1109/ICASSP40776.2020.9053889.
- [30]. Karpov N., Denisenko A., Minkin F. Golos: Russian Dataset for Speech Research. In Proceedings of Interspeech (Interspeech'2021), 2021, pp. 1419-1423. DOI: 10.21437/Interspeech.2021-46.

Информация об авторах / Information about authors

Илья Юрьевич СМОЛИН – ведущий специалист по анализу данных. Области научных интересов: искусственный интеллект, компьютерное зрение, детектирование и сегментация объектов, генеративные нейронные сети.

Ilya Urevich SMOLIN – Senior Data Scientist. Research areas: artificial intelligence, computer vision, object detection and segmentation, generative neural networks.

Александр Александрович ЛЮБЧЕНКО – кандидат технических наук, заведующий лабораторией искусственного интеллекта и нейронных сетей ООО «ОЦРВ», филиал Сириус. Доцент кафедры «Средства связи и информационная безопасность» Омского государственного технического университета. Области научных интересов: методы машинного обучения, оптимизации и имитационного моделирования.

Alexander Alexandrovich LYUBCHENKO – Cand. Sci. (Tech.), Head of the Laboratory of Artificial Intelligence and Neural Networks LLC «OCRВ», Sirius Branch. Associate Professor of the Department of Communications and Information Security at Omsk State Technical University. Research areas: machine learning techniques, methods of optimization and simulation.

Максим Владимирович ИСАКОВ – старший специалист по анализу данных. Области научных интересов: искусственный интеллект, компьютерное зрение, распознавание действий на видео.

Maksim Vladimirovich ISAKOV – Middle Data Scientist. Research areas: artificial intelligence, computer vision, action recognition in video.

Дарья Александровна ВОЖДАЕВА – старший специалист по анализу данных. Области научных интересов: искусственный интеллект, распознавание и обработка речи, цифровая обработка сигналов.

Darya Aleksandrovna VOZHDAEVA – Middle Data Scientist. Research areas: artificial intelligence, automatic speech recognition, speech processing, digital signal processing.



DOI: 10.15514/ISPRAS-2026-38(1)-15

Сбор данных для обучения и тестирования моделей сверхразрешения, работающих в реальном времени

¹ Е.Н. Богатырев, ORCID: 0000-0002-6173-3561 <evgeney.bogatyrev@graphics.cs.msu.ru>

^{1,2} И.А. Молодецких, ORCID: 0000-0002-8294-0770 <ivan.molodetskikh@graphics.cs.msu.ru>

^{1,2,3} Х.Н. Абуд, ORCID: 0009-0009-7131-5839 <khaled.abud@graphics.cs.msu.ru>

^{1,2,3,4} Д.С. Ватолин, ORCID: 0000-0002-8893-9340 <dm.vatolin@ispras.ru>

¹ Московский государственный университет имени М.В. Ломоносова,
Россия, 119991, Москва, Ленинские горы, д. 1.

² Центр искусственного интеллекта МГУ,
Россия, 119991, г. Москва, Ленинские горы, ГСП-1

³ Институт перспективных исследований проблем
искусственного интеллекта и интеллектуальных систем МГУ имени М.В. Ломоносова,
Россия, 119192, г. Москва, Ломоносовский проспект, 27, корп. 1.

⁴ Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Современные модели сверхразрешения (SR) реального времени плохо справляются с особенностями сжатого видео. Часто используемые наборы данных не отражают в полной мере характеристики стримингового контента, что снижает релевантность существующих бенчмарков. Авторами предложен новый набор данных StreamSR, собранный из видеороликов с платформы YouTube и охватывающий широкий спектр жанров и разрешений видео, характерных для реальных сценариев. Протестировано 11 современных моделей сверхразрешения, работающих в реальном времени, оценена их эффективность для задач стримингового вещания. В работе демонстрируется, что дообучение других моделей на представленном наборе данных приводит к значительному повышению качества, которое хорошо обобщается на стандартные бенчмарки. Предлагается EfRFLN – эффективная модель сверхразрешения, в которую интегрированы Efficient Channel Attention и гиперболический тангенс в качестве функции активации, что является новым решением в контексте сверхразрешения в реальном времени. Архитектура оптимизирована для максимизации эффективности, также разработана составная функция потерь, улучшающая сходимость обучения. EfRFLN объединяет преимущества существующих архитектур, одновременно улучшая как визуальное качество, так и производительность.

Ключевые слова: сверхразрешение; видео; набор данных; кодирование видео; оценка качества видео.

Для цитирования: Богатырев Е.Н., Молодецких И.А., Абуд Х.Н., Ватолин Д.С. Сбор данных для обучения и тестирования моделей сверхразрешения, работающих в реальном времени. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 221–240. DOI: 10.15514/ISPRAS–2026–38(1)–15.

Благодарности: Работа выполнена при финансовой поддержке Министерства экономического развития Российской Федерации (соглашение № 139-15-2025-012 от 20.06.2025). Работа выполнялась с использованием суперкомпьютера «МГУ-270» МГУ имени М.В. Ломоносова.

Data Collection for Training and Testing Real-Time Super-Resolution Models

¹ E.N. Bogatyrev, ORCID: 0000-0002-6173-3561 <evgeney.bogatyrev@graphics.cs.msu.ru>

^{1,2} I.A. Molodetskikh, ORCID: 0000-0002-8294-0770 <ivan.molodetskikh@graphics.cs.msu.ru>

^{1,2,3} K.N. Abud, ORCID: 0009-0009-7131-5839 <khaled.abud@graphics.cs.msu.ru>

^{1,2,3,4} D.S. Vatolin, ORCID: 0000-0002-8893-9340 <dm.vatolin@ispras.ru>

¹ Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

² MSU AI Center,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

³ MSU Artificial Intelligence Institute
build. 1, 27, Lomonosovskiy, ave., Moscow, 119992, Russia.

⁴ Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Existing methods used within modern real-time super-resolution (SR) often struggle with the specific artifacts present in compressed video. Commonly used datasets do not fully capture the characteristics of streaming content, which reduces the relevance of existing benchmarks. To address this issue, this paper introduces a novel dataset collected from videos on the YouTube platform. This dataset covers a wide range of video genres and resolutions typical of real-world streaming scenarios. We tested 11 state-of-the-art real-time super-resolution models and evaluated their performance on streaming-related tasks. Our study demonstrates that fine-tuning existing models on the proposed dataset leads to a significant improvement in quality, which generalizes well across various standard benchmarks. We also propose EfRLFN – an efficient super-resolution network that integrates Efficient Channel Attention and the hyperbolic tangent (Tanh) as an activation function, which is a novel approach in the context of real-time super-resolution. The architecture was meticulously optimized for maximum efficiency, and a composite loss function was developed to improve training convergence. EfRLFN combines the advantages of existing architectures while enhancing both visual quality and performance.

Keywords: super-resolution; video; dataset; video compression; video quality assessment.

For citation: Bogatyrev E.N., Molodetskikh I.A., Abud K.N., Vatolin D.S. Data Collection for Training and Testing Real-Time Super-Resolution Models. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 221-240 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-15.

Acknowledgements. This work was supported by the The Ministry of Economic Development of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000C313925P4H0002; grant No 139-15-2025-012). The research was carried out using the MSU-270 supercomputer of Lomonosov Moscow State University.

1. Введение

В последние годы наблюдается необходимость в высококачественном стриминге видео, обусловленная широким распространением таких платформ, как YouTube, Twitch и Netflix [1]. Из-за ограничений пропускной способности сети потоковые сервисы часто сжимают видеоконтент, что может приводить к появлению визуальных искажений, таких как блочность, размытие и потеря мелких деталей. Несмотря на то, что существуют технологии, позволяющие повышать разрешение видео в реальном времени, такие методы часто не справляются с уникальными искажениями, создающимися после сильного сжатия видео [2]. Проведённый анализ показывает, что существует ограниченное число методологий SR (Super-Resolution, сверхразрешение), способных эффективно увеличивать разрешение

сжатых видео [3]. Кроме того, отмечается заметный дефицит моделей сверхразрешения, способных работать в реальном времени. Соревнования NTIRE [4, 5] сыграли ключевую роль в продвижении разработки моделей SR реального времени, способствуя значительному прогрессу в этой области. Однако следует отметить, что многие из этих моделей SR обучались только на наборах данных, таких как DIV2K [6] или Vimeo90K [7], которые не покрывают сценарии работы со сжатым видео.

Недавно был представлен и популяризирован метод Video Super-Resolution от NVIDIA (NVIDIA VSR) [8] – это основанный на машинном обучении метод SR, интегрированный в драйвер GPU. Однако, несмотря на свою эффективность, NVIDIA VSR часто не восстанавливает мелкие текстуры и вносит чрезмерное сглаживание, что ограничивает его практическую пользу, как проиллюстрировано на рис. 1. Аналогично, другие современные методы SR реального времени, такие как SPAN [9] и RLFN [10], показывают высокие результаты на стандартных наборах данных, но не оптимизированы для сложностей, появляющихся при сжатии видео [2].



Рис. 1. Примеры искажений, создаваемых NVIDIA VSR, в сравнении с бикубической интерполяцией (bicubic). Первая и вторая пара изображений показывают разницу в цвете. Третья пара показывает, как NVIDIA VSR сильно сглаживает текстуры.

Fig. 1. Examples of artifacts introduced by NVIDIA VSR in comparison to bicubic interpolation. The first and second image pairs demonstrate the color difference. The third pair shows how NVIDIA VSR aggressively oversmooths textures.

Для решения указанных проблем в данной работе представлен набор данных **StreamSR**, содержащий 5,200 видеороликов с видеохостинга YouTube (в сумме более 10 млн отдельных кадров), охватывающих различные жанры и соотношения сторон. Этот набор данных является надежным бенчмарком для оценки методов SR в условиях потоковой передачи. Кроме того, разработана модель **Efficient Residual Lightweight Feature Network (EfRLFN)**. Данная модель адаптирована для применений в реальном времени и демонстрирует значительное улучшение как объективных, так и субъективных метрик качества видео.

Основные вклады работы заключаются в следующем.

StreamSR – набор данных для сверхразрешения потокового видео. Собран и проанализирован многомасштабный набор данных, полученный из YouTube. Набор включает 5,200 пользовательских видеороликов разнообразной тематики и типов контента. Также проведен сравнительный анализ с существующими наборами данных.

Системное тестирование моделей сверхразрешения реального времени. Оценены 11 методов сверхразрешения реального времени с использованием представленного набора данных. При этом их производительность анализировалась как с дообучением на тренировочном наборе, так и без него. Были использованы 7 различных объективных метрик качества видео и проведено масштабное исследование пользовательских предпочтений с участием более 3,800 человек.

Метод сверхразрешения реального времени. Исходя из результатов тестирования моделей SR, были предложены модификации существующих архитектур. Архитектура SPAN была улучшена изменением функции активации, что показало прирост качества по многим объективным и субъективным метрикам. На основе популярной архитектуры RLFN проведена тщательная оптимизация, включающая новую конструкцию блоков, механизм Efficient Channel Attention и усовершенствованную процедуру обучения. Проведённые

оценки показывают, что разработанная архитектура EfRFLN показывает лучшие результаты в задаче сверхразрешения реального времени, превосходя другие методы по соотношению качество-сложность, что подтверждается расширенными экспериментами.

2. Обзор литературы

2.1 Методы сверхразрешения

Современные модели SR реального времени достигают различного соотношения между качеством реконструкции и вычислительной эффективностью за счёт инновационных архитектурных решений.

Среди моделей, способных работать в реальном времени, метод **Bicubic++** [11] сочетает быстрое обратимое извлечение признаков с глобальным структурированным прореживанием сверточных слоев, достигая скорости обработки, сравнимой с традиционной бикубической интерполяцией. Метод **AsConvSR** [12] вводит динамические свертки, которые адаптируют свои ядра на основе характеристик входных данных. Данная модель оптимизирована для мобильных устройств. Для увеличения разрешения до высоких значений метод **RT4KSR** [13] использует перестановку пикселей и структурную репараметризацию блоков NAFNet для эффективного повышения разрешения с 720p/1080p до 4K. Архитектура метода **SPAN** [9] характеризуется механизмом внимания с симметричными активациями для усиления важных признаков и подавления избыточных. Это уменьшает вычислительные затраты на 40% по сравнению с традиционными модулями внимания. Аналогично, метод **RLFN** [10] реализует упрощенную трехслойную сверточную структуру, достигая повышения разрешения с 1080p до 4K менее, чем за 15 мс, при сохранении сильной агрегации признаков. Метод **TMP** [14] улучшает повышение разрешения видео за счёт минимизации вычислительной избыточности в оценке движения.

В дополнение к легковесным методам реального времени, трансформерные модели, такие как **SwinIR** [15] и **HiT-SR** [16], достигают превосходного качества за счёт механизмов самовнимания и иерархической многомасштабной обработки. Существуют и другие подходы к задаче SR, такие как **COMISR** [3] и **CAVSR** [17], которые специализируются на увеличении разрешения сжатого видео. Метод **Real-ESRGAN** [18] является популярным генеративным методом, который использует моделирование деградации высокого порядка для симуляции реальных искажений. Хотя эти методы демонстрируют превосходные результаты по сравнению с моделями SR реального времени, они являются вычислительно затратными, что делает их непригодными для приложений реального времени.

Несмотря на то, что все эти модели продемонстрировали различную степень успеха в улучшении качества изображений, проблемы со сверхразрешением сжатого видеоконтента остаются актуальными. Данная работа строится на этих фундаментальных исследованиях с целью создания наиболее подходящей модели SR для использования в реальном времени.

2.2 Наборы данных и соревнования в задаче SR реального времени

Наборы данных играют ключевую роль в обучении и оценке моделей SR. **DIV2K** [6] и **Vimeo90K** [7] широко используются для тестирования, так как содержат эталонные изображения высокого разрешения. Однако этим наборам данных не хватает примеров с артефактами сжатия, характерными для потокового контента. Аналогично, набор данных **YouTube UGC** [19] содержит разнообразные видео, но не фокусируется на сценариях, релевантных для SR в реальном времени. Для оценки качества SR также широко используются наборы данных Set14 [20], BSD100 [21] и Urban100 [22], содержащие естественные и структурированные изображения с фиксированным масштабом. Для задач видео-SR набор данных REDS [23] предоставляет динамические сцены с размытием

движения и артефактами сжатия, что делает его подходящим для обучения и тестирования сложных архитектур SR.

Соревнование NTIRE 2023 Real-Time SR Challenge [4] было сфокусировано на разработке методов сверхразрешения, способных работать в реальном времени при сохранении высокого качества изображения. Перед участниками стояла задача разработки архитектуры, балансирующей вычислительную эффективность и производительность. Среди лидирующих решений присутствовали модели, использующие продвинутые методы прореживания и стратегии квантования для уменьшения размера модели без потери качества выходных данных. Также применялись методы дистилляции знаний для переноса представлений из больших моделей в меньшие и более эффективные сети, что обеспечивало обработку в реальном времени для таких приложений, как потоковое вещание и видеоконференции.

Продолжая фокус предыдущего года, **Соревнование NTIRE 2024 Efficient SR Challenge** [5] акцентировало внимание на эффективных моделях SR, способных обеспечивать высококачественные результаты при минимизации потребления ресурсов. Участники исследовали различные стратегии, включая использование трансформерных архитектур, которые адаптивно распределяют вычислительные ресурсы на основе характеристик входных данных. Кроме того, наблюдался повышенный интерес к методам, использующим обучение под несколько задач, где модели обучались не только для сверхразрешения, но и для таких задач, как шумоподавление и удаление блочности, повышая общую эффективность обработки сжатого видеоконтента. Несколько подходов интегрировали механизмы внимания для фокусировки на критически важных областях изображения, улучшая как скорость, так и качество процесса увеличения разрешения.

На основе описанных выше соревнований было разработано много качественных методов SR реального времени. Однако тренировочные наборы данных в этих соревнованиях не содержат сжатых видео, из-за чего обученные на них модели SR не приспособлены к работе с потоковым контентом.

3. Набор данных и оценка SR реального времени

В данном разделе описывается методология, использованная для сбора видео в наборе данных StreamSR и их последующей категоризации.

3.1 Сбор данных

Получение видео. Для генерации поисковых запросов для сбора видео использовалась большая языковая модель (GPT-4o) [24]. Исходя из 20 различных категорий YouTube (например, путешествия, образование, игры), GPT-4o генерировала 100 разнообразных поисковых запросов для каждой категории. Затем загружались топ-100 видео для каждого запроса. Список некоторых тем и примеры сгенерированных запросов приведены в табл. 1.

Фильтрация. Из всех видео рассматривались только видео с лицензией Creative Commons (CC BY 4.0) с доступными разрешениями 360p, 720p и 1440p. Для обеспечения точного выравнивания кадров требовалось постоянное значение частоты кадров, которое проверялось автоматически. Разрешения были выбраны для поддержки двух треков бенчмарка SR: $2\times$ (720p→1440p) и $4\times$ (360p→1440p). Для каждого разрешения выбирался поток с наивысшим битрейтом для обеспечения оптимального качества эталонного высококачественного потока. Из каждого видео извлекались первые 30 секунд ($\leq 2,000$ кадров). Видео с длинными статичными заставками отфильтровывались путем сравнения 1-ого, 100-ого и 150-ого кадра видео. Видео отбраковывались, если среднее значение метрики PSNR (peak signal-to-noise ratio, пиковое отношение сигнала к шуму) между этими кадрами превышал 40 дБ. В результате был получен набор из 5,200 видео, который был назван набором данных **StreamSR**.

Категоризация. Для построения разнообразного и репрезентативного тестового набора видео были кластеризованы с использованием признаков SI, TI [19], битрейта, MDTVSA [25] и текстовых эмбеддингов поисковых запросов, посчитанных с помощью SigLIP [26]. Эмбеддинг SigLIP проецировался в 3-мерное пространство с помощью метода главных компонент для снижения размерности. Все признаки были нормализованы до нулевого среднего и единичной дисперсии. Кластеризация выполнялась алгоритмом К-средних (K-Means) с использованием евклидовой метрики расстояния. Количество кластеров $K=20$ было выбрано эмпирически на основе анализа метода локтя (elbow method), который показал, что это значение обеспечивает хороший компромисс между внутрикластерной схожестью и разнообразием набора данных. Из каждого из 20 полученных кластеров для формирования фиксированного тестового набора выбиралось одно видео, наиболее близкое к центроиду. Остальные видео случайным образом разделялись на обучающий и валидационный наборы в соотношении 10:1, сохраняя пропорции кластеров для предотвращения смещения распределения.

Табл. 1. Запросы для скачивания видео, сгенерированные с помощью LLM. В каждой ячейке представлено название категории и соответствующие ей запросы.

Table 1. Download query prompts generated using an LLM. Each cell contains a category name and its corresponding search queries.

Спорт	Животные	Активность
"highlights of football"	"wildlife documentary"	"people dancing"
"sports slow motion"	"cute animal videos"	"cooking tutorials"
"extreme sports action"	"slow motion animals"	"fitness workouts"
"basketball dunks"	"pets playing"	"street performers"
"soccer goals"	"nature interactions"	"DIY projects"
Искусство	Съемка вблизи	История
"time-lapse painting"	"macro insects"	"historical footage"
"sculpting techniques"	"close-up flowers"	"old documentaries"
"DIY crafts"	"macro techniques"	"historical events"
"artistic creations"	"tiny world"	"classic film clips"
"watercolor tutorial"	"everyday objects"	"moon landing"
Анимация	Наука	Природа
"animation films"	"science experiments"	"nature landscapes"
"animated scenes"	"physics demos"	"time-lapse nature"
"cartoon clips"	"chemical reactions"	"ocean waves"
"stop motion"	"biology explained"	"forest scenery"
"3D animation"	"science phenomena"	"mountain views"

В табл. 2 проводится сравнение набора данных StreamSR с существующими наборами пользовательского контента. В отличие от наборов данных для классификации (YouTube-8M, Kinetics) или сегментации (YouTube-VOS), набор StreamSR разработан для задачи SR, предоставляя выровненные видео с разрешением от 360p до 1440p и с треками увеличения 2× и 4×. Благодаря более длинным клипам (25-30с), видео с различными разрешениями и естественными артефактами сжатия, а также более широкому диапазону качества, он лучше представляет реальные сценарии потоковой передачи по сравнению с предыдущими наборами данных SR (REDS, Vimeo-90K).

Рис. 2 показывает распределение признаков видео в собранном наборе данных и в других. Из графиков видно, что видео в собранном наборе данных покрывают широкий спектр значений по каждому из признаков, однако содержатся в отдельном кластере стриминговых видео.

Табл. 2. Сравнение наборов данных пользовательского видео. Диапазон качества показывает 5-й и 95-й проценты метрики MDTVSA [25]. Длительность показывает типичную длительность видеоклипа.

Table 2. A comparison of user-generated video datasets. The quality range shows the 5th and 95th percentiles of the MDTVSA [25] metric. Duration indicates the typical clip length.

Набор данных	Год	Разрешения	Диапазон качества	Применение	Длительность
YouTube-8M	2016	240p–1080p	0.45–0.58	Классификация	5 мин
Kinetics-400	2017	1080p	0.48–0.59	Распознавание действий	12 сек
Kinetics-700	2019	1080p	0.47–0.58	Распознавание действий	12 сек
YouTube-VOS	2018	360p–1080p	0.46–0.57	Сегментация объектов	25 сек
YouTube-BB	2017	1080p	0.50–0.60	Отслеживание объектов	12 сек
YT-Temporal 180M	2020	144p–1080p	0.42–0.56	Языковое предобучение	7 сек
DIV2K	2017	2K	0.55–0.65	Сверхразрешение изображений	-
REDS	2019	360p–720p	0.50–0.60	Сверхразрешение видео	5 сек
Vimeo-90K	2019	448p	0.48–0.58	Сверхразрешение видео	1 сек
UDM10	2019	540p–4K	0.45–0.58	Сверхразрешение изображений	-
RealSR	2019	4K	0.43–0.59	Сверхразрешение изображений	-
VideoLQ	2021	270p–1080p	0.43–0.54	Сверхразрешение видео	10 сек
StreamSR	2025	360p–1440p	0.41–0.61	Сверхразрешение	30 сек

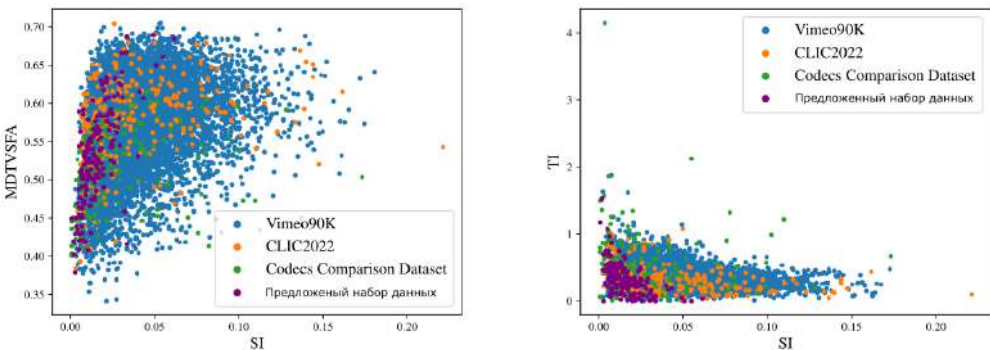


Рис. 2. Распределение пространственной и временной сложности [19] и MDTVSA [25] для видео из разных наборов данных.

Fig. 2. The distribution of spatial and temporal complexity [19] and MDTVSA [25] scores for videos from different datasets.

3.2 Дообучение моделей SR

Для дообучения и тестирования был выбран широкий список моделей SR на основе результатов соревнований NTIRE за последние годы, включающий как классические подходы, такие как бикубическая интерполяция, Bicubic++ [11] и ESPCN [27], так и современные методы, такие как SMFANet [28], RLFN [10] и SPAN [9]. В сравнение также была включена проприетарная модель NVIDIA VSR. В табл. 3 собрана информация о тестируемых моделях.

Для обеспечения справедливого сравнения каждой модели все тестируемые модели предварительно обучались на тренировочной части StreamSR, а результаты оценивались как для дообученных, так и для стандартных моделей. Если для модели не существовало предварительно обученных весов для конкретного коэффициента увеличения (2× или 4×), она обучалась с нуля, и использовалась только обученная версия. Каждая модель была обучена в течение 500 эпох, используя процесс обучения, описанный в оригинальной статье. Все модели оценивались на видео разрешения 720p с увеличением разрешения в 2× до 1440p. Для измерений FPS каждая модель запускалась на видеокарте NVIDIA GeForce RTX 2080, и время выполнения рассчитывалось на одном и том же тестовом фрагменте из 100 кадров, усредненное по 3 последовательным запускам.

Табл. 3. Характеристики использованных моделей сверхразрешения реального времени.

Table 3. Characteristics of the Real-Time Super-Resolution Models Used.

Метод	Параметры (М)	FLOPs (Гб)	Увеличение	Тип архитектуры
ESPCN [27]	0.04	2.43	2×,3×,4×	Sub-pixel CNN
XLSR [29]	0.03	1.81	4×	CNN
RLFN [10]	0.40	136	2×,4×	CNN+Attention
AsConvSR [12]	2.35	9	2×	Assembled Convolution
Bicubic++ [11]	0.05	0.83	3×	CNN
NVIDIA VSR [8]	-	-	2×,3×,4×	CNN+Attention
RT4KSR [13]	0.05	172	2×	CNN+Attention
SAFMN [30]	0.23	52	4×	CNN с модуляцией признаков
SMFANet [28]	0.20	11	4×	CNN с модуляцией признаков
SPAN [9]	0.43	28	2×,4×	Беспараметровое внимание

3.3 Оценивание качества методов

Для оценивания различных аспектов производительности алгоритмов SR был выбран разнообразный набор объективных метрик. PSNR и SSIM [31] служат фундаментальными мерами точности на уровне пикселей и структурного сходства. LPIPS [32], с другой стороны, выходит за рамки традиционных метрик и использует глубокие признаки для лучшего соответствия человеческому восприятию. Помимо метрик с эталоном (Full-Reference, FR), в сравнение также были включены метрики, не требующие эталона (No-Reference, NR). Они оценивают качество изображений после сверхразрешения без сравнения с реальными примерами высокого разрешения. В частности, были использованы две NR метрики: MUSIQ [33] – мощная трансформерная модель, превосходно справляющаяся с оценкой реального контента, и CLIP-IQA [34] – популярная модель на основе CLIP.

3.4 Субъективное сравнение

Для более точной оценки лучших моделей SR было проведено субъективное сравнение. Поскольку пользовательские исследования часто могут быть дорогостоящими, на предварительном этапе были исключены модели, показавшие низкое качество на основе объективных метрик. Для оценки и ранжирования качества различных моделей SR с субъективной точки зрения было проведено попарное краудсорсинговое сравнение с использованием сервиса Subjectify.us [36]. Сравнение включало 11 моделей SR (+ эталонное видео) и в общей сложности 660 пар видео.

Оценка была организована в виде двух отдельных треков: один сфокусирован на увеличении 2x, другой – на увеличении 4x. При попарном сравнении видео точность разметки выше, так как участники видят оба сравниваемых видео. Однако, учитывая ограниченное количество участников с разрешением экрана выше Full HD, было принято решение представлять каждое тестовое видео в виде центрального кадрирования в разрешении Full HD, причем видео показывались последовательно.

В ходе эксперимента участникам показывались пары видео, сгенерированные двумя случайно выбранными моделями SR. Им предлагалось выбрать видео с меньшим количеством визуальных искажений, а также была предоставлена возможность указать, что видео «неразличимы». Каждая пара видео оценивалась в общей сложности 30 участниками, при этом каждый участник оценивал 10 пар видео. Количество оценок на пару было определено на основе расчета статистической мощности, что обеспечивает достаточную точность для ранжирования моделей с использованием модели Брэдли-Терри [37] при ожидаемом уровне согласия между участниками.

Для обеспечения достоверности результатов среди 10 пар были включены два контрольных вопроса с предопределенными правильными ответами. Ответы участников, которые не ответили правильно на один или более контрольных вопросов, исключались из дальнейшего анализа. В субъективной оценке приняли участие 3 822 уникальных участника. Итоговые субъективные баллы были получены на основе 37 184 ответов с применением модели Брэдли-Терри [37]. Для количественной оценки надежности полученных субъективных данных был рассчитан коэффициент межэкспертной согласованности Флейса [38], который составил 0.65, что указывает на существенное согласие между участниками и подтверждает надежность собранных оценок.

На рис. 3 визуализирован интерфейс сервиса Subjectify.us, использованного для субъективного сравнения.

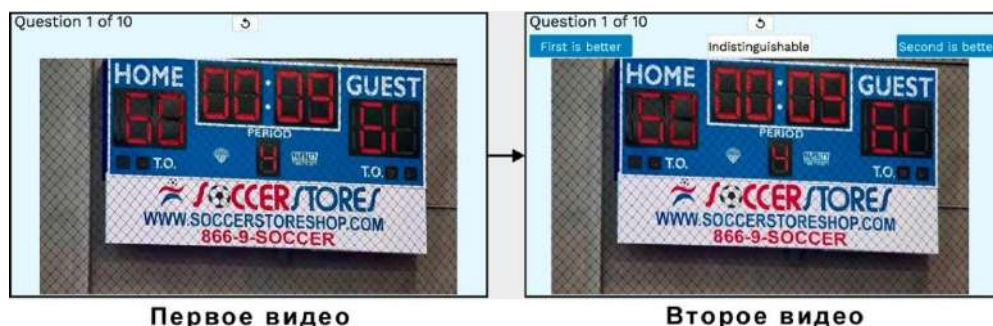


Рис. 3. Интерфейс платформы Subjectify.us.
Fig. 3. Subjectify.us interface.

Перед началом сравнения каждому участнику показывалась следующая инструкция:

«Вам будут показаны пары видео, демонстрируемые одно за другим. Для каждой пары выберите видео с лучшим визуальным качеством (более четкое, с меньшим

количеством искажений, таких как размытие или пикселизация). Если видео выглядят одинаково, выберите «Неразличимы». Будьте внимательны – в тесте присутствуют контрольные вопросы. Неправильные ответы на контрольные вопросы приведут к отклонению ваших результатов».

На рис. 4 представлен анализ корреляции между объективными метриками качества видео и субъективными оценками людей. Результаты экспериментов демонстрируют, что четыре метрики – CLIP-IQA, ERQA, LPIPS и MUSIQ – показывают значимые корреляции с субъективными оценками качества.

Данные результаты согласуются с предыдущими исследованиями в трех ключевых аспектах. Во-первых, метрики, использующие глубокие признаки (LPIPS, MUSIQ), показывают лучшую согласованность с человеческим восприятием по сравнению с традиционными подходами. Во-вторых, ERQA показывает хорошие результаты в задаче оценки качества восстановления видео. В-третьих, слабая корреляция PSNR с человеческим восприятием, что согласуется с другими работами [2,35].

Табл. 3. Сравнение качества SPAN (Sigmoid – 0.5) и SPAN (tanh).
Table 3. The comparison between SPAN (Sigmoid – 0.5) and SPAN (tanh).

	Set14	BSD100	Urban100	DIV2K
Модель	SSIM↑/LPIPS↓	SSIM↑/LPIPS↓	SSIM↑/LPIPS↓	SSIM↑/LPIPS↓
SPAN (Sigmoid – 0.5)	0.836 / 0.142	0.837 / 0.239	0.847 / 0.118	0.890 / 0.175
SPAN (tanh)	0.837 / 0.140	0.841 / 0.228	0.853 / 0.115	0.891 / 0.167

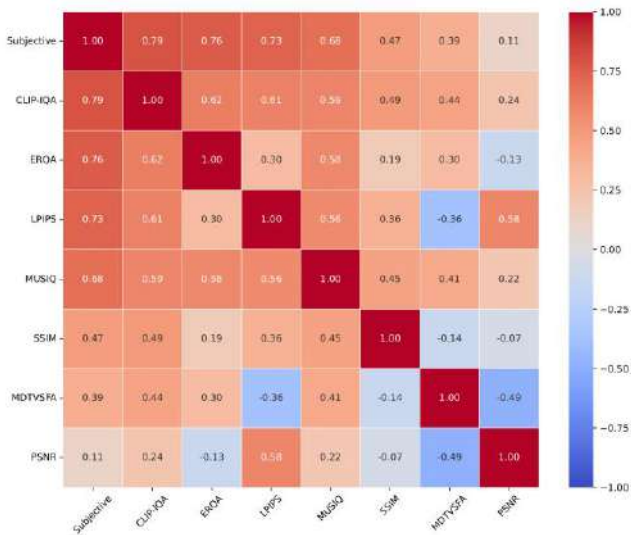


Рис. 4. Корреляция Пирсона между объективными метриками качества и субъективными оценками.
Fig. 4. Pearson correlation between objective metrics and pairwise subjective scores.

4. Разработка моделей SR реального времени

4.1 Улучшение архитектуры SPAN

Основное изменение заключается в использовании гиперболического тангенса (tanh) в качестве функции активации. Данная модификация мотивирована исследованиями [9],

демонстрирующими, что нечётные симметричные функции активации, такие как $Sigmoid(x) - 0.5$ или гиперболический тангенс, сохраняют как величину, так и знак признаков. В отличие от функции активации ReLU, которая отбрасывает отрицательные активации, $\tanh$ обеспечивает более богатый поток градиентов и предотвращает потерю информации в картах внимания, что приводит к уточнению признаков. Эмпирические исследования демонстрируют, что активация на основе $\tanh$ может улучшить распространение градиента в глубоких сетях [39] и повысить производительность в задачах плотного предсказания [40, 41]. Проведенные эксперименты демонстрируют, что замена смещенной сигмоиды ($Sigmoid(x) - 0.5$) на активацию гиперболическим тангенсом в архитектуре SPAN приводит к улучшениям по сравнению с оригинальной архитектурой. В табл. 3 проводится сравнение обоих вариантов на стандартных бенчмарках.

4.2 Архитектура сети EfRFLN

Предлагаемая модель **Efficient Residual Local Feature Network (EfRFLN)** улучшает архитектуру RFLN [10] за счёт целевых модификаций, направленных на повышение эффективности и качества реконструкции. На рис. 5 представлена общая архитектура модели. Архитектура состоит из блока извлечения признаков, основного модуля обработки признаков и блока реконструкции. В силу простоты и параметрической эффективности, блоки извлечения и реконструкции признаков были перенесены из оригинальной модели. Ключевые инновации предложенной модели сосредоточены в конструкции блока ERLFB и в процедуре обучения.

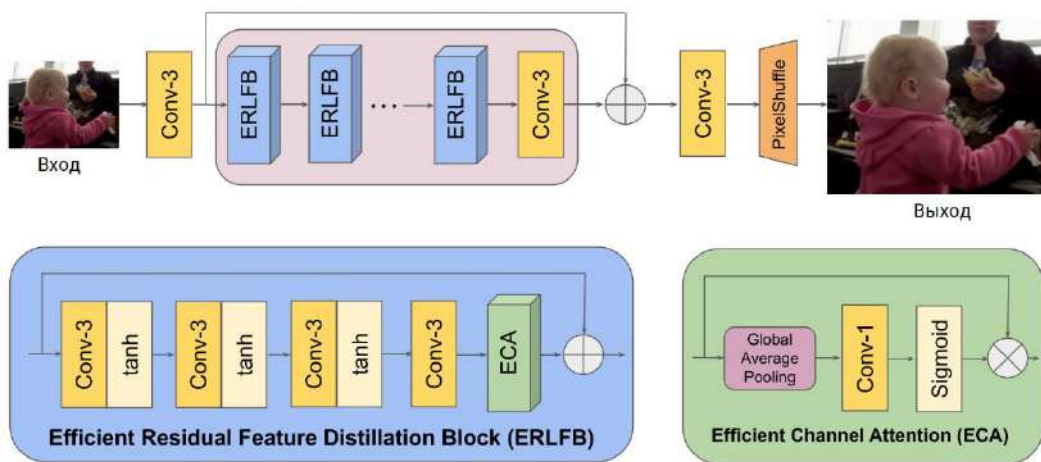


Рис. 5. Архитектура предложенной модели EfRFLN. Основная часть состоит из 6 блоков ERLFB.
Fig. 5. The architecture of the proposed EfRFLN. The main part consists of 6 ERLFB.

Блоки эффективных остаточных локальных признаков (ERLFB) заменяют оригинальные блоки RLFB в RFLN. По аналогии с архитектурой SPAN основное изменение заключается в использовании **гиперболического тангенса ($\tanh$)** вместо ReLU в модулях уточнения.

Еще одной важной модификацией является механизм внимания, используемый для глобальной агрегации информации. В разработанной модели был использован механизм **Efficient Channel Attention (ECA)** вместо Enhanced Spatial Attention (ESA), используемый в RFLN. В то время как ESA использует многослойные сверточные группы для пространственного внимания, ECA использует облегченный механизм канального внимания на основе глобального усреднения (Global Average Pooling) и свертки 1×1 . Данное

изменение значительно сокращает вычислительные затраты при сохранении конкурентоспособной производительности, что подтверждается проведенными исследованиями.

Кроме того, архитектура ERLFB оптимизирована за счёт удаления избыточных соединений и упрощения этапа сглаживания признаков до одной свертки 3×3 . Данная оптимизация уменьшает вычислительную фрагментацию, улучшая скорость вывода без ухудшения качества реконструкции.

4.3. Функция потерь

Контрастная функция потерь, используемая в методе RLFN, эффективно выравнивает промежуточные признаки с помощью пар позитивных и негативных образцов. Однако она создает две практические проблемы: чувствительность к выбору слоев экстрактора признаков (для задач, ориентированных на PSNR, требуются поверхностные слои [10]), и увеличение вычислительных затрат из-за попарного сравнения признаков. В EfRLFN вводится составная функция потерь, разработанная для устранения ограничений контрастной потери RLFN с акцентом на структурную точность и сохранение границ. Функция потерь состоит из трех ключевых компонентов:

$$\mathcal{L} = \lambda_{\text{Charb}} \mathcal{L}_{\text{Charb}} + \lambda_{\text{VGG}} \mathcal{L}_{\text{VGG}} + \lambda_{\text{Sobel}} \mathcal{L}_{\text{Sobel}}$$

Реконструкция Восприятие Резкость границ

Функция потерь Charbonnier обеспечивает точность на уровне пикселей, одновременно обеспечивая устойчивость к выбросам и устраняя ограничение потерь L1/L2, используемых на ранних этапах обучения RLFN. Она гарантирует стабильные градиенты даже для больших невязок, что делает её особенно подходящей для задач сверхразрешения.

$$\mathcal{L}_{\text{Charb}} = \sqrt{\| \mathbf{I}_{\text{HR}} - \mathbf{I}_{\text{SR}} \|^2 + \epsilon^2}, \quad \epsilon > 0$$

где $\mathbf{I}_{\text{HR}}$ обозначает эталонное изображение высокого разрешения, а $\mathbf{I}_{\text{SR}}$ обозначает выход модуля SR.

Перцептивная функция потерь на основе VGG использует глубокие признаки из предварительно обученной сети VGG-19 [42]. Данная функция потерь направляет модель к генерации реалистичных результатов.

$$\mathcal{L}_{\text{VGG}} = \| \phi_{\text{VGG}}(\mathbf{I}_{\text{HR}}) - \phi_{\text{VGG}}(\mathbf{I}_{\text{SR}}) \|_1,$$

где $\phi_{\text{VGG}}(\cdot)$ обозначает активации ReLU из слоя *conv5_4* сети VGG-19.

Функция потерь Собеля оптимизирует резкость границ путем штрафования расхождений в картах градиентов между изображениями после сверхразрешения и исходными изображениями.

$$\mathcal{L}_{\text{Sobel}} = \| S(\mathbf{I}_{\text{HR}}) - S(\mathbf{I}_{\text{SR}}) \|_2^2,$$

где $S(\cdot)$ применяет оператор Собеля для извлечения горизонтальных и вертикальных границ. Данная функция потерь устраняет критический пробел в подходе RLFN, который полагается на неявное сохранение границ посредством контрастного обучения на поверхностных признаках. Путем прямого воздействия на качество границ, предложенная функция потерь обеспечивает более четкое восстановление для высокочастотных деталей. Пользовательские исследования подтверждают, что такие результаты соответствуют предпочтениям зрителей, достигая более высоких субъективных оценок [2,32].

При обучении EfRLFN были установлены коэффициенты $\lambda_{\text{Charb}} = 1$, $\lambda_{\text{VGG}} = 10^{-3}$ и $\lambda_{\text{Sobel}} = 10^{-1}$ для нормализации, адаптированные из работы [2].

5. Эксперименты

5.1 Оценивание на тестовой части StreamSR

В табл. 4 (слева) показано, что предложенная модель EfRFLN демонстрирует наилучшее качество на тестовой части StreamSR среди моделей реального времени. Увеличение качества хорошо обобщается для различных объективных метрик и дополнительно подтверждается пользовательским исследованием (рис. 6). Результаты также указывают на то, что модели SPAN, RLFN и ESPCN значительно улучшаются после дообучения на нашем наборе данных: наблюдается заметный положительный разрыв как в объективных, так и в субъективных оценках. После дообучения некоторые из этих моделей превосходят проприетарную модель NVIDIA VSR по пользовательским предпочтениям. Для сравнения также включены некоторые стандартные модели SR, не работающие в реальном времени.

Рис. 6 и 7 предоставляют визуальное сравнение между различными моделями, дополнительно демонстрируя эффективность EfRFLN в суперразрешении реального времени. Визуальные сравнения ясно показывают преимущества EfRFLN: модель производит более четкие изображения с более точными деталями, в то время как конкурирующие модели часто не могут воспроизвести четкие границы и тонкие текстуры, что приводит к заметным артефактам или размытию в сложных областях.

На рис. 8 демонстрируются результаты попарного субъективного сравнения с другими моделями SR: очевидно, что пользователи в значительной степени предпочитают результаты EfRFLN другим методам. В частности, EfRFLN предпочтительнее NVIDIA VSR в 77.4% случаев. Единственные конкурентоспособные альтернативы (SPAN-tuned и RLFN-tuned) также обучены на тренировочном наборе StreamSR.

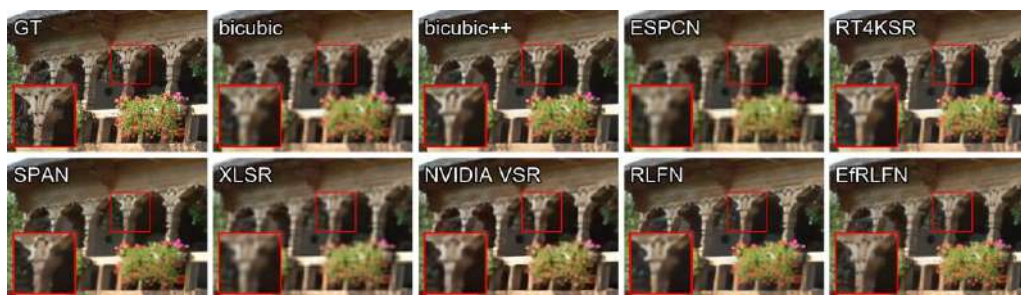


Рис. 6. Сравнение EfRFLN с несколькими 4× моделями реального времени.
Fig. 6. The comparison between EfRFLN and several other 4× real-time SR models.



Рис. 7. Сравнение между несколькими 2× моделями реального времени.
Fig. 7. The comparison between several other 2× real-time SR models.

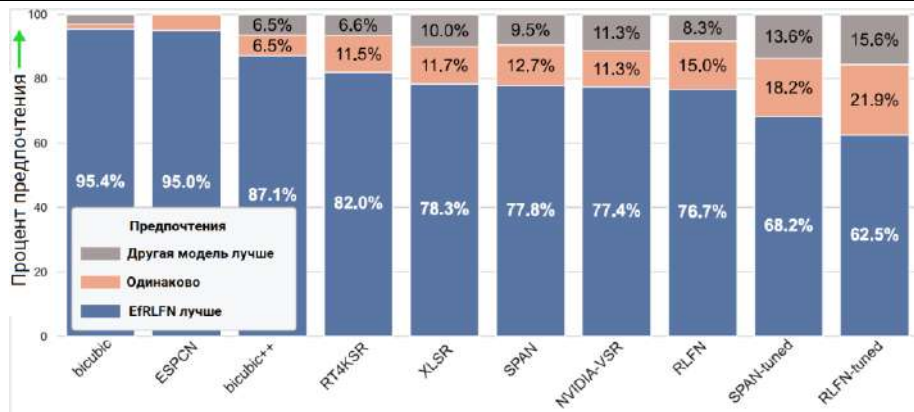


Рис. 8. Предпочтение EfRFLN по сравнению с другими методами сверхразрешения реального времени на основе попарных субъективных оценок.

Fig. 8. Preference for EfRFLN over other real-time super-resolution methods based on pairwise subjective comparisons.

5.2 Валидация на других наборах данных

Была проведена валидация EfRFLN и других методов SR реального времени на пяти широко признанных наборах данных: Set14 [20], BSD100 [21], Urban100 [22], REDS [23] и валидационном наборе DIV2K [6]. Усредненные результаты метрик представлены в табл. 4 (справа). EfRFLN стабильно превосходит все конкурирующие подходы во всех оцененных сценариях, достигая наилучшего качества как для увеличения разрешения 2×, так и 4×.

Проведенные эксперименты показывают, что обучение на предложенном наборе данных приносит пользу не только EfRFLN, но и улучшает производительность других конкурирующих моделей, что подтверждается количественными результатами в табл. 4 и 5 и рис. 10. Это особенно заметно в случаях с SPAN и RLFN.

Анализ функций потерь. На рис. 9 иллюстрируется влияние различных функций потерь на производительность EfRFLN по метрике SSIM в процессе обучения на треке 4× предложенного набора данных. Комбинированная функция потерь (синяя линия) достигает наивысшего значения SSIM, равного 0.865. Это демонстрирует взаимодополняющие преимущества каждой функции потерь. Исключение любого компонента приводит к заметному снижению, особенно удаление потери Charbonnier или VGG, что значительно ухудшает сходимость SSIM. Более простые функции потерь, такие как L_1 , L_2 и LPIPS, показывают худшие результаты на протяжении всего обучения. Примечательно, что предложенный подход превосходит RLFN как по качеству, так и по эффективности обучения. В отличие от RLFN, который использует двухэтапную процедуру обучения с потерями L_1 и контрастной потерей, наша сеть обучается в один этап, достигая превосходного визуального качества при сокращении времени обучения на 16%.

Анализ архитектуры: В табл. 6 представлено исследование функций активации и механизмов внимания в EfRFLN. Комбинация активации гиперболическим тангенсом и механизма внимания ECA демонстрирует наилучшее общее качество, достигая наивысшего SSIM и LPIPS и самой быстрой скорости работы с минимальным количеством параметров. Замена $\tanh$ на смещенную сигмоиду заметно ухудшает SSIM и LPIPS для обоих вариантов внимания. Аналогично, использование более тяжелого блока внимания ESA приводит к снижению скорости работы и в лучшем случае к маргинальному улучшению LPIPS. Эти результаты указывают на то, что активация $\tanh$ в паре с облегченным модулем ECA обеспечивает оптимальный баланс между качеством и эффективностью для сверхразрешения в реальном времени.

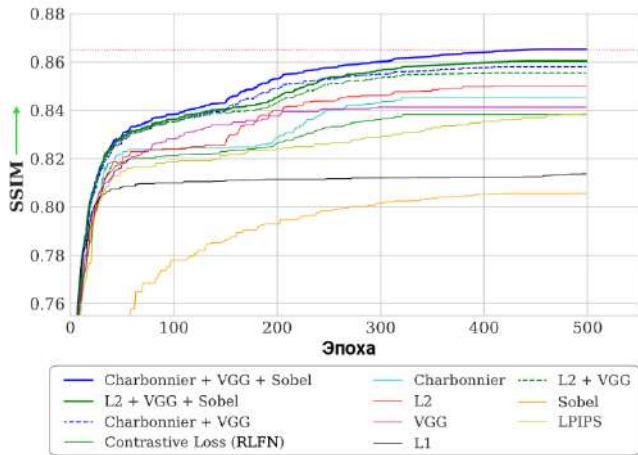


Рис. 9. Влияние функций потерь на обучение EfRLFN на треке 4 × предложенного набора данных. “Contrastive Loss” – оригинальная функция потерь модели RLFN.
 Fig. 9. Impact of Loss Functions on EfRLFN Performance for 4× Upscaling on the Proposed Dataset. “Contrastive Loss” represents the original loss function of the RLFN model.

Табл. 4. Сравнение методов сверхразрешения реального времени с 95% доверительными интервалами на треке увеличения 2X. Наилучшие результаты выделены **полужирным**, вторые лучшие – подчеркиванием. «T» обозначает дообучение на StreamSR. «Subj.» обозначает субъективные
 Table 4. A comparison of real-time super-resolution methods with 95% confidence intervals for the 2× upscaling track. The best results are highlighted in **bold**, and the second best are underlined. “T” indicates fine-tuning on the StreamSR dataset. “Subj.” denotes subjective Bradley-Terry scores.

Модель	StreamSR				Другие бенчмарки			FPS↑
	Subj.↑	PSNR↑	LPIPS↓	CLIP-IQA↑	BSD100	Urban100	DIV2K	
AsConvSR ^T	1.93±0.11	35.25±0.15	0.214±0.008	0.48±0.02	0.832/0.270	0.824/0.208	0.883/0.206	213±8
RT4KSR	2.40±0.12	36.45±0.16	0.213±0.009	0.49±0.02	0.805/0.296	0.769/0.244	0.860/0.221	102±5
RT4KSR ^T	2.43±0.11	37.55±0.14	0.070±0.003	0.53±0.02	0.812/0.254	0.766/0.223	0.864/0.186	102±5
bicubic	2.44±0.14	30.32±0.21	0.076±0.005	0.44±0.03	0.752/0.386	0.711/0.324	0.820/0.292	1829±50
ESPCN	2.48±0.12	30.71±0.18	0.078±0.004	0.45±0.02	0.774/0.534	0.505/0.437	0.514/0.515	201±7
ESPCN ^T	2.49±0.11	35.76±0.15	0.072±0.003	0.51±0.02	0.814/0.236	0.840/0.150	0.862/0.180	201±7
Bicubic++ ^T	2.44±0.13	36.79±0.15	0.087±0.004	0.52±0.02	0.768/0.360	0.725/0.298	0.832/0.270	<u>1629±45</u>
NVIDIA VSR	2.57±0.14	37.40±0.13	0.082±0.003	0.56±0.02	0.788/0.243	0.786/0.152	0.858/0.180	52±3
XLSR ^T	2.56±0.12	37.25±0.14	0.230±0.010	0.47±0.02	0.817/0.244	0.828/0.158	0.864/0.183	429±15
SMFANet ^T	2.37±0.14	37.14±0.15	0.158±0.007	0.51±0.02	0.803/0.239	0.798/0.153	0.865/0.177	327±12
SAFMN ^T	2.26±0.15	37.09±0.15	0.122±0.006	0.53±0.02	0.813/0.236	0.808/0.150	0.871/0.175	273±10
RLFN	2.17±0.15	37.03±0.15	0.086±0.004	0.54±0.02	0.805/0.238	0.803/0.147	0.876/0.175	225±8
RLFN ^T	2.69±0.13	37.63±0.12	0.072±0.003	0.58±0.02	0.834/0.239	0.845/0.153	0.881/0.178	225±8
SPAN	2.55±0.12	37.45±0.13	0.066±0.003	0.57±0.02	0.836/ <u>0.222</u>	0.841/0.139	<u>0.887/0.168</u>	60±3
SPAN ^T	<u>3.13±0.15</u>	<u>37.73±0.12</u>	<u>0.063±0.003</u>	<u>0.61±0.02</u>	0.837/0.239	<u>0.847/0.118</u>	0.890/0.175	60±3
EfRLFN^T	3.33±0.14	37.85±0.11	0.059±0.003	0.65±0.02	0.847/0.190	0.856/0.116	0.892/0.145	271±10
Модели SR, не предназначенные для работы в реальном времени								
Real-ESRGAN	3.87±0.13	37.65±0.12	0.048±0.007	0.66±0.02	0.857/0.172	0.867/0.112	0.901/0.137	9±1
BasicVSR++	4.87±0.13	38.05±0.13	0.037±0.007	0.70±0.02	0.860/0.158	0.868/0.110	0.907/0.131	15±1
COMISR	4.32±0.14	38.15±0.12	0.033±0.006	0.67±0.03	0.894/0.161	0.871/0.108	0.913/0.125	7±1

Табл. 5. Сравнение методов сверхразрешения реального времени с 95% доверительными интервалами на треке увеличения 4X. Наилучшие результаты выделены **полужирным**, вторые лучшие – подчеркиванием. «T» обозначает дообучение на StreamSR. «Subj.» обозначает субъективные оценки по Брэдли-Терри.

Table 5. A comparison of real-time super-resolution methods with 95% confidence intervals for the 4× upscaling track. The best results are highlighted in **bold**, and the second best are underlined. "T" indicates fine-tuning on the StreamSR dataset. "Subj." denotes subjective Bradley-Terry scores.

	StreamSR				Другие бенчмарки			
Модель	Subj.↑	PSNR↑	LPIPS↓	CLIP-IQA↑	BSD100	Urban100	DIV2K	FPS↑
					SSIM↑/LPIPS↓			
AsConvSR ^T	1.32±0.14	32.6±0.15	0.382±0.008	0.38±0.02	0.433/0.586	0.500/0.512	0.701/0.471	213±8
RT4KSR	1.40±0.14	32.65±0.16	0.382±0.009	0.36±0.02	-	-	-	102±5
RT4KSR ^T	1.45±0.14	32.65±0.14	0.382±0.003	0.39±0.02	0.409/0.575	0.489/0.521	0.696/0.489	102±5
bicubic	1.49±0.13	32.8±0.21	0.240±0.005	0.40±0.024	0.423/0.589	0.493/0.589	0.562/0.484	1829±50
ESPCN	1.52±0.11	32.04±0.18	0.276±0.004	0.41±0.02	0.434/0.671	0.432/0.597	0.539/0.572	201±7
ESPCN ^T	1.96±0.15	32.08±0.15	0.215±0.003	0.44±0.02	0.442/0.662	0.452/0.585	0.551/0.557	201±7
Bicubic++ ^T	1.86±0.13	33.12±0.15	0.265±0.004	0.39±0.02	-	-	-	<u>1629±45</u>
NVIDIA VSR	2.31±0.12	33.55±0.13	0.207±0.003	0.47±0.02	-	-	-	52±3
XLSR ^T	2.17±0.12	33.44±0.12	0.263±0.010	0.43±0.02	0.659/0.450	0.672/ <u>0.341</u>	0.759/0.339	429±15
SMFANet ^T	2.02±0.14	33.06±0.14	0.183±0.007	0.43±0.02	0.612/0.482	0.608/0.450	0.722/0.380	327±12
SAFMN ^T	1.95±0.15	33.38±0.15	0.199±0.006	0.42±0.02	0.632/0.468	0.632/0.434	0.736/0.372	273±10
RLFN	2.24±0.16	<u>34.42±0.15</u>	0.247±0.004	0.42±0.02	0.603/0.492	0.597/0.460	0.721/0.386	225±8
RLFN ^T	<u>4.32±0.13</u>	33.82±0.12	<u>0.182±0.003</u>	<u>0.51±0.02</u>	<u>0.674/0.445</u>	<u>0.688/0.336</u>	<u>0.774/0.334</u>	225±8
SPAN	1.87±0.15	32.47±0.12	0.377±0.003	0.46±0.02	0.632/ <u>0.481</u>	0.639/0.369	0.739/0.341	60±3
SPAN ^T	3.14±0.12	33.51±0.12	0.206±0.003	0.49±0.02	0.649/0.483	0.650/0.406	0.750/0.375	60±3
EfRLFN^T	4.52±0.13	34.55±0.11	0.173±0.003	0.58±0.02	0.682/0.429	0.699/0.327	0.778/0.331	271±10

Табл. 6. Исследование функций активации и модулей внимания для EfRLFN. Наилучшие результаты выделены **полужирным**. S(x) обозначает сигмоиду.

Table 6. An Exploration of Activation Functions and Attention Modules for EfRLFN. The best results are highlighted in **bold**. S(x) denotes the sigmoid function.

Компоненты		Метрики			
Активация	Внимание	SSIM ↑	LPIPS ↓	FPS ↑	Параметры ↓
<i>tanh</i>	ECA	0.865	0.173	314	0.37M
<i>tanh</i>	ESA	0.863	0.171	234	0.4M
<i>S(x) – 0.5</i>	ECA	0.856	0.179	305	0.37M
<i>S(x) – 0.5</i>	ESA	0.852	0.181	237	0.4M
<i>ReLU</i>	ECA	0.847	0.184	303	0.37M
<i>ReLU</i>	ESA	0.849	0.182	235	0.4M

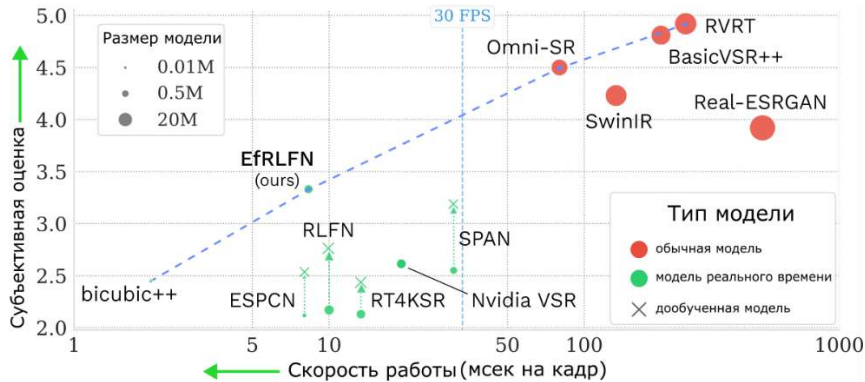


Рис. 10. Соотношение между пользовательской оценкой предпочтения и скоростью работы для различных моделей сверхразрешения. Все модели увеличивают разрешение в $2\times$ раза. Синей линией обозначен Парето-оптимальный фронт. Модели, достигающие производительности реального времени, выделены зелёным цветом, тогда как более медленные модели – красным. Знаком "x" отмечены модели, дообученные на наборе данных StreamSR.

Fig. 10. Trade-off between user preference score and inference speed for various super-resolution models. All models perform $2\times$ upscaling. The blue line indicates the Pareto-optimal frontier. Models achieving real-time performance are highlighted in green, while slower models are shown in red. An "x" marker denotes models fine-tuned on the StreamSR dataset.

6. Заключение

Проведённое исследование продвигает область SR реального времени за счёт введения комплексного многомасштабного набора данных из 5,200 сжатых видео, специально разработанного для потоковых приложений. На этом наборе данных проведено систематическое тестирование 11 моделей SR, каждая из которых была предобучена на тренировочной части. Данные субъективных оценок качества работы моделей SR могут служить ценным материалом для обучения методов оценки качества видео.

Кроме того, разработана эффективная модель EfRFLN, содержащая новые архитектурные решения. Предложена стратегия обучения на основе составной функции потерь, демонстрирующая лучшую сходимость. Расширенная оценка демонстрирует превосходное качество EfRFLN и практическое повышение эффективности по сравнению с существующими методами.

Данная работа предоставляет основу для развития исследований в области сверхразрешения реального времени. Опубликованные ресурсы, включая данные субъективных оценок, открывают возможности для будущих работ как в области сверхразрешения, так и в области субъективной оценки качества.

Код и веса разработанных моделей, а также видео из собранного набора данных опубликованы [43].

Список литературы / References

- [1]. Grand View Research. Video Streaming Market Size & Share Industry Report, 2030. Available at: <https://www.grandviewresearch.com/industry-analysis/video-streaming-market>, 2025, accessed 08.11.2025.
- [2]. Bogatyrev E., Molodetskikh I., Vatin D.S. SR+Codec: a Benchmark of Super-Resolution for Video Compression Bitrate Reduction. 35th British Machine Vision Conference 2024, BMVC 2024, Glasgow, UK, November 25-28, 2024. BMVA, 2024.
- [3]. Li Y. et al. Comisr: Compression-informed video super-resolution. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2543-2552.

- [4]. Conde M.V. et al. Efficient Deep Models for Real-Time 4K Image Super-Resolution. NTIRE 2023 Benchmark and Report. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2023, pp. 1495-1521.
- [5]. Ren B. et al. The ninth NTIRE 2024 efficient super-resolution challenge report. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 6595-6631.
- [6]. Agustsson E., Timofte R. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017.
- [7]. Xue T. et al. Video enhancement with task-oriented flow. *International Journal of Computer Vision*. Springer, 2019, vol. 127, issue 8, pp. 1106-1125.
- [8]. Choi B. Pixel Perfect: RTX Video Super Resolution Now Available for GeForce RTX 40 and 30 Series GPUs. Available at: <https://blogs.nvidia.com/blog/rtx-video-super-resolution/>, 2023, accessed 08.11.2025.
- [9]. Wan C. et al. Swift parameter-free attention network for efficient super-resolution. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 6246-6256.
- [10]. Kong F. et al. Residual local feature network for efficient super-resolution. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 766-776.
- [11]. Bilecen B.B., Ayazoglu M. Bicubic++: Slim, slimmer, slimmest-designing an industry-grade super-resolution network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 1623-1632.
- [12]. Guo J. et al. Asconvsr: Fast and lightweight super-resolution network with assembled convolutions. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 1582-1592.
- [13]. Zamfir E., Conde M.V., Timofte R. Towards real-time 4k image super-resolution. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 1522-1532.
- [14]. Zhang Z. et al. Tmp: Temporal motion propagation for online video super-resolution. *IEEE Transactions on Image Processing*. IEEE, 2024.
- [15]. Liang J. et al. Swinir: Image restoration using swin transformer. Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 1833-1844.
- [16]. Zhang X., Zhang Y., Yu F. HiT-SR: Hierarchical transformer for efficient image super-resolution. *European Conference on Computer Vision*. Springer, 2025, pp. 483-500.
- [17]. Wang Y. et al. Compression-aware video super-resolution. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 2012-2021.
- [18]. Wang X. et al. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 1905-1914.
- [19]. Wang Y., Inguva S., Adsumilli B. YouTube UGC dataset for video compression research. 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2019, pp. 1-5.
- [20]. Zeyde R., Elad M., Protter M. On single image scale-up using sparse-representations. *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7*. Springer, 2012, pp. 711-730.
- [21]. Silva A. BSD100 dataset. Available at: <https://www.kaggle.com/datasets/asilva1691/bsd100>, 2001, accessed 08.11.2025.
- [22]. Huang J.-B., Singh A., Ahuja N. Single image super-resolution from transformed self-exemplars. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 5197-5206.
- [23]. Nah S. et al. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2019, pp. 1-10.
- [24]. Achiam J. et al. Gpt-4 technical report. arXiv preprint. Available at: <https://arxiv.org/abs/2303.08774>, accessed 08.11.2025.
- [25]. Li D., Jiang T., Jiang M. Unified quality assessment of in-the-wild videos with mixed datasets training. *International Journal of Computer Vision*. Springer, 2021, vol. 129, issue 4, pp. 1238-1257.
- [26]. Zhai X. et al. Sigmoid loss for language image pre-training. Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 11975-11986.
- [27]. Talab M.A., Awang S., Najim S.A.M. Super-low resolution face recognition using integrated efficient sub-pixel convolutional neural network (ESPCN) and convolutional neural network (CNN). 2019 IEEE international conference on automatic control and intelligent systems (I2CACIS). IEEE, 2019, pp. 331-335.

- [28]. Zheng M. et al. SMFANet: A Lightweight Self-Modulation Feature Aggregation Network for Efficient Image Super-Resolution. ECCV. 2024.
- [29]. Ayazoglu M. Extremely lightweight quantization robust real-time single-image super resolution for mobile devices. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 2472–2479.
- [30]. Sun L. et al. Spatially-Adaptive Feature Modulation for Efficient Image Super-Resolution. ICCV. 2023.
- [31]. Wang Z. et al. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing. IEEE, 2004, vol. 13, issue 4, pp. 600-612.
- [32]. Zhang R. et al. The unreasonable effectiveness of deep features as a perceptual metric. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 586-595.
- [33]. Ke J. et al. Musiq: Multi-scale image quality transformer. Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 5148–5157.
- [34]. Wang J., Chan K.C., Loy C.C. Exploring clip for assessing the look and feel of images. Proceedings of the AAAI conference on artificial intelligence. 2023, vol. 37, pp. 2555-2563.
- [35]. Huynh-Thu Q., Ghanbari M. Scope of validity of PSNR in image/video quality assessment. Electronics letters. IET, 2008, vol. 44, no. 13, pp. 800–801.
- [36]. Subjectify.us: Crowd-sourced subjective quality evaluation platform, 2025. Available at: <https://subjectify.us/>, accessed 08.11.2025.
- [37]. Bradley R.A., Terry M.E. Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika. JSTOR, 1952, vol. 39, issue 3/4, pp. 324-345.
- [38]. Fleiss, J. L. Measuring nominal scale agreement among many raters. Psychological bulletin 76.5, 1971, p. 378.
- [39]. Clevert D.-A., Unterthiner T., Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint, 2015. Available at: <https://arxiv.org/abs/1511.07289v5>, accessed 08.11.2025. vol. 4, issue 5. p. 11.
- [40]. Isola P. et al. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125-1134.
- [41]. Wang X. et al. ESRGAN: Enhanced super-resolution generative adversarial networks. Proceedings of the European conference on computer vision (ECCV) workshops, 2018, pp. 1-16.
- [42]. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint, 2014. Available at: <https://arxiv.org/abs/1409.1556>, accessed 08.11.2025.
- [43]. EfRLLFN real-time Super-Resolution. Available at: <https://github.com/EvgeneyBogatyrev/EfRLLFN>, accessed 08.11.2025.

Информация об авторах / Information about authors

Евгений Николаевич БОГАТЫРЕВ получил степень бакалавра по прикладной математике и информатике в Московском государственном университете имени М. В. Ломоносова в 2023 году. В настоящее время является студентом магистратуры в лаборатории компьютерной графики и мультимедиа МГУ. В область его научных интересов входят суперразрешение, оценка качества видео, методы сжатия видео и мультимодальное видео-языковое моделирование.

Evgeney Nikolaevich BOGATYREV received his B.S. degree in computer science from the Lomonosov Moscow State University in 2023. Currently, he is working on his master's degree at Lomonosov Moscow State University, Graphics & Media Lab. His research interests involve super-resolution, video quality assessment, video compression methods, and video-language modeling.

Иван Андреевич МОЛОДЕЦКИХ окончил аспирантуру по специальности математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей в Московском государственном университете имени М. В. Ломоносова в 2024 году. В настоящее время работает над диссертацией и является научным сотрудником в центре искусственного интеллекта МГУ, лаборатории компьютерной графики и мультимедиа. В область его научных интересов входят суперразрешение, семантическое матирование видео и машинное обучение. Иван курировал разработку бенчмарка методов суперразрешения для

улучшения качества видео и был одним из организаторов соревнований по оценке качества суперразрешения видео на ECCV-AIM 2024 и ICCV-AIM 2025.

Ivan Andreevich MOLODETSKIKH completed his postgraduate coursework in computer science in the Moscow State University in 2024. He is currently working on his dissertation and is a researcher at the MSU AI Center, Graphics & Media Lab. His research interests include super-resolution, semantic video matting and machine learning. Ivan supervised the development of MSU video upscalers and SR+codec benchmarks and was one of the organizers of the ECCV-AIM 2024 and the ICCV-AIM 2025 video super-resolution quality assessment challenges.

Халед Набиль АБУД получил степень бакалавра компьютерных наук в Московском государственном университете имени М. В. Ломоносова в 2024 году. В настоящее время продолжает обучение в магистратуре МГУ в Лаборатории компьютерной графики и мультимедиа. Его научные интересы включают обработку изображений, оценку качества изображений и видео, состязательное машинное обучение и глубокое обучение. Халед также является одним из ключевых участников проекта «Доверенный ИИ» ИСП РАН, занимаясь анализом устойчивости методов оценки качества изображений и нейросетевых кодеков сжатия изображений к состязательным атакам.

Khaled Nabil ABUD received his B.S. degree in Computer Science from Lomonosov Moscow State University in 2024. Currently, he is working on his master's degree at Lomonosov Moscow State University, Graphics & Media Lab. His research interests include image processing, image and video quality assessment, adversarial machine learning and deep learning. Khaled is also one of the key contributors to ISP RAS Trusted AI project analyzing adversarial robustness of image quality assessment methods and neural image compression codecs.

Дмитрий Сергеевич ВАТОЛИН закончил ВМК МГУ в 1996, защитил диссертацию в 2000, кандидат физико-математических наук, заведующий лабораторией компьютерной графики ВМК МГУ. Специализируется на исследованиях в области алгоритмов сжатия видео, современных методах измерения качества и обработке цифрового видео. Читает курсы по компьютерной графике и методам сжатия и обработки видео с 1997 года. Создатель популярных сайтов, посвященных алгоритмам обработки и сжатия видео.

Dmitriy Sergeevich VATOLIN – Cand. Sci. (Phys.-Math.). He received his degree in 2000 from Moscow State University and is currently head of the CS MSU Graphics & Media Lab. His research interests include compression methods, video processing, and 3D-video techniques (depth from motion, focus and other cues, video matting, background restoration, and high-quality stereo generation), as well as 3D-video quality assessment (metrics for 2D-to-3D-conversion artifacts, temporal asynchrony, swapped views, and more). He is the creator of popular websites devoted to video processing and compression algorithms.

DOI: 10.15514/ISPRAS-2026-38(1)-16



AnAT: состязательное дообучение с использованием якорей для метрик оценки качества изображений

А.Е. Гушин, ORCID: 0000-0002-4055-7394 <alexander.gushchin@graphics.cs.msu.ru>

А.В. Анциферова, ORCID: 0000-0002-1272-5135 <Aantsiferova@graphics.cs.msu.ru>

Д.С. Ватолин, ORCID: 0000-0002-8893-9340 <dm.vatolin@ispras.ru>

*Институт перспективных исследований проблем
искусственного интеллекта и интеллектуальных систем МГУ имени М.В. Ломоносова,
Россия, 119192, г. Москва, Ломоносовский проспект, 27, кор. 1.*

*Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.*

*Московский государственный университет имени М.В. Ломоносова,
Россия, 119991, Москва, Ленинские горы, д. 1.*

Аннотация. Оценка качества изображения с эталоном (FR-IQA) имеет важное значение для сжатия, восстановления и генеративного моделирования изображений, однако существующие нейронные метрики по-прежнему работают медленно и очень уязвимы для враждебных возмущений. Мы представляем Anchored Adversarial Training (AnAT) – теоретически обоснованную стратегию защиты, которая использует чистые «якорные» образцы и ранжирование для явного ограничения точечной ошибки прогнозирования при атаках. AnAT может быть легко интегрирован в существующие архитектуры FR-IQA для повышения их надежности без значительного ухудшения производительности на чистых данных. Обширные эксперименты на пяти общедоступных тестовых наборах FR-IQA демонстрируют, что AnAT значительно улучшает устойчивость к невиданным атакам типа «белый ящик», повышая SROCC с 0,30–0,57 до 0,60–0,84 на KADID-10k, при этом сохраняя или превосходя точность незащищенных базовых моделей. Насколько нам известно, это первая техника состязательного обучения, разработанная для моделей FR-IQA.

Ключевые слова: состязательное дообучение; оценка качества изображений; машинное обучение.

Для цитирования: Гушин А.Е., Анциферова А.В., Ватолин Д.С. AnAT: состязательное дообучение с использованием якорей для метрик оценки качества изображений. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 241–254. DOI: 10.15514/ISPRAS-2026-38(1)-16.

Благодарности: Работа выполнялась с использованием суперкомпьютера «МГУ-270» МГУ имени М.В. Ломоносова.

AnAT: Anchor-Based Adversarial Training for Image Quality Assessment

A.E. Gushchin, ORCID: 0000-0002-4055-7394 <alexander.gushchin@graphics.cs.msu.ru>

A.V. Antsiferova, ORCID: 0000-0002-1272-5135 <Aantsiferova@graphics.cs.msu.ru>

D.S. Vatolin, ORCID: 0000-0002-8893-9340 <dm.vatolin@ispras.ru>

*MSU Institute for Artificial Intelligence,
27k1, Lomonosovsky Prospekt, Moscow, 119192, Russia.*

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

*Lomonosov Moscow State University,
1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. Full-Reference image quality assessment (FR-IQA) is essential for image compression, restoration, and generative modeling, yet current neural metrics remain slow and highly vulnerable to adversarial perturbations. We introduce Anchored Adversarial Training (AnAT) – a theoretically grounded defense strategy that employs clean “anchor” samples and a ranking-based regularization to explicitly bound pointwise prediction error under attacks. AnAT can be seamlessly integrated into existing FR-IQA architectures to enhance their robustness without degrading performance on clean data. Extensive experiments on five public FR-IQA benchmarks demonstrate that AnAT significantly improves resistance to unseen white-box attacks, lifting SROCC from 0.30–0.57 to 0.60–0.84 on KADID-10k, while maintaining or exceeding the accuracy of unprotected baselines. To our knowledge, this is the first adversarial training framework tailored specifically for full-reference IQA models.

Keywords: adversarial training; image quality assessment; machine learning.

For citation: Gushchin A.E., Antsiferova A.V., Vatolin D.S. AnAT: Anchor-Based Adversarial Training for Image Quality Assessment. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 241-254 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-16.

Acknowledgements. The research was carried out using the MSU-270 supercomputer of Lomonosov Moscow State University.

1. Введение

Оценка качества изображений (Image Quality Assessment, IQA) – фундаментальная задача компьютерного зрения с применениями в восстановлении изображений, сжатии и генеративном моделировании. Полно-референсная (Full-Reference, FR) IQA оценивает воспринимаемое качество, сравнивая искажённое изображение с его безупречным эталоном. Классические подходы, такие как оценка пикового отношения сигнал-шум (Peak Signal-to-Noise Ratio, PSNR) и индекс структурного сходства (Structural Similarity Index, SSIM), быстры, но упускают многие сложные детали человеческого восприятия, что поддерживает интерес к методам на основе глубокого обучения.

Тем не менее даже сильные нейросетевые модели IQA сталкиваются с двумя насущными проблемами: (i) невысокой скоростью вывода, ограничивающей применение в реальном времени, и (ii) высокой уязвимостью к состязательным возмущениям, подрывающей надёжность в критически важных сценариях.

Состязательные атаки вносят малозаметные возмущения, которые вводят в заблуждение нейросетевые модели IQA. Несмотря на внедрение относительно новых методов защиты для FR-IQA, системы оценки устойчивости показывают, что многие метрики остаются уязвимыми [1], что делает их непригодными для таких областей, как медицинская визуализация, автономное вождение и аутентификация контента, где оценки должны оставаться надёжными как при состязательных, так и при обычных возмущениях. Кроме того,

эти уязвимости позволяют манипулировать результатами поиска изображений, поскольку поисковые системы (например, Bing) опираются на метрики IQA для ранжирования [2]. Подобные атаки также могут искажать результаты публичных оценок [3, 4], эксплуатируя слабости моделей IQA и искусственно повышая кажущееся качество алгоритмов. Например, использование уязвимой метрики IQA в качестве функции потерь в задачах восстановления изображений может ухудшать реальное качество [5] или вызывать визуальные артефакты [6]. Эти риски подчёркивают насущную необходимость подходов FR-IQA, сочетающих точность и устойчивость к атакам.

В этой работе мы предлагаем Anchored Adversarial Training (AnAT) – теоретически обоснованный метод защиты для моделей FR-IQA. AnAT дообучает базовую модель IQA так, чтобы сохранялся порядок предсказаний под атакой относительно чистых «якорных» образцов. Наш теоретический анализ связывает целевую функцию якорной оптимизации с верхней оценкой максимальной точечной ошибки предсказания.

Обширные эксперименты на стандартных бенчмарках IQA показывают, что AnAT повышает устойчивость к разнообразным, ранее не встречавшимся атакам белого ящика, при сохранении сильной точности на чистых данных. Основные результаты авторы сводятся к следующему:

- Anchored Adversarial Training (AnAT). Теоретически обоснованная схема, использующая чистые якорные образцы и ранговую функцию потерь для ужесточения оценки ошибки предсказания; повышает SROCC под атаками на величину до 0.30 по сравнению с незащищёнными моделями и до 0.05 по сравнению с предыдущими методами защиты.
- Комплексные эмпирические результаты. На пяти публичных FR-IQA бенчмарках и четырёх невиданных ранее атаках белого ящика AnAT стабильно улучшает интегральные показатели устойчивости (до 12%) без ухудшения точности на чистых наборах данных.

2. Обзор литературы

2.1 Полно-референсная оценка качества изображений (FR-IQA)

Оценка качества изображений важна для множества приложений, включая сжатие, суперразрешение и другие методы обработки. FR-IQA оценивает качество, связанное с восприятием, сравнивая искажённое изображение с его безупречным эталоном. Хотя оценка PSNR вычисляется быстро, ее корреляция со зрительной системой человека (HVS) невысока. Метрики SSIM [7], MS-SSIM [8], FSIM [9], SR-SIM [10] и VIF [11] учитывают структуру, фазу, салиентность или видимость для лучшего соответствия восприятию при низких затратах. Эти методы опираются на эвристики и аналитические признаки, обеспечивая вычислительную эффективность.

Глубокое обучение далее улучшило качество предсказаний с помощью моделей LPIPS [12] и DISTS [13]. Трансформерные подходы продолжают эту линию: SwinIQA [14], IQT [15], ANIQ [16] и современная SOTA-модель TOPIQ [17], использующая многомасштабную схему сверху-вниз с механизмом Cross-Self Attention [41]. Ряд современных работ исследуют многомасштабное внимание и межмасштабный обмен: одни полагаются на тяжёлые трансформерные конвейеры, другие – на более лёгкие архитектуры; применяются как передачи сверху-вниз, так и снизу-вверх, а также их комбинации. Помимо многомасштабного внимания, изучаются методы обучения ранжированию качества: система RankIQA [18] обучает сиамскую сеть для NR-IQA. Насколько нам известно, сочетание ранжирования с состязательным обучением именно для модели FR-IQA остаётся малоизученным направлением.

2.2 Робастные методы IQA

В FR-IQA атака добавляет малозаметное возмущение, вводя метрику в заблуждение. Различают атаки белого ящика (известны градиенты и параметры) и чёрного ящика (доступны только выходные оценки) [18]. Специальные для моделей IQA сценарии атак предложены в [20-24].

Подходы к защите делятся на сертифицируемые и эмпирические. Первые обеспечивают формальные гарантии, но чаще всего слишком медленны для работы в реальном времени. Эмпирические стратегии практичнее и обычно опираются на состязательное обучение или «очистку» входа. Для NR-IQA изучались, в частности, входные «очистки» (E-LPIPS [24]), состязательное дообучение (метрика R-LPIPS [25]; метод AT [26]) и модификации архитектур (Grad.Norm [27]). В отличие от предыдущих вариантов AT, мы рассматриваем якорную ранговую формулировку состязательного обучения, направленную на ужесточение верхней оценки точечной ошибки предсказания и повышение устойчивости при атаках.

3. Метод

3.1 Anchored Adversarial Training

Мы предлагаем стратегию состязательного дообучения AnAT, которая задействует состязательные примеры, не вводя прямого штрафа по меткам качества. Подход опирается на предположение, что состязательные возмущения либо незаметны, либо едва заметны, что является наиболее распространённым сценарием в реальных атаках. Ключевая идея – использовать наличие чистых примеров (изображений с надёжно предсказанным качеством) как «якорей» для обучения.

Постановка задачи.

Системы обработки изображений всё чаще работают в недоверенной среде, где малозаметные возмущения способны манипулировать оценками качества.

Мы формализуем атакующего, добавляющего ограниченный по норме l_p шум (δ) ($\|\delta\|_p \leq \epsilon$) к искажённому изображению x_d из пары (x_r, x_d) .

Если большие значения соответствуют лучшему качеству, атакующий стремится максимизировать $f_\theta(x_r, x_d + \delta)$; если меньшие значения означают более высокое качество, противник стремится его уменьшить. Тот же подход может решать и противоположную задачу, как показано в [28]. Эта постановка не ограничивает общность результатов, поскольку принципы применимы симметрично.

Формально состязательная атака определяется как

$$\max_{\|\delta\|_p \leq \epsilon} f_\theta(x_r, x_d + \delta)$$

Для данной модели f_θ , обучающего набора D с парами изображений и ассоциированной меткой качества y , а также функции потерь L , базовое состязательное обучение формулируется как задача min–max [26]:

$$m \in E_{(x_r, x_d, y) \sim D} \left[\max_{\|\delta\|_p \leq \epsilon} L(f_\theta(x_r, x_d + \delta), y) \right].$$

Внутренняя максимизация генерирует сильные состязательные примеры $x_d + \delta$, а внешняя минимизация подстраивает параметры модели θ для повышения устойчивости.

В задачах классификации истинная метка не меняется под воздействием атаки. В оценке качества, однако, сами значения могут слегка изменяться вместе с содержанием, поэтому

прямое применение приводит к сдвигу меток. Ранее это обходили, вводя штраф по меткам или их рескейлинг¹ [26], что порождает две практические проблемы:

- субъективные эксперименты используют разные протоколы, из-за чего согласование шкал MOS (Mean Opinion Score) приходится повторять для каждого набора данных;
- метрика, по которой выполняется штрафование, сама может быть уязвимой, что открывает путь переносу атаки.

3.2 Якорная ранговая функция потерь

Чтобы использовать надёжные «якоря» (чистые изображения без атаки), определим якорную ранговую функцию потерь:

$$\mathcal{L}_{\text{anchor}}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{|S|N} \sum_{i \in S} \sum_{j=1}^N \frac{\max\{0, R(\mathbf{y}, \tilde{\mathbf{y}})_{i,j}\}}{|R(\mathbf{y}, \tilde{\mathbf{y}})|_{\infty}}$$

где $R(\mathbf{y}, \tilde{\mathbf{y}})$ – покомпонентная ранговая матрица с элементами $R(\mathbf{y}, \tilde{\mathbf{y}})_{i,j} = (y_i - y_j) \text{sign}(\tilde{y}_j - \tilde{y}_i)$, N – размер мини-батча², $|S|$ – число якорей.

Интуитивно функция штрафует любые отклонения в предсказанном порядке относительно истинного, выделяя сравнения с чистыми образцами. Использование надёжных якорей из подмножества S позволяет эффективнее контролировать предсказания на состязательных примерах. В указанной формулировке получаем следующий результат:

Обозначения.

Пусть $\mathbf{y} = (y_1, \dots, y_N) \in \mathbb{R}^N$ – истинные оценки MOS, а $\tilde{\mathbf{y}} = (\tilde{y}_1, \dots, \tilde{y}_N)$ – предсказания модели на текущем мини-батче.

Пусть $S \subseteq \{1, \dots, N\}$ – множество индексов якорей, то есть чистых образцов в мини-батче. Остальные образцы атакованы.

Допущения.

Существуют константы $\lambda, \epsilon, \Delta > 0$ такие, что:

- Плотность якорей. Значения оценок MOS для якорей отсортированы, и соседние якоря отличаются не более чем на λ :

$$\Delta \leq y_{i_{k+1}} - y_{i_k} \leq \Delta \text{ для всех } i_k \in S$$

- Точность модели для якорных элементов. $|\tilde{y}_i - y_i| \leq \epsilon$ для всех $i \in S$.
- Распределение оценок MOS. $|y_i - y_j| \geq \Delta$ для всех $i \neq j$.
- Якорная ранговая потеря. Введём покомпонентную ранговую матрицу $R_{i,j}(\mathbf{y}, \tilde{\mathbf{y}}) = (y_i - y_j) \text{sign}(\tilde{y}_j - \tilde{y}_i)$ и норму $|R|_{\infty} := \max_{i,j} |R_{i,j}|$.

$$\text{Тогда } \mathcal{L}_{\text{anchor}}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{|S|N} \sum_{i \in S} \sum_{j=1}^N \frac{\max\{0, R_{i,j}(\mathbf{y}, \tilde{\mathbf{y}})\}}{|R|_{\infty}}.$$

$$\text{Обозначим также } E := \max_{1 \leq j \leq N} |\tilde{y}_j - y_j|, R := \max_{i,j} |y_i - y_j|.$$

Теорема 1 (Граница точечной ошибки через якорную ранговую потерю).

$$\text{Если } \mathcal{L}_{\text{anchor}}(\mathbf{y}, \tilde{\mathbf{y}}) < \delta, \text{ то } E \leq \epsilon + \lambda \left(1 + \sqrt{\frac{2\delta NR}{\Delta}} \right).$$

¹ Рейскейлинг (rescaling) – техника нормализации, которая преобразует значения признаков в некоторый стандартный диапазон, например, в диапазон $[0, 1]$.

² Мини-батч – это небольшое подмножество обучающих данных, используемое для приближенных вычислений.

Доказательство:

1. Выбор наихудшего примера. Положим $j^* := \operatorname{argmax}_{1 \leq j \leq N} |\tilde{y}_j - y_j|$, $e := \tilde{y}_{j^*} - y_{j^*}$ так что $|e| = E$. Случай $e < 0$, симметричен; далее предполагаем $e > 0$.
2. Определение «перерасхода». Положим

$$\tau := e - \epsilon - \lambda.$$
3. Если $e \leq \epsilon + \lambda$, то неравенство тривиально выполнено; далее считаем $\tau > 0$.
4. Подсчёт нарушенных сравнений с якорями. Так как соседние якоря отстоят не более чем на λ , то каждое наблюдение находится не далее, чем на $\lambda/2$ от некоторого якоря. Сдвиг предсказания для j^* ещё на $\tau + \lambda/2 > \lambda/2$ переворачивает порядок по меньшей мере с $m := \lceil \frac{\tau}{\lambda} \rceil \geq \frac{\tau}{\lambda}$ якорями, чьи истинные оценки MOS строго больше y_{j^*} .
5. Нижняя оценка вклада в функцию потерь. Для каждого такого якоря i имеем $y_i > y_{j^*}$, но $\tilde{y}_{j^*} > \tilde{y}_i$, следовательно $R_{i,j^*}(y, \tilde{y}) \geq y_i - y_{j^*}$. По допущению о распределении MOS каждая нарушенная пара даёт вклад не менее Δ , а суммарно $\sum_{k=1}^m (y_{j^*} + k\Delta - y_{j^*}) = \Delta \frac{m(m+1)}{2} \geq \frac{\Delta m^2}{2}$.
6. Связь с $\mathcal{L}_{anchor}$. Используется только один столбец (соответствующий j^*), поэтому

$$\mathcal{L}_{anchor} \geq \frac{\frac{\Delta m^2}{2}}{NR} = \frac{\Delta m^2}{2NR}.$$
7. Оценка для τ . Из условия $\mathcal{L}_{anchor} < \delta$ следует $\frac{\Delta m^2}{2NR} < \delta$. Подставляя $m \geq \tau/\lambda$, получаем $\tau^2 < \frac{2\delta NR}{\Delta} \lambda^2$.
8. Переход к e и E . Так как $\tau = e - \epsilon - \lambda$ и $E = |e|$, то $e \leq \epsilon + \lambda + \lambda \sqrt{\frac{2\delta NR}{\Delta}}$, неравенство, указанное в теореме.

Теорема 1 требует лишь (i) верхней границы на расстояние между якорями λ – чего тривиально добиться, отсортировав мини-батч и выбрав равномерно разнесённые примеры; и (ii) допуска по точности якорей (ϵ), который автоматически выполняется для хорошо обученной модели на чистых изображениях. Не требуются ни статистики конкретного набора данных, ни рескейлинг MOS, ни сертифицированные гарантии, так что условия выполняются для любого практического FR-IQA набора данных.

3.2.1 Численный пример

Рассмотрим набор данных с оценками MOS в интервале $[0, 100]$, почти совершенную модель IQA ($\epsilon = 0.1$) и размер мини-батча $N=16$. Пусть при формировании каждого батча расстояние между соседними якорями не превышает $\lambda = 0.5$, а минимальный шаг по MOS равен $\Delta = 0.4$; тогда $R \leq \lambda N = 8$. Если якорная ранговая потеря равна $\delta = 0.01$, то по условию Теоремы 1 получаем $|\tilde{y}_j - y_j| \leq 0.1 + 0.5 \left(1 + \sqrt{\frac{2 \cdot 0.01 \cdot 16 \cdot 8}{0.4}} \right) = 0.1 + 0.5 \times 3.53 \approx 1.86$.

3.3 Сходимость якорной ранговой функции потерь и проверка теоретических границ

На рис. 1 (а) показана динамика якорной ранговой функции потерь $\mathcal{L}_{anchor}$ на первых 1,000 шагах оптимизации (мини-батчах). Показаны две кривые: значение потери по мини-батчам (синяя) и её экспоненциальное скользящее среднее с окном в 20 итераций (оранжевая). К итерации ≈ 650 сглаженная кривая опускается ниже 10^{-3} и остаётся там до конца обучения, наблюдается лишь небольшое «дребезжание» по мини-батчам (относительная амплитуда

менее 10%). Это соответствует целевому уровню $\delta = 10^{-3}$, используемому в Теореме 1, так что теоретическая граница на точечную ошибку предсказания гарантируется уже менее чем через две эпохи. График подтверждает, что функция потерь при якорном состязательном дообучении сходится быстро и стабильно, достигая требуемо низких уровней потери без длительной подстройки гиперпараметров.

На рис. 1 (b) сравниваются теоретическая граница из Теоремы 1 и наблюдаемая максимальная точечная ошибка в ходе дообучения AnAT на наборе данных KADID-10k. Эмпирическая кривая нигде не превышает теоретическую границу и убывает с тем же экспоненциальным темпом, что даёт наглядное подтверждение корректности полученной оценки.

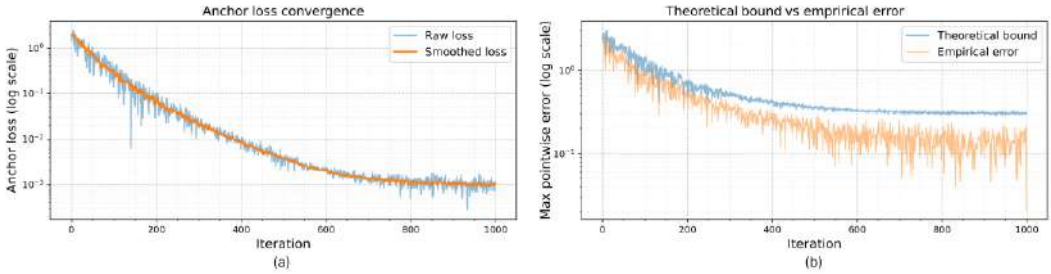


Рис. 1. (a) Сходимость якорной функции потерь за 1000 итераций. (b): сравнение границ, предоставленных теоремой 1, с эмпирическими значениями максимальных точечных ошибок.
Fig.1. (a) Convergence of the Anchored Ranking Loss over 1,000 iterations. (b): comparison of bounds, provided by Theorem 1 with empirical values of maximum pointwise errors.

4. Детали реализации

4.1 Состязательное дообучение

Во время состязательного дообучения каждый мини-батч формируется из узкой полосы по оценкам MOS, что ограничивает как размах меток R , так и межякорное расстояние λ .

В пределах этой полосы случайное подмножество изображений оставляется чистым (якоря), тогда как оставшиеся подвергаются атаке. Мы считаем, что модель уже достаточно хорошо предсказывает качество для чистых изображений, поэтому ошибка для якорных элементов ϵ остаётся пренебрежимо малой. Поскольку R и λ ограничены шириной полосы, граница из Теоремы 1 упрощается до выражения, где доминирует λ ; следовательно, более узкие полосы ужесточают гарантию и ускоряют сходимость.

Итоговая функция потерь имеет вид $\mathcal{L}_{AnAT} = \frac{1}{2}\mathcal{L}_{anchor} + \frac{1}{2}\mathcal{L}$.

Такой составной штраф одновременно обеспечивает устойчивое относительное ранжирование изображений даже под состязательными возмущениями и продолжает ограничивать абсолютную ошибку предсказания.

4.2 Наборы данных

Для сравнения нашего подхода с текущими решениями мы представляем ряд экспериментальных результатов. Мы провели оценки как внутри наборов данных, так и между ними, а также выполнили подробное сравнение устойчивости к атакам. Эксперименты выполнены на нескольких публичных наборах: LIVE [29], CSIQ [30], TID2013 [31], KADID-10k [32], PIPAL [33] и наборе данных с двухальтернативным вынужденным выбором (2AFC): BAPPS [12]. При наличии мы использовали официальные разбиения на train/val/test и усредняли результат по 10 запускам со случайными разбиениями в пропорции 6/2/2. Разбиения формировались по эталонным изображениям, чтобы исключить пересечение контента.

4.3 Метрики оценки

Мы оцениваем качество с помощью двух широко используемых корреляционных показателей для наборов с MOS: линейного коэффициента корреляции Пирсона (PLCC) и рангового коэффициента Спирмена (SROCC). Оба показателя лежат в диапазоне $[-1,1]$, положительное значение означает прямую взаимосвязь. Большой SROCC указывает на более точное ранжирование, тогда как большой PLCC – на более точное приближение абсолютных значений. Дополнительно применяется метод выборки с замещением (10^3 повторных выборок) для проверки статистической значимости различий по SROCC.

4.4 Методология сравнения устойчивости к атакам

Мы сравниваем эффективность предложенного состязательного дообучения для различных FR-IQA-моделей. Для состязательного дообучения и тестирования использовались обучающая и тестовая части набора KADID-10k; в качестве метода атаки – Projected Gradient Descent с 10 итерациями (PGD-10) [34], бюджет атаки $\epsilon = 8/255$. Различные варианты состязательного обучения сравниваются по SROCC и по интегральному показателю робастности IR-Score [26] на чистых и атакованных данных.

Показатель *IR-Score* оценивает способность модели выдерживать возмущения разной силы, как рекомендовано в [25]. Состязательные примеры генерировались при величинах возмущения $\epsilon \in E = \{2,4,8,10\}/255$. Оценки нормировались min-max-масштабированием и сопоставлялись в едином пространстве с помощью модели нейронного оптимального транспорта, чтобы учесть различия распределений. *IR-Score* вычисляется как

$$IRscore = R_f - R_g; R_g = \frac{1}{N/E} \sum_{i=1}^N \sum_{\epsilon \in E} (g(x_r^{(i)}, x_d^{(i)}) - g(x_r^{(i)}, x_d^{(i)} + \delta_{g,\epsilon}^{(i)})).$$

Здесь $(x_r^{(i)}, x_d^{(i)})$ – пары эталон/искажённое изображение, $\delta_{g,\epsilon}^{(i)}$ – возмущение с бюджетом ϵ , сгенерированное против модели $g \in \{f, \hat{f}\}$; где $\hat{f}$ – вариант после состязательного дообучения, N – число изображений. Большие значения *IRscore* соответствуют большему среднему снижению падения качества под атакой по сравнению с базовой моделью.

5. Результаты

Мы сравниваем четыре варианта трёх полно-референсных FR-IQA моделей (LPIPS, TOPIQ и ANIQ) под распространёнными атаками белого ящика с нормой l_∞ . Рассматриваемые варианты:

- (1) **base**: без состязательного дообучения;
- (2) **R**:- базовое состязательное обучение;
- (3) **AT**:- состязательное обучение с «разглаживанием» меток [26];
- (4) **AnAT**-(предложенный): Anchored Adversarial Training.

Во всех вариантах AT и AnAT в обучении использовалась PGD-атака на 10 итераций (PGD-10) с бюджетом $\epsilon = 8/255$.

На этапе тестирования мы оцениваем четыре невиданных атаки: FGSM [36], C&W [37], AutoAttack (AA) [38] и перцептивную атаку FACPA [39]. Устойчивость измеряется по SROCC и IR-Score на наборе данных KADID-10k.

Табл. 1 показывает, что AnAT достигает результатов уровня SOTA по робастности и превосходит другие подходы на 0.02–0.06 пункта SROCC, с сопоставимыми преимуществами по IR-Score. AnAT также даёт наилучший SROCC на неиспорченной тестовой выборке по сравнению с другими методами защиты. Без каких-либо защит TOPIQ демонстрирует более высокую устойчивость, чем LPIPS и ANIQ, опережая их по IR-Score на 0.02–0.03. Хотя AnAT требует больше времени на фазе состязательного дообучения, он обеспечивает наилучшие суммарные результаты.

Таблица 1. Устойчивость состязательно обученных моделей IQA на KADID-10k. SROCC оценивается при $\epsilon = 8/255$; для IR-Score используется $\epsilon \in \{2, 4, 8, 10\}/255$. Полужирным шрифтом выделены лучшие результаты для каждой модели. Все состязательные варианты обучались только с PGD-10. Table 1. Robustness of adversarially trained IQA models on KADID-10k. SROCC is evaluated at $\epsilon = 8/255$; IR-Score uses $\epsilon \in \{2, 4, 8, 10\}/255$. Bold numbers denote the best result for each model. All adversarial variants were trained only with PGD-10.

Model	SROCC $\uparrow$					IR-Score $\uparrow$				Train time (min)
	Clean	FGSM	C&W	AutoAttack	FACPA	FGSM	C&W	AutoAttack	FACPA	
base LPIPS	0.893	0.542	0.260	0.239	0.496	—	—	—	—	46
R-LPIPS	0.858	0.570	0.327	0.266	0.515	0.541	0.403	0.385	0.507	123
AT-LPIPS	0.852	0.730	0.523	0.481	0.753	0.722	0.596	0.510	0.613	101
AnAT-LPIPS (ours)	0.865	0.753	0.589	0.536	0.810	0.783	0.652	0.582	0.630	345
base AHIQ	0.938	0.524	0.284	0.269	0.512	—	—	—	—	352
R-AHIQ	0.879	0.533	0.305	0.314	0.509	0.572	0.431	0.456	0.520	437
AT-AHIQ	0.892	0.839	0.513	0.552	0.760	0.763	0.615	0.550	0.611	514
AnAT-AHIQ (ours)	0.917	0.844	0.593	0.580	0.811	0.802	0.691	0.601	0.645	558
base TOPIQ	0.954	0.568	0.295	0.350	0.503	—	—	—	—	105
R-TOPIQ	0.902	0.571	0.291	0.357	0.502	0.563	0.427	0.414	0.525	218
AT-TOPIQ	0.907	0.801	0.573	0.560	0.788	0.769	0.638	0.551	0.620	205
AnAT-TOPIQ (ours)	0.943	0.836	0.614	0.602	0.819	0.811	0.690	0.614	0.657	267

5.1 Результаты для 2AFC наборов данных

Мы оценили четыре варианта трёх полно-референсных FR-IQA моделей (LPIPS, TOPIQ и AHIQ) на наборе данных BAPPS (двухальтернативный вынужденный выбор, 2AFC). Во всех вариантах AT и AnAT использовалась PGD-атака на 10 итераций (PGD-10) с бюджетом $\epsilon = 8/255$. На этапе тестирования модели подвергались семи невиданным ранее атакам: FGSM, C&W, AutoAttack (AA), FACPA, перцептивная атака Zhang et al., SquareAttack и Parsimonious; всего 5 атак белого ящика и 2 – чёрного ящика. Для всех тестовых атак использовался одинаковый бюджет $\epsilon = 8/255$. Мы сообщаем точность (ассигасу) на чистой и атакованной версиях тестовой части BAPPS.

Табл. 2 показывает, что AnAT достигает уровня SOTA по робастности и превосходит другие подходы на 0.02–0.10 пункта точности. AnAT также обеспечивает наилучшую точность на неатакованной тестовой выборке по сравнению с другими методами защиты. Даже без защит AHIQ модель демонстрирует большую устойчивость, чем LPIPS и TOPIQ, опережая их по метрике ‘accuracy’ на 0.03–0.06.

Таблица 2. Точность на тестовом наборе BAPPS (2AFC) для различных техник состязательного обучения, применённых к моделям LPIPS, TOPIQ и AHIQ. Лучшие результаты для каждой модели выделены полужирным шрифтом. Во время обучения использовался PGD-10 ($\epsilon = 8/255$).

Table 2. Accuracy on 2AFC BAPPS test set of different adversarial training techniques, which were applied to LPIPS, TOPIQ and AHIQ models. The best results for each model are bolded. PGD-10 with $\epsilon = 8/255$ was used during training.

Model	White-Box Attacks						Black-Box Attacks	
	Clean	FGSM	C&W	AutoAttack	FACPA	Zhang et al.	SquareAttack	Parsimonious
base LPIPS	0.742	0.260	0.102	0.135	0.415	0.301	0.511	0.513
R-LPIPS	0.728	0.487	0.306	0.298	0.471	0.375	0.567	0.533
AT-LPIPS	0.729	0.495	0.440	0.412	0.589	0.436	0.615	0.586
AnAT-LPIPS (ours)	0.736	0.520	0.486	0.459	0.632	0.494	0.643	0.618
base AHIQ	0.784	0.358	0.320	0.341	0.496	0.416	0.542	0.552
R-TOPIQ	0.762	0.420	0.391	0.350	0.523	0.459	0.589	0.581
AT-AHIQ	0.768	0.493	0.467	0.479	0.584	0.510	0.634	0.614
AnAT-AHIQ (ours)	0.775	0.526	0.544	0.552	0.612	0.546	0.658	0.653
base TOPIQ	0.794	0.405	0.350	0.376	0.521	0.462	0.571	0.574
R-TOPIQ	0.771	0.491	0.382	0.417	0.545	0.526	0.614	0.587
AT-TOPIQ	0.771	0.573	0.473	0.485	0.570	0.581	0.662	0.672
AnAT-TOPIQ (ours)	0.782	0.594	0.561	0.580	0.637	0.610	0.690	0.703

5.2 Сравнение с методами защиты с помощью очищения

В табл. 3 сравнивается предложенная техника AnAT с методами состязательной «очистки» (purification). Эти подходы не модифицируют саму модель IQA, а выполняют предварительную обработку входных изображений. Мы сопоставляем AnAT в связке с нашей FR-IQA моделью с базовыми приёмами предобработки (Random Flip, Random Rotate), а также со специализированной защитой DiffPure [40]. Результаты показывают, что AnAT превосходит все методы «очистки», за исключением случая SquareAttack, где DiffPure достигает наилучшего результата. Вероятная причина – наибольшее отличие возмущений SquareAttack от профиля PGD-10, использованного в процессе состязательного дообучения, что ведёт к большему «сдвигу домена» между обучением и тестированием.

Таблица 3. Точность (%) на тестовом наборе BAPPS 2AFC для AnAT-LPIPS по сравнению с базовыми методами состязательной очистки. Лучшие результаты выделены полужирным. AnAT-LPIPS обучался с PGD-10 ($\epsilon = 8/255$).

Table 3. Accuracy on 2AFC BAPPS test set for AnAT-LPIPS compared to some adversarial purification methods. The best results are bolded. PGD-10 with $\epsilon = 8/255$ was used during training of AnAT-LPIPS.

Model	White-Box Attacks						Black-Box Attacks	
	Clean	FGSM	C&W	AutoAttack	FACPA	Zhang et al.	SquareAttack	Parsimonious
base LPIPS	0.794	0.405	0.350	0.376	0.521	0.462	0.571	0.574
base LPIPS + Random Flip	0.751	0.471	0.467	0.422	0.569	0.570	0.606	0.608
base LPIPS + Random Rotate	0.720	0.446	0.431	0.390	0.534	0.512	0.541	0.553
base LPIPS + DiffPure	0.741	0.522	0.498	0.452	0.619	0.603	0.712	0.698
AnAT-LPIPS (ours)	0.782	0.594	0.561	0.580	0.637	0.610	0.690	0.703

6. Заключение

В работе предложен метод Anchored Adversarial Training (AnAT) для полно-референсной оценки качества изображений, который объединяет якорное ранжирование с состязательным дообучением и даёт теоретически обоснованную границу на точечную ошибку предсказания под атаками. Показано, что якорная ранговая функция потерь контролирует перевероты порядка относительно «чистых» образцов и тем самым ограничивает максимальное отклонение предсказаний; эмпирически наблюдаемая динамика ошибки согласуется с полученной границей.

Экспериментальная оценка на пяти общедоступных FR-IQA бенчмарках и под невиданными ранее атаками белого (и, для 2AFC-набора, чёрного) ящика демонстрирует, что AnAT стабильно повышает робастность моделей без деградации на чистых данных. В частности, на KADID-10k AnAT увеличивает SROCC под атаками по сравнению с базовыми моделями и альтернативными вариантами AT, улучшая также интегральный показатель устойчивости IR-Score; на BAPPS достигается прирост ассигасу по сравнению с другими методами. При этом подход интегрируется в существующие архитектуры (LPIPS, TOPIQ и компактная CNN-модель) и сохраняет или улучшает их качество на неизменённых выборках.

Практически значимо, что AnAT не требует согласования шкал MOS между наборами данных и обходится без прямого штрафа по меткам качества для атакованных примеров, что упрощает перенос между наборами данных и снижает риск «переноса» уязвимостей через используемую метрику. Вместе с тем авторы отмечают, что цена за прирост устойчивости – дополнительное время на фазе состязательного дообучения.

Ограничения и направления развития включают: расширение анализа за пределы выбранных бюджетов и типов атак; распространение методики на NR-IQA и видео-качество; автоматизацию выбора/планирования «полос» MOS и стратегии отбора якорей; изучение компромисса между числом якорей, сходимостью и итоговой границей ошибки; а также исследование сочетания AnAT с сертифицируемыми защитами.

Список литературы / References

- [1]. A. Gushchin, K. Abud, G. Bychkov, E. Shumitskaya, A. Chistyakova, S. Lavrushkin, B. Rasheed, K. Malyshev, D. Vatolin, and A. Antsiferova, Guardians of Image Quality: Benchmarking Defenses Against Adversarial Attacks on Image Quality Metrics, in Proc. 42nd Int. Conf. on Machine Learning (ICML), Vancouver, Canada, Jul. 2025, poster.
- [2]. Microsoft. 2013. A Behind the Scenes Look at How Bing is Improving Image Search Quality. Available at: <https://blogs.bing.com/search-quality-insights/2013/08/23/a-behind-the-scenes-look-at-how-bing-is-improving-image-search-quality>, accessed 12.10.2025.
- [3]. Huang, Ziqi, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Wei Liu, Cewu Lu, and Yipeng Qin. 2024. VBench: Comprehensive Benchmark Suite for Video Generative Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21807-21818.
- [4]. Wu, Rongyuan, Tao Yang, Lingchen Sun, Zhengqiang Zhang, Shuai Li, and Li Song. 2024. SeeSR: Towards Semantics-Aware Real-World Image Super-Resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 25456-25467.
- [5]. Ding, Keyan, Kede Ma, Shiqi Wang, and Eero P. Simoncelli. 2021. Image Quality Assessment: Unifying Structure and Texture Similarity. *International Journal of Computer Vision* 129(4): pp. 1258-1281.
- [6]. Kashkarov, Egor, Egor Chistov, Ivan Molodetskikh, and Dmitriy Vatolin. 2024. Can Full-Reference Image-Quality-Assessment Methods Serve as Perceptual Losses for Super-Resolution? Available at: <https://arxiv.org/abs/2405.20392>, accessed 12.10.2025.
- [7]. Wang, Zhou, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13(4): pp. 600-612.
- [8]. Wang, Zhou, Eero P. Simoncelli, and Alan C. Bovik. 2003. Multiscale Structural Similarity for Image Quality Assessment. In The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003, pp. 1398-1402. IEEE.
- [9]. Zhang, Lin, Lei Zhang, Xuanqin Mou, and David Zhang. 2011. FSIM: A Feature Similarity Index for Image Quality Assessment. *IEEE Transactions on Image Processing* 20(8): pp. 2378-2386.
- [10]. Zhang, Lin, and Hongyu Li. 2012. SR-SIM: A Fast and High Performance IQA Index Based on Spectral Residual. In 2012 19th IEEE International Conference on Image Processing, pp. 1473-1476. IEEE.
- [11]. Sheikh, Hamid R., and Alan C. Bovik. 2006. Image Information and Visual Quality. *IEEE Transactions on Image Processing* 15(2): pp. 430-444.
- [12]. Zhang, Richard, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 586-595.
- [13]. Ding, Keyan, Kede Ma, Shiqi Wang, and Eero P. Simoncelli. 2022. Image Quality Assessment: Unifying Structure and Texture Similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(5): pp. 2567-2581.
- [14]. Liu, Jianzhao, Xin Li, Yanding Peng, Tao Yu, and Zhibo Chen. 2022. SwinIQA: Learned Quality Assessment Using a Swin Transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1795-1799.
- [15]. Cheon, Manri, Sung-Jun Yoon, Byungyeon Kang, and Junwoo Lee. 2021. Perceptual Image Quality Assessment with Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 433-442.
- [16]. Lao, Shanshan, Yuan Gong, Shuwei Shi, Sidi Yang, Tianhe Wu, Jiahui Chen, Shi Guo, and Zhou Wang. 2022. Attentions Help CNNs See Better: Attention-Based Hybrid Image Quality Assessment Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1140-1149.
- [17]. Hosu, Vlad, Hanhe Lin, and Dietmar Saupe. 2020. TOPIQ: A Top-Down Approach to 3D-to-2D Image Quality Assessment. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 2003-2012.
- [18]. Liu, Yuming, Fan Zhang, and David R. Bull. 2017. RankIQA: Learning from Rankings for No-Reference Image Quality Assessment. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5178-5187.
- [19]. A. Gushchin, K. Abud, G. Bychkov, E. Shumitskaya, A. Chistyakova, S. Lavrushkin, B. Rasheed, K. Malyshev, D. Vatolin, and A. Antsiferova, Guardians of Image Quality: Benchmarking Defenses Against Adversarial Attacks on Image Quality Metrics, in Proc. 42nd Int. Conf. on Machine Learning (ICML), Vancouver, Canada, Jul. 2025, poster.

- [20]. Korhonen, Joni, and David M. Bull. 2019. Attacking Image Quality Assessment: Adversarial Examples for No-Reference IQA. In 2019 IEEE International Conference on Image Processing (ICIP), pp. 2344-2348. IEEE.
- [21]. Shumitskaya, Anastasia, Dmitriy Vatolin, and others. 2023. Adversarial Attacks against No-Reference IQA Models. In CVPR Workshops, 2023.
- [22]. Shumitskaya, Anastasia, and Dmitriy Vatolin. 2023. FACPA: Fast Adversarial Corruption and Patch Attack on No-Reference Image- and Video-Quality Metrics. In *Tiny Papers @ ICLR 2023*, Kigali, Rwanda, May 5, 2023. OpenReview.net.
- [23]. Zhang, Richard, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. ELPIPS: Enhanced Learned Perceptual Image Patch Similarity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.
- [24]. Zhang, Richard, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. ELPIPS: Enhanced Learned Perceptual Image Patch Similarity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.
- [25]. Agarwal, Siddhant, et al. 2020. R-LPIPS: Robust Learned Perceptual Image Patch Similarity. In ECCV Workshops.
- [26]. Chistyakova, Anna, Anastasia Antsiferova, Maksim Khrebtov, Sergey Lavrushkin, Denis Turdakov, Dmitriy Vatolin, and Denis Turdakov. 2024. Increasing the Robustness of Image Quality Assessment Algorithms to Adversarial Attacks via Gradient-Norm Regularization. *Technologies* 12: 220. Available at: <https://doi.org/10.3390/technologies12110220>, accessed 12.10.2025.
- [27]. Liu, Yujia, Chenxi Yang, Dingquan Li, Jianhao Ding, and Tingting Jiang. 2024. Defending Against Adversarial Attacks on No-Reference Image Quality Models with Gradient Norm Regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 25554-25563.
- [28]. Antsiferova, Anastasia, Khaled Abud, Aleksandr Gushchin, Ekaterina Shumitskaya, Georgii Bychkov, Sergey Lavrushkin, and Dmitriy Vatolin. 2024. Comparing the Robustness of Modern No-Reference IQA Metrics to Adversarial Attacks. In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 700-708.
- [29]. Sheikh, Hamid, Muhammad Sabir, and Alan Bovik. 2006. A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms. *IEEE Transactions on Image Processing* 15: 3440–51. DOI: 10.1109/TIP.2006.881959.
- [30]. Larson, Eric, and Damon Chandler. 2010. Most Apparent Distortion: Full-Reference Image Quality Assessment and the Role of Strategy. *Journal of Electronic Imaging* 19: 011006. DOI: 10.1117/1.3267105.
- [31]. Ponomarenko, Nikolay, O. Ieremeiev, Vladimir Lukin, Karen Egiazarian, J. Astola, B. Vozel, et al. 2013. Color Image Database TID2013: Peculiarities and Preliminary Results. In 2013 European Workshop on Visual Information Processing, EUVIP 2013, pp. 106-111.
- [32]. Lin, Hanhe, Vlad Hosu, and Dietmar Saupe. 2019. KADID-10k: A Large-Scale Artificially Distorted IQA Database. In 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), 1–3. DOI: 10.1109/QoMEX.2019.8743252.
- [33]. Gu, Jinjin, Cai Haoming, Chen Haoyu, Ye Xiaoxing, Jimmy S. Ren, and Chao Dong. 2020. “PIPAL: A Large-Scale Image Quality Assessment Dataset for Perceptual Image Restoration.” In *Computer Vision – ECCV 2020*. August 23-28, 2020, Proceedings, Part XI 16, pp. 633-651. Springer.
- [34]. Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. Available at: <https://arxiv.org/pdf/1706.06083>, accessed 12.10.2025.
- [35]. Carlini, Nicholas, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On Evaluating Adversarial Robustness. Available at: <https://arxiv.org/pdf/1902.06705>, accessed 12.10.2025.
- [36]. Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. Available at: <https://arxiv.org/pdf/1412.6572>, accessed 12.10.2025.
- [37]. Carlini, Nicholas, and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In 2017 IEEE Symposium on Security and Privacy (SP), pp. 39-57. IEEE.
- [38]. Croce, Francesco, and Matthias Hein. 2020. Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-Free Attacks. In International Conference on Machine Learning, pp. 2206-2216. PMLR.
- [39]. Shumitskaya, Anastasia Antsiferova, and Dmitriy Vatolin. 2023. FACPA: Fast Adversarial Corruption and Patch Attack on No-Reference Image- and Video-Quality Metrics. In *Tiny Papers @ ICLR 2023*, Kigali, Rwanda, May 5, 2023. OpenReview.net.

- [40]. Nie, Weili, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. 2022. DiffPure: Diffusion Models for Adversarial Purification. Available at: <https://arxiv.org/pdf/2205.07460>, accessed 12.10.2025.
- [41]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Информация об авторах / Information about authors

Александр Евгеньевич ГУЩИН получил степень магистра по прикладной математике и информатике в Московском государственном университете имени М. В. Ломоносова в 2024 году. В настоящее время является студентом аспирантуры в лаборатории компьютерной графики и мультимедиа. В область его научных интересов входят методы оценивания качества видео и изображений, а также исследование устойчивости нейросетевых моделей и методов их защиты.

Aleksandr Evgenievich GUSHCHIN received his M.S. degree in computer science from the Moscow State University in 2024. He is currently a postgraduate student at the MSU Graphics & Media Lab. His research interests involve image and video processing, quality assessment, and machine learning. Aleksandr is also a key contributor to the project analyzing video quality assessment methods, including their robustness to adversarial attacks.

Анастасия Всеволодовна АНЦИФЕРОВА аспирантка ВМК МГУ. Окончила магистратуру ВМК МГУ по специальности анализ больших данных в 2018 году. В настоящее время она является аспирантом и участником видеогруппы лаборатории графики и мультимедиа МГУ. Сфера ее научных интересов включает анализ и оптимизацию видеокодексов, оценку субъективного качества стереоскопического видео. Анастасия является одним из организаторов проекта международного сравнения видеокодексов, ежегодно проводимого в МГУ и проекта измерения качества 3D-видео VQMT3D.

Anastasia Vsevolodovna ANTSHIFEROVA received her M.S. degree in computer science from Moscow State University in 2018. Currently, she is a postgraduate student at Moscow State University and a member of Video Group in MSU Graphics&Media Lab. Her research interests involve video codecs analysis and optimization, stereoscopic video subjective quality assessment. Anastasia is one of the contributors to MSU Video Codec Comparison Project and to the 3D video quality measurement project VQMT3D.

Дмитрий Сергеевич ВАТОЛИН закончил ВМК МГУ в 1996, защитил диссертацию в 2000, кандидат физико-математических наук, заведующий лабораторией компьютерной графики ВМК МГУ. Специализируется на исследованиях в области алгоритмов сжатия видео, современных методах измерения качества и обработке цифрового видео. Читает курсы по компьютерной графике и методам сжатия и обработки видео с 1997 года. Создатель популярных сайтов, посвященных алгоритмам обработки и сжатия видео.

Dmitriy Sergeevich VATOLIN – Cand. Sci. (Phys.-Math.). He received his degree in 2000 from Moscow State University and is currently head of the CS MSU Graphics & Media Lab. His research interests include compression methods, video processing, and 3D-video techniques (depth from motion, focus and other cues, video matting, background restoration, and high-quality stereo generation), as well as 3D-video quality assessment (metrics for 2D-to-3D-conversion artifacts, temporal asynchrony, swapped views, and more). He is the creator of popular websites devoted to video processing and compression algorithms.

DOI: 10.15514/ISPRAS-2026-38(1)-17



Адаптивное восстановление фрагментов видеопотока на основе машинного обучения с учётом динамики сцены

Н.А. Думкин, ORCID: 0009-0006-4280-8253 <ndumkin@hse.ru>

Д.В. Александров, ORCID: 0000-0002-9759-8787 <dvaalexandrov@hse.ru>

М.А. Прозорский, ORCID: 0009-0006-9250-7280 <mprozorskiy@hse.ru>

*Национальный исследовательский университет «Высшая школа экономики»,
Россия, 109028, г. Москва, Покровский бульвар, д. 11.*

Аннотация. Предложен теоретически обоснованный подход к адаптивному восстановлению видеофрагментов на стороне клиента с использованием методов машинного обучения и анализа сцены. Метод включает формальную постановку задачи, модель конечного автомата для принятия решений, функцию стоимости восстановления, а также новый этап в подготовке видео – оценку динамики сцены с последующей записью признака в HLS-плейлист. Такой признак позволяет повысить точность выбора методов восстановления фрагментов видео. Описана архитектура алгоритма, предложены правила выбора стратегии восстановления и теоретически обосновано использование дополнительных признаков в формализованной модели. Работа носит преимущественно теоретико-методический характер: формализуется стратегия восстановления и предлагается расширение плейлиста. Дополнительно приведена ограниченная симуляционная оценка, цель которой – проверить внутреннюю согласованность правил модели и чувствительность к пороговым параметрам, а не продемонстрировать превосходство над существующими методами.

Ключевые слова: адаптивный видеостриминг; восстановление видеоконтента; интерполяция видеок кадров; суперразрешение видео; клиентские алгоритмы восстановления; стандарт HLS.

Для цитирования: Думкин Н.А., Александров Д.В., Прозорский М.А. Адаптивное восстановление фрагментов видеопотока на основе машинного обучения с учётом динамики сцены. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 255–274. DOI: 10.15514/ISPRAS-2026-38(1)-17.

Благодарности: Работа выполнена при поддержке Научно-учебной лаборатории облачных и мобильных технологий Национального исследовательского университета «Высшая школа экономики».

Machine Learning-based Adaptive Reconstruction of Video Stream Fragments Taking into Account Scene Dynamics

N.A. Dumkin, ORCID: 0009-0006-4280-8253 <ndumkin@hse.ru>

D.V. Alexandrov, ORCID: 0000-0002-9759-8787 <dval Alexandrov@hse.ru>

M.A. Prozorskiy, ORCID: 0009-0006-9250-7280 <mprozorskiy@hse.ru>

National Research University Higher School of Economics,
11, Pokrovsky Boulevard, Moscow, 109028, Russia.

Abstract. A theoretically sound approach to adaptive client-side video fragment restoration is proposed using machine learning and scene analysis methods. The method includes a formal problem statement, a finite-state machine model for decision making, a restoration cost function, and a new stage in video preparation: scene dynamics assessment followed by recording a feature in an HLS playlist. This feature improves the accuracy of selecting video fragment restoration methods. The algorithm architecture is described, rules for selecting a restoration strategy are proposed, and the use of additional features in the formalized model is theoretically justified. The work is primarily theoretical and methodological in nature: a restoration strategy is formalized and a playlist extension is proposed. A limited simulation evaluation is also provided, the purpose of which is to verify the internal consistency of the model rules and sensitivity to threshold parameters, rather than to demonstrate superiority over existing methods.

Keywords: adaptive video streaming; video content restoration; video frame interpolation; video super-resolution; client-side restoration algorithms; HLS.

For citation: Dumkin N.A., Alexandrov D.V., Prozorskiy M.A. Machine learning-based adaptive reconstruction of video stream fragments taking into account scene dynamics. *Trudy ISP RAN/Proc. ISP RAS*, vol. 38, issue 1, 2026, pp. 255-274 (in Russian). DOI: 10.15514/ISPRAS-2026-38(1)-17.

Acknowledgements. This work was supported by the Research and Educational Laboratory of Cloud and Mobile Technologies of the National Research University Higher School of Economics.

1. Введение

Потоковая передача видео в настоящее время является одним из доминирующих видов интернет-трафика, и масштаб его потребления продолжает расти [1]. Качество восприятия потокового видео существенно зависит от условий сети: нестабильное подключение приводит к потере фрагментов, «фризам» (замиранию изображения) и резким скачкам качества, что значительно снижает удовлетворённость зрителей [2]. Для поддержания плавного воспроизведения современными алгоритмами адаптивного управления скоростью передачи данных (Adaptive Bitrate, ABR) динамически регулируется качество видео в зависимости от пропускной способности канала [3-4]. Применение методов машинного обучения позволило улучшить эффективность ABR-алгоритмов (например, с использованием обучения с подкреплением в системе Pensieve [4] или обучения с учителем в алгоритме SMASH [5]), однако такие решения фокусируются лишь на выборе качества видео и не решают проблему восстановления утраченного контента. Существуют также подходы, нацеленные на предотвращение видеопазов с помощью прогнозирования и проактивной загрузки сегментов [6], но они не устраняют артефакты уже потерянных фрагментов.

На стороне клиента традиционно применяются методы маскировки потерь, такие как повтор последнего фрагмента или простая интерполяция, однако их эффективность ограничена [7]. С развитием глубокого обучения появились алгоритмы интерполяции видеоряда, способные восстанавливать недостающие участки с высокой достоверностью [8-9]. Развиваются и технологии сверхразрешения видео, позволяющие улучшать качество низкоскоростных видеопотоков: например, метод ABUV [10] интегрирует повышение разрешения на стороне клиента для компенсации качества видеопотока, сниженного в результате адаптивного

управления скоростью передачи данных. Комбинация подобных методов способна заметно повысить качество восприятия видео даже в условиях ограниченной сети.

Недавно появились комплексные подходы, объединяющие восстановление фрагментов с адаптивной передачей. Так, система Repago [11] преднамеренно отбрасывает некоторые фрагменты на стороне отправителя при ограниченной пропускной способности, а на стороне получателя восстанавливает их с помощью нейросетевой интерполяции, что улучшает плавность воспроизведения. Другая работа предложила единую модель 3RE-Net для одновременного исправления потерь и повышения чёткости видео [12]. Также продемонстрирована возможность выполнения подобных методов в реальном времени на мобильных устройствах [13]. Однако существующим решениям недостаёт формального аппарата для принятия решений, учитывающего как условия сети, так и динамику видеосцены.

В данной работе предложен формализованный подход, позволяющий клиентскому плееру адаптивно выбирать стратегию восстановления пропущенных фрагментов (интерполяция, повышение разрешения или повторный запрос) на основе как сетевых метрик, так и содержательных характеристик видеопотока. Основной вклад заключается в введении формальной модели принятия решений с функцией стоимости, интеграции признака динамики сцены в процесс выбора и теоретическом обосновании оптимальности предложенной стратегии.

Предложенный механизм совместим с HLS (HTTP Live Streaming) – стандартом потоковой передачи медиаданных по протоколу HTTP [14] (RFC 8216), поскольку опирается на предусмотренную спецификацией возможность определения пользовательских тегов [15] (RFC 2119). Клиенты, не поддерживающие тег EXT-X-SCENE-DYN, безопасно его игнорируют, что обеспечивает обратную совместимость. Нововведение не изменяет семантику сегментов, а лишь расширяет пространство признаков для алгоритмов адаптации и восстановления, не нарушая синтаксис и правила загрузки манифеста.

Основное внимание в работе уделяется формализации стратегии восстановления и описанию расширения интерфейса плейлиста. Симуляционное исследование служит прежде всего для проверки внутренней согласованности модели и анализа ее чувствительности к вариациям параметров.

Таким образом, основной вклад носит теоретико-методологический характер: формализуется стратегия восстановления и предлагается расширение плейлиста, а симуляция служит для валидации внутренней согласованности модели и ее чувствительности к параметрам, а не для демонстрации превосходства над существующими методами.

Объект, предмет, гипотеза, новизна и значимость

Объект исследования. Процесс адаптивного видеостриминга и восстановления потерянных фрагментов на стороне клиента.

Предмет исследования. Формализованная стратегия выбора метода восстановления на основе сетевых условий, вычислительных ресурсов и динамики сцены.

Гипотеза. Интеграция модели принятия решений в виде конечного автомата с метрикой динамики сцены, передаваемой через расширение индексного файла HLS, может привести к более контекстнообоснованным решениям по восстановлению по сравнению с подходами, учитывающими только сетевые параметры.

Научная новизна. (i) Формализация выбора действия как оптимизации по функции стоимости, учитывающей качество и вычислительные затраты; (ii) модель на основе конечного автомата, сочетающая сетевые признаки, ресурсные ограничения и динамику сцены; (iii) обратно совместимое расширение HLS для инъекции признаков сцены; (iv) аналитическая калибровка пороговых параметров без необходимости обширных обучающих выборок.

Практическая значимость. Спецификация клиентского алгоритма и расширение индексного файла HLS позволяют интегрировать интерполяцию и повышение разрешения в адаптивную логику без изменений серверной части.

2. Постановка задачи

Обозначим последовательность видеосегментов как $V = \{v_1, v_2, \dots, v_n\}$. В момент времени t на стороне клиента доступен вектор признаков $x_t = (d_t, c_t, q_t, s_t, m_t)$, где:

- d_t – длительность серии потерь, измеряемая в числе сегментов, определяемая как:

$$d_t = \begin{cases} 0, L_t = 0 \text{ (нет потери)}, \\ 1, L_t = 1 \text{ (один потерянный сегмент)}, \\ 2, L_t \geq 2 \text{ (два или более потерянных сегмента)}, \end{cases}$$

где L_t – число последовательных потерянных сегментов, обнаруженных на шаге t ;

- c_t – наличие контекста (0: отсутствует, 1: присутствует), определяемое как:

$$c_t = \begin{cases} 1, \text{ если сегменты } v_{i-1} \text{ и } v_{i+1} \text{ успешно получены и декодированы,} \\ 0, \text{ в противном случае (хотя бы один соседний сегмент отсутствует),} \end{cases}$$

где v_i – потерянный сегмент (или первый сегмент в серии потерь);

- q_t – текущий относительный уровень пропускной способности;
- s_t – уровень динамики сцены, предварительно вычисленный при подготовке индексного файла HLS.
- m_t – нормализованная вычислительная мощность устройства (от 0 до 1), характеризующая доступную производительность для постобработки (интерполяции, повышения разрешения) в реальном времени.

Значение $c_t = 1$ является необходимым условием для применения методов интерполяции (S_1, S_2), так как для восстановления требуется контекст с обеих сторон потерянного фрагмента. Порог $L_t \geq 2$ для $d_t = 2$ выбран, поскольку интерполяция более одного подряд потерянного сегмента приводит к недопустимому накоплению ошибки (см. раздел 4.2).

Необходимо выбрать действие a_t из множества возможных действий A , минимизирующее функцию потерь:

$$\min_{a_t \in A} C(a_t | x_t) = \alpha \cdot D_{\text{vis}}(a_t | x_t) + \beta \cdot C_{\text{comp}}(a_t), \quad \text{s.t. } T(a_t; m_t) \leq \Delta t,$$

$$C(a_t | x_t) = \alpha \cdot D_{\text{vis}}(a_t | x_t) + \beta \cdot C_{\text{comp}}(a_t),$$

где $x_t = (d_t, c_t, q_t, s_t, m_t)$ – вектор признаков, D_{vis} – ожидаемая визуальная ошибка (например, индекс структурного сходства, SSIM), C_{comp} – вычислительная/временная стоимость. Ограничение $T(a_t; m_t) \leq \Delta t$ запрещает выбор методов, не укладывающихся в реальное время (здесь Δt – допустимый бюджет задержки для текущего фрагмента). Оптимальное действие выбираем из допустимых: $a_t^* \in \operatorname{argmin}_{a_t \in A: T(a_t; m_t) \leq \Delta t} \alpha D_{\text{vis}} + \beta C_{\text{comp}}$.

2.1 Индикатор пропускной способности сети.

Пусть $b_i = \frac{\text{bytes}_i \cdot 8}{t_{\text{download}}^i}$ – измеренная сквозная скорость загрузки i -го сегмента (бит/с). Оценку устойчивой пропускной способности на момент t получаем гармоническим средним по окну из W сегментов:

$$\hat{T}_t = \frac{W}{\sum_{i=t-W+1}^t \frac{1}{b_i}}.$$

Нормируем $\hat{T}_t$ в $q_t \in [0,1]$ лог-шкалой на опорный диапазон, заданный «лестницей» представлений: $T_{\text{low}} = (1 + \epsilon)B_{\text{min}}, T_{\text{high}} = (1 + \epsilon)B_{\text{max}}$, где $B_{\text{min}}, B_{\text{max}}$ – минимальная и максимальная скорости передачи данных профилей, $\epsilon \in [0.03, 0.10]$ – запас на накладные расходы.

$$q_t = \text{clip} \left(\frac{\ln \hat{T}_t - \ln T_{\text{low}}}{\ln T_{\text{high}} - \ln T_{\text{low}}}, 0, 1 \right).$$

Для устойчивости применяем EWMA-сглаживание (Exponentially Weighted Moving Average): $q_t \leftarrow \eta q_{t-1} + (1 - \eta) q_t$, где $\eta \in [0.3, 0.6]$ (по умолчанию 0.5), а $W = 5$. Для интерпретации порогов удобно использовать обратное отображение в физическую шкалу (бит/с):

$$T(q) = T_{\text{low}} \cdot \left(\frac{T_{\text{high}}}{T_{\text{low}}} \right)^q.$$

3. Теоретическая модель

Для формализации процесса восстановления предлагается модель принятия решений в виде конечного автомата и алгоритм выбора действия на его основе.

3.1 Модель принятия решений

Предлагаемый конечный автомат $\mathcal{A}$ имеет множество состояний S , каждое из которых соответствует определённому способу восстановления видеопотока:

- S_0 : нормальное воспроизведение без потерь;
- S_1 : восстановление простой интерполяцией;
- S_2 : восстановление с помощью ML-интерполяции;
- S_3 : повышение разрешения полученного видео;
- S_4 : повторный запрос недостающих данных;
- S_5 : понижение качества потока.

Переходы зависят от расширенного вектора признаков x_t и включают *ресурсные ограничения*. В частности, при $s_t \rightarrow 1$ интерполяция (S_1, S_2) не применяется из-за резкого падения качества; при q_t высоком и стабильном предпочтителен повторный запрос (S_4), иначе – понижение качества потока (S_5) с возможным повышением разрешения (S_3) при достаточных ресурсах (m_t).

Переходы $\delta(s_t, x_t) \rightarrow s_{t+1}$ определяются набором правил, представленным в табл. 1. На рис. 1 приведена схема состояний и переходов автомата.

Пример правил выбора действия.

- $d_t = 1, c_t = 1, s_t < 0.4 \Rightarrow$ простая интерполяция (переход в состояние S_1).
- $d_t = 1, c_t = 1, s_t \geq 0.4 \Rightarrow$ ML-интерполяция (переход в состояние S_2).
- $d_t > 1$ или $c_t = 0 \Rightarrow$ повторный запрос (переход в состояние S_4).

Каждому сочетанию условий из таблицы соответствует действие a_t , которое в данных условиях обладает минимальной оценкой стоимости $C(a_t | x_t)$ согласно функционалу (1). Таким образом, алгоритм на каждом шаге выбирает локально оптимальную по (1) стратегию восстановления.

Пороговые значения m^{int} и m^{up} определяются согласно разделу 5. Выбор $s^*, s^{\text{hi}}, q_{\text{low}}$ и q_{hi} обоснован аналитической калибровкой (см. раздел 4.2).

3.2 Алгоритм адаптивного восстановления

На основе описанной модели состояний разработан алгоритм, реализующий правила принятия решений в псевдокоде (см. рис. 2). Алгоритм периодически вызывается на клиенте при обнаружении потери фрагмента или существенного ухудшения качества потока.

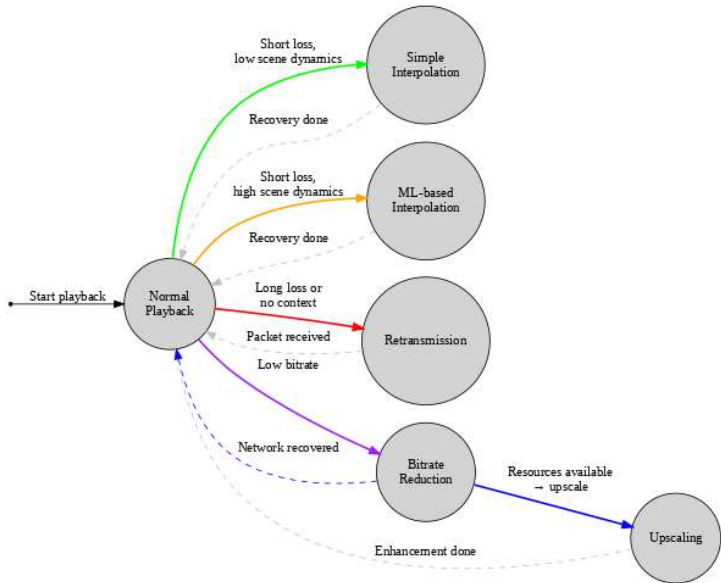


Рис. 1. Структура автомата принятия решений.
Fig. 1. Structure of the decision-making machine.

Табл. 1. Фрагмент таблицы принятия решений автомата.
Table 1. A fragment of the decision-making table of the machine.

d_t	c_t	s_t	q_t	m_t	Выбранное действие a_t
1	1	$< s^*$	$> q_{low}$	*	Простая интерполяция (S_1)
1	1	$[s^*, s^{hi})$	$> q_{low}$	$\geq m^{int}$	ML-интерполяция (S_2)
1	1	$[s^*, s^{hi})$	$> q_{low}$	$< m^{int}$	Простая интерполяция (S_1)
1	1	$\geq s^{hi}$	$> q_{hi}$	*	Повторный запрос (S_4)
2	*	$\geq s^{hi}$	$\leq q_{hi}$	$\geq m^{up}$	Понижение качества потока + повышение разрешения ($S_5 + S_3$)
2	*	$\geq s^{hi}$	$\leq q_{hi}$	$< m^{up}$	Понижение качества потока (S_5)
*	0	*	$> q_{hi}$	*	Повторный запрос (S_4)
$d_t \geq 2$	*	*	*	*	Повторный запрос (S_4)
*	*	*	$\leq q_{low}$	*	Понижение качества потока (S_5)

О пороговых значениях. Числа $s^*, s_{hi}, q_{low}, q_{hi}$ выведены аналитически из уравнений баланса затрат и сетевых ограничений (см. раздел 4.2). Выбранные значения ($s^* = 0.4, s_{hi} = 0.9, q_{low} = 0.2, q_{hi} = 0.6$) представляют типичные рабочие точки для широкого диапазона типов контента и условий сети. Эти пороги не являются статичными и могут динамически корректироваться на основе локальных измерений качества.

3.3 Метрика динамики сцены

Для каждого видеосегмента v_k при его подготовке (на этапе кодирования контента) рассчитывается средняя величина движения s_k :

$$s_k = \frac{1}{N-1} \sum_{i=1}^{N-1} \text{Flow}(f_i, f_{i+1}),$$

где $\text{Flow}(f_i, f_{i+1})$ – оценка оптического потока между последовательными изображениями f_i и f_{i+1} внутри сегмента (N – число изображений в сегменте). Затем значение s_k записывается в плейлист HLS в виде специального тега:

#EXT-X-SCENE-DYN:0.62

Этот дополнительный признак динамики сцены становится частью вектора x_t на стороне клиента и используется при выборе метода восстановления фрагментов (как отражено в правилах выше).

Конкретный метод оценки оптического потока спецификацией не фиксируется. На практике могут применяться как классические методы (TV-L1, Farneback), так и лёгкие нейросетевые реализации (например, RAFT-GMA-Lite или SPyNet). Важным требованием является стабильность работы в условиях высокой компрессии и сохранение времени вычисления, не превышающего бюджет обработки сегмента при подготовке контента. Для унификации поведение $\text{Flow}()$ рекомендуется калибровать на валидационном наборе с типичными артефактами кодека.

3.3.1 Динамика сцены

Для последовательности кадров $f_1, \dots, f_T$ определим

$$s_k = \frac{1}{N-1} \sum_{i=1}^{N-1} \text{Flow}(f_i, f_{i+1}).$$

Нормировку $s_k \in [0,1]$ выполняем как $s_k' = \text{clip}\left(\frac{s_k - p_{05}}{p_{95} - p_{05}}, 0, 1\right)$, где p_{05}, p_{95} – перцентили распределения Flow на обучающей/валидационной выборке (или в «холодном старте» на первых F фрагментах). В дальнейшем используем s_k' и переобозначаем его как s_t .

Практические соображения по оценке оптического потока

Конфигурации.

- *Быстрая (CPU):* Farneback, предварительное уменьшение до 540p, шаг по кадрам 2.
- *Точная (лёгкая ML, GPU):* RAFT-Lite/SpyNet, плиточная обработка для 720p/1080p.

Нормировка. Пороговые квантили p_{05} и p_{95} оцениваются по первым $F = 60$ сегментам (холодный старт), далее обновляются экспоненциальным сглаживанием с коэффициентом $\eta = 0.1$.

Сцены со склейками. Для обнаруженных монтажных склеек вклад пар (f_i, f_{i+1}) в $\text{Flow}(\cdot)$ обнуляется в окне ± 1 кадр вокруг разрыва.

Примечание по вычислениям. Подсчёт $\text{Flow}(\cdot)$ выполняется оффлайн/на этапе кодирования; на клиенте используются только нормированные значения $s_t \in [0,1]$.

4. Аналитические свойства

4.1 Границы выбора действий

Для любых текущих условий x_t и двух возможных действий $a_i, a_j \in A$ граница принятия решения между a_i и a_j определяется условием равенства их стоимостей:

$$\alpha(D_{\text{vis}}(a_i | x_t) - D_{\text{vis}}(a_j | x_t)) + \beta(C_{\text{comp}}(a_i) - C_{\text{comp}}(a_j)) = 0.$$

Данное соотношение задаёт пороговое условие (гиперповерхность) в пространстве признаков x_t , разделяющее области оптимального применения действий a_i и a_j . Иными словами, если левая часть неравенства отрицательна, действие a_i обладает меньшей стоимостью и предпочтительно, а при положительной – выгоднее a_j .

Require: $x_t = (d_t, c_t, q_t, s_t, m_t)$	
1: $s^* \leftarrow 0.4, \quad s^{hi} \leftarrow 0.9$	
2: $q_{low} \leftarrow 0.2, \quad q_{hi} \leftarrow 0.6$	
3: $m^{\text{int}} \leftarrow 0.3$	▷ порог для ML-интерполяции, см. раздел 5
4: $m^{\text{up}} \leftarrow 0.2$	▷ порог для апскейлинга, см. раздел 5
5: function FEASIBLE(a)	
6: return $T(a; m_t) \leq \Delta t$	
7: if $d_t > 0$ then	▷ Есть потерянные сегменты
8: if $d_t \geq 2$ or $c_t = 0$ then	▷ Длинная серия или нет контекста
9: if $q_t > q_{hi}$ then	
10: $a_t \leftarrow$ Повторный запрос	▷ S_4
11: else	
12: $a_t \leftarrow$ Понижение битрейта	▷ S_5
13: if $m_t \geq m^{\text{up}}$ and FEASIBLE(Upscale) then	
14: $a_t \leftarrow$ Апскейлинг	▷ S_3 (дополнительное улучшение при наличии ресурса)
15: else	▷ Одиночная потеря, контекст есть
16: if $s_t \geq s^{hi}$ then	▷ Сцена слишком динамична – интерполяция неустойчива
17: if $q_t > q_{hi}$ then	
18: $a_t \leftarrow$ Повторный запрос	▷ S_4
19: else	
20: $a_t \leftarrow$ Понижение битрейта	▷ S_5
21: if $m_t \geq m^{\text{up}}$ and FEASIBLE(Upscale) then	
22: $a_t \leftarrow$ Апскейлинг	▷ S_3
23: else if $s_t \geq s^*$ then	▷ Средняя динамика – возможна ML-интерполяция
24: if $m_t \geq m^{\text{int}}$ and FEASIBLE(ML-Interp) then	
25: $a_t \leftarrow$ ML-интерполяция	▷ S_2
26: else	
27: $a_t \leftarrow$ Простая интерполяция	▷ S_1 (фолбек из-за вычислительных ограничений)
28: else	▷ Статичная сцена
29: $a_t \leftarrow$ Простая интерполяция	▷ S_1
30: else	▷ Потерь нет
31: if $q_t \leq q_{low}$ then	
32: $a_t \leftarrow$ Понижение битрейта	▷ S_5
33: if $m_t \geq m^{\text{up}}$ and FEASIBLE(Upscale) then	
34: $a_t \leftarrow$ Апскейлинг	▷ S_3
35: else	
36: $a_t \leftarrow$ Ничего не делать	▷ S_0 ; нормальное воспроизведение

Рис. 2. Алгоритм адаптивного восстановления видеофрагмента с учётом вычислительных ресурсов.

Fig. 2. An algorithm for adaptive restoration of a video fragment taking into account computing resources.

В качестве примера рассмотрим выбор между простой интерполяцией (S_1) и ML-интерполяцией (S_2) для одиночного потерянного фрагмента ($d_t = 1, c_t = 1$). Порог по метрике динамики сцены s_t в данном случае определяется решением уравнения $\alpha[D_{\text{vis}}(S_1) - D_{\text{vis}}(S_2)] = \beta[C_{\text{comp}}(S_2) - C_{\text{comp}}(S_1)]$. Если рост D_{vis} при переходе от статичной сцены к подвижной для простого метода S_1 значителен, а для метода S_2 – относительно меньше, то существует критическое значение s^* , при превышении которого более высокая вычислительная цена метода S_2 окупается снижением визуальной ошибки. При $s_t < s^*$ простая интерполяция обеспечивает минимальную стоимость, тогда как при $s_t > s^*$

предпочтение смещается в пользу ML-интерполяции. В разработанном алгоритме данная граница установлена примерно на уровне $s_t \approx 0.4$, что отражено в правилах (см. табл. 1).

4.2 Обоснование порогов динамики сцены

Одним из ключевых параметров алгоритма является порог по динамике сцены s_t . Его выбор можно обосновать анализом баланса между визуальными потерями и вычислительными затратами. При низких значениях s_t (слабо меняющаяся сцена) простая интерполяция даёт приемлемое качество восстановления при минимальных затратах, и разница в качестве между простым и ML-методом не оправдывает высокую стоимость последнего. Однако по мере роста динамики сцены вклад визуальной ошибки от простой интерполяции стремительно увеличивается: после некоторого порога артефакты начинают доминировать в полной стоимости $C(a_t | x_t)$, и более сложный метод S_2 становится выгоднее. Таким образом, пороговое значение s_t устанавливается так, чтобы при s_t выше него метод S_2 имел меньшую ожидаемую стоимость, чем S_1 , то есть доминировал в области высокодинамичных сцен. Оптимальное значение этого порога зависит от характера видеоматериала и коэффициентов α , β . В наших правилах на основе анализа типичных сценариев выбрано значение 0.4: для большинства видов контента оно разделяет умеренно динамичные сцены (где простая интерполяция еще эффективна) от высокодинамичных (требующих более продвинутых методов). Это значение рекомендуется как отправная точка и подлежит уточнению для конкретных областей применения (например, спортивные трансляции).

Аналогично выбирается порог по длине серии потерь M . Предлагаемая стратегия устанавливает $M = 2$: если потеряно более двух сегментов подряд, дальнейшая интерполяция становится ненадёжной по качеству, и алгоритм переключается на повторный запрос. Это значение основано на доминировании по стоимости: восстановление длинной серии пропусков методом интерполяции приводит к накоплению ошибки, превышающей выигрыш во времени по сравнению с дозагрузкой. Ограничение $M = 2$ также оправдано статистически: при относительно низком уровне потерь вероятность столкнуться с серией более чем из двух пропусков мала, поэтому выигрыш от более сложного анализа этих случаев невелик.

В целом, алгоритм демонстрирует устойчивость к вариациям указанных пороговых параметров: их умеренное изменение не приводит к кардинальному изменению результатов, позволяя настраивать систему под конкретные условия.

5. Аналитическая оценка вычислительных затрат

Для практической реализации важно, чтобы выбранные действия могли выполняться в условиях ограниченного времени (реального времени воспроизведения). Оценим временную сложность различных методов восстановления.

Для каждого действия a_t обозначим через κ_{a_t} количество вычислительных операций (FLOPs), требуемое для обработки единицы пиксельных данных. Тогда для сегмента с разрешением $W \times H$ оценка числа операций составит $\kappa_{a_t} \cdot W \cdot H$. Время выполнения данного действия можно приблизительно выразить как:

$$T(a_t) = \frac{\kappa_{a_t} \cdot W \cdot H}{R_{\text{proc}}},$$

где R_{proc} – производительность устройства (например, в FLOPS, операций в секунду). Если алгоритму требуется обработать несколько кадров (как при интерполяции видео), то итоговое время масштабируется пропорционально числу кадров.

Для соблюдения требования реального времени значение $T(a_t)$ не должно превышать доступный бюджет времени для данного фрагмента (например, длительность сегмента или допустимую задержку буфера). Иными словами, клиентский плеер не может выбирать метод a_t , если $T(a_t) > \Delta t$, где Δt – максимально допустимая задержка воспроизведения. В таких

случаях необходимо применить более простой и быстрый метод восстановления. Например, если высокоточная ML-интерполяция для видео 720p требует порядка 0.5 с на кадр, а на воспроизведение одного кадра отведено лишь 0.04 с (25 кадр/с), то данный метод не может быть использован без остановки воспроизведения – клиенту следует заменить его более быстрым способом (например, простой интерполяцией или пропуском кадра).

Величина m_t вычисляется динамически на стороне клиента с использованием локальных измерений времени выполнения. Реализация может включать измерение времени выполнения эталонных операций (например, свёртки и операции повышения разрешения) с последующим сопоставлением результатов с референсной таблицей производительности. Подобная стратегия применяется в мобильных программных системах вывода нейросетевых моделей (Android NNAPI, Apple CoreML, TensorRT-Mobile), что обеспечивает стандартизируемый и переносимый механизм оценки производительности устройства в реальном времени.

Табл. 2. Фрагмент таблицы принятия решений автомата.

Table 2. A fragment of the decision-making table of the machine.

Класс действия a	Порядок κ_a (оп/px)
Простая интерполяция (удержание кадра / смешивание кадров)	10^0 – 10^1
Повышение разрешения (бикубическая интерполяция)	10^1 – 10^2
Нейросетевое повышение разрешения (облегчённое сверхразрешение)	10^3 – 10^4
Нейросетевая межкадровая интерполяция	10^4 – 10^5

Числовая иллюстрация. Рассмотрим 720p (1280×720), 25 fps. Пусть бюджет на шаг t равен C_t операций, а оценка времени $T(a_t)$ аппроксимируется как

$$T(a_t) \approx \frac{\kappa_{a_t} W H}{R_t},$$

где $W \times H$ – размер кадра, R_t – эффективная скорость вычислений (оп/с) у клиента. При $R_t = 1.5 \times 10^{11}$ оп/с (средний мобильный GPU), для $\kappa_a = 10^4$ получаем

$$T(a) \approx \frac{10^4 \cdot 1280 \cdot 720}{1.5 \times 10^{11}} \approx 0.00006 \text{ с (на пиксельный проход)}.$$

С учётом константных накладных расходов T_0 и количества проходов n_{pass} итог оценивается как $T_{\text{tot}} \approx n_{\text{pass}} \cdot T(a) + T_0$. Это позволяет вывести минимальный запас времени m_{min} , который в дальнейшем отождествляется с минимально допустимым значением m_a^* .

Пороговые константы $m_{\text{int}}, m_{\text{up}}$. Пусть время выполнения метода a аппроксимируется как

$$T(a; m_t) = \frac{\kappa_a (W \times H)}{\gamma(m_t) R_{\text{max}}} + \tau_a,$$

где κ_a – удельная вычислительная сложность (FLOPs/px), $W \times H$ – целевое разрешение, R_{max} – пиковая производительность устройства (px/s или FLOPs/s), $\gamma(m_t) \in (0, 1]$ – доля доступного ресурса, τ_a – фиксированная накладная (инициализация, копирования, синхронизация). Требование реального времени $T(a; m_t) \leq \Delta t$ эквивалентно

$$\gamma(m_t) \geq \frac{\kappa_a (W \times H)}{R_{\text{max}} (\Delta t - \tau_a)} \triangleq \Gamma_a.$$

Если $\gamma(m) = m$ (линейная шкала ресурса), минимально допустимый ресурс для метода a равен

$$m_a^* = \text{clip}(\Gamma_a, 0, 1).$$

Тогда пороги задаём как

$$m_{\text{int}} = m_{a=(\text{интерполяция})}^*, \quad m_{\text{up}} = m_{a=(\text{повышение разрешения})}^*.$$

Для учёта различий в производительности устройств вводим запас $\rho \geq 1$ (напр., $\rho = 1.15$) и используем $\tilde{m}_a^* = \text{clip}(\rho m_a^*, 0, 1)$. На практике $R_{\max}$ и τ_a калибруются при инициализации системы на основе локальной экспериментальной оценки производительности; κ_a оценивается на клиенте как $\tilde{\kappa}_a = \rho \frac{\hat{t}_a(W_0, H_0) - \tau_a}{W_0 H_0}$.

6. Соответствие и тестовые векторы

6.1 Нормативные требования и формат тега

В данном разделе формулируются требования к реализации предлагаемого подхода. Ключевые слова **MUST**, **SHOULD** и **MAY** используются в соответствии с документом RFC 2119 [15] для однозначного разграничения обязательных, рекомендуемых и допустимых условий реализации предлагаемого алгоритма.

- Плейлист HLS должен (**MUST**) включать тег #EXT-X-SCENE-DYN с указанием значения динамики сцены для каждого сегмента видео.
- Значение тега #EXT-X-SCENE-DYN должно (**MUST**) быть числом с плавающей точкой в диапазоне 0.0–1.0 (включительно), представляющим уровень динамики соответствующего сегмента.
- Если значение выходит за допустимый диапазон или формат тега нарушен, клиенту следует (**SHOULD**) игнорировать этот тег и продолжать воспроизведение, используя значение динамики сцены по умолчанию или считая сцену статичной.
- Если тег #EXT-X-SCENE-DYN присутствует, клиентский плеер должен (**MUST**) использовать его значение при выборе метода восстановления сегмента в соответствии с описанным алгоритмом.
- Если тег #EXT-X-SCENE-DYN отсутствует (например, в плейлисте старого формата), клиент может (**MAY**) принять для данного сегмента значение s_t по умолчанию либо применить консервативную стратегию восстановления.
- Клиенту следует (**SHOULD**) реализовать все перечисленные типы действий (S_1 – S_5) или их эквиваленты, чтобы обеспечить покрытие всех сценариев потерь. Отсутствие какого-либо метода может снизить качество восстановления при соответствующих условиях.
- Генерирующему контент приложению или серверу (**SHOULD**) вычислять метрику динамики s_k для каждого сегмента при подготовке потока и включать тег #EXT-X-SCENE-DYN в плейлист, повышая точность адаптивного восстановления.
- Выбранные клиентом методы восстановления должны (**MUST**) удовлетворять ограничению реального времени: ни одно действие не должно вызывать задержку, превышающую длительность сегмента (см. раздел "Аналитическая оценка вычислительных затрат"). Если ожидается нарушение этого условия, клиент должен (**MUST**) перейти на менее ресурсоёмкий метод восстановления.
- Клиент не должен (**MUST NOT**) выбирать метод восстановления a_t , если нарушается ограничение реального времени $T(a_t; m_t) \leq \Delta t$ (см. раздел 5).
- Применение ML-интерполяции (S_2) должно (**MUST**) происходить только при $m_t \geq m^{\min}(\text{ML-Interp})$; иначе следует (**SHOULD**) использовать простую интерполяцию (S_1).
- Применение метода нейросетевого повышения разрешения (S_3) должно (**MUST**) происходить только при $m_t \geq m^{\min}(\text{Up})$; иначе следует (**SHOULD**) отображать видеопоток пониженного качества без постобработки.

- При $s_t \rightarrow 1$ (очень высокая динамика) не следует (**SHOULD NOT**) применять интерполяция; клиенту следует (**SHOULD**) либо выполнить повторный запрос (S_4) при $q_t > q_{hi}$, либо снизить качество видеопотока (S_5) и, при возможности, применить нейросетевое повышение разрешения (S_3).

Ниже приведено формальное определение нового тега на языке ABNF:

$SceneDynTag = \text{"\#EXT-X-SCENE-DYN:"} SceneValue$
 $SceneValue = \text{"0" / "0." 1*DIGIT / "1" / "1.0"}$

Согласно стандарту HLS, плееры, не распознающие пользовательский тег #EXT-X-SCENE-DYN, проигнорируют его, сохраняя совместимость. Таким образом, внедрение нового тега не нарушает работу существующих клиентов, не поддерживающих данное расширение.

6.2 Тестовые векторы и примеры

Для проверки корректности работы алгоритма приведены примеры входных условий (значения d_t, c_t, s_t, q_t) и соответствующих действий a_t . Табл. 3 иллюстрирует набор тестовых векторов для различных ситуаций.

Табл. 3. Примеры тестовых векторов и ожидаемые действия алгоритма.
Table 3. Examples of test vectors and expected algorithm actions.

d_t	c_t	s_t	q_t	Выбранное действие a_t
1	1	0.2	0.5	Простая интерполяция (S_1)
1	1	0.4	0.5	ML-интерполяция (S_2)
1	1	0.1	0.4	Простая интерполяция (S_1)
2	1	0.3	0.8	Повторный запрос (S_4)
1	0	0.5	0.7	Повторный запрос (S_4)
0	1	0.3	0.1	Понижение качества видеопотока (S_5)

Далее представлены две примерные последовательности событий, иллюстрирующие работу автомата состояний в разных сценариях потерь:

1. **Короткая одиночная потеря.** Воспроизведение начинается в нормальном режиме (S_0). Когда теряется один сегмент ($d_t = 1$) на статичной сцене (низкое s_t) при хорошей пропускной способности (q_t среднее, $c_t = 1$), алгоритм выбирает простую интерполяцию (переход в S_1). Восстановленный фрагмент сразу отображается, и воспроизведение продолжается без задержки (возврат в S_0).
2. **Серия из нескольких потерь.** Начальное состояние воспроизведения S_0 . Предположим, подряд теряются три сегмента на динамичной сцене при достаточно низком качестве соединения (q_t низкое). Все потерянные сегменты ($d_t = 3, c_t = 0$, высокое s_t) будут получены путем перезапроса с понижением качества потока (S_5 ; при достатке вычислительных ресурсов переход в S_3), поскольку интерполяция серии стала бы ненадёжной по качеству. После получения данных воспроизведение возвращается в нормальное состояние (S_0).

6.3 Совместимость со стандартом HLS

Механизм основан на пользовательском теге EXT-X-SCENE-DYN. Согласно RFC 8216, клиенты обязаны игнорировать неизвестные теги, поэтому обратная совместимость сохраняется. Нововведение не вмешивается в структуру сегментов, не влияет на ключевую безопасность и совместимо с существующими CDN и кэшами. Предложение аналогично используемым в индустрии расширениям (например, EXT-X-SESSION-DATA, EXT-X-DATERANGE).

7. Валидация модели и симуляционное исследование

В данном разделе представлено симуляционное исследование предложенной стратегии адаптивного восстановления. Основная цель симуляции – проверить внутреннюю согласованность логики конечного автомата и проанализировать чувствительность алгоритма к вариациям пороговых параметров. Имитация работы алгоритма проводилась с использованием симулятора клиентского видеоплеера.

Были рассмотрены 50 видеоклипов, охватывающих различные типы сцен (от статичных до высокодинамичных, включая клипы с монтажными склейками), чтобы учесть классы контента, наиболее чувствительные к выбору стратегии восстановления. Смоделировано семь профилей сетевых условий, различающихся уровнем пропускной способности, вероятностью потерь и характером деградации канала (см. табл. 4). Пороговые значения S^* , S_{hi} , Q_{low} , Q_{hi} , M использовались согласно табл. 1.

Табл. 4. Параметры используемых сетевых сценариев.

Table 4. Parameters of used network scenarios.

Сценарий	Вероятность потерь	q_{min}	q_{max}
good	0.05	0.60	1.00
medium	0.15	0.30	0.90
bad	0.30	0.10	0.60
very_bad	0.45	0.05	0.25
burst_heavy	0.20	0.20	0.70
catastrophic	0.35	0.00	1.00
motion_sensitive	0.20	0.10	0.80

Эффективность оценивалась по двум показателям: среднему значению SSIM для восстановленных сегментов и интегральной метрике QoE (Quality of Experience), учитывающей штраф за частые переключения состояний автомата.

Для анализа поведения автомата были рассмотрены следующие стратегии:

1. предлагаемая конечная модель (*жестко пороговая стратегия*);
2. *freeze baseline*: заморозка последнего кадра;
3. *downscale baseline*: понижение качества видеопотока для сокращения времени передачи данных при деградации сети;
4. *refetch baseline*: повторная загрузка недостающего сегмента.

Такой набор покрывает как простейшие эвристики, так и методы, ориентированные исключительно на сетевой контекст, позволяя выявить вклад пороговых правил и признака динамики сцены.

7.1 Результаты симуляции

На рис. 3 приведены средние значения SSIM для всех методов во всех сетевых сценариях. Предлагаемая стратегия демонстрирует качество, сопоставимое с результатами бэйзлайнов. В отдельных профилях наблюдается небольшой выигрыш или снижение качества, однако величина отличий остается в пределах стохастической дисперсии, обусловленной вариативностью модели восстановления. Основная цель данной симуляции состояла не в том, чтобы превзойти существующие методы по качеству, а в проверке корректности работы правил принятия решений в различных условиях.

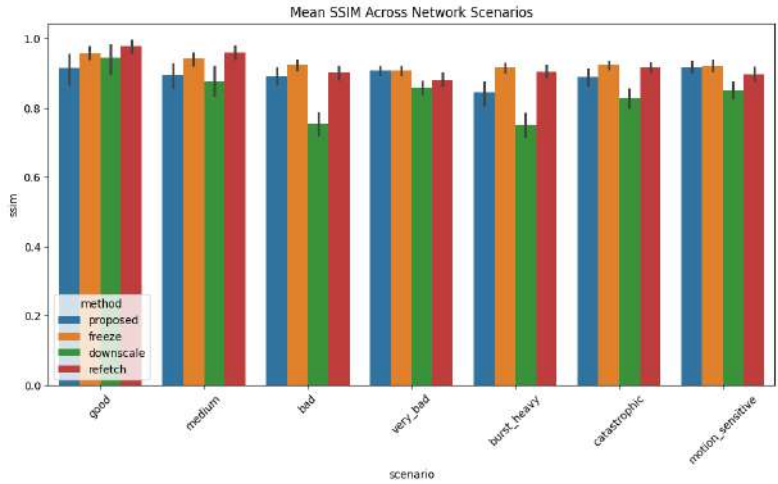


Рис. 3. Средний SSIM для различных методов восстановления во всех сетевых сценариях.

Fig. 3. Average SSIM for different recovery methods across all network scenarios.

Полученные результаты подтверждают, что предложенная стратегия работает согласованно при различных сетевых условиях. Хотя значения SSIM сопоставимы с простыми базовыми подходами, основной целью данной симуляции являлась проверка того, что правила решения дают корректные действия для разных сочетаний входных условий, а не демонстрация абсолютного выигрыша в качестве.

Аналогичный вывод можно сделать по интегральной метрике QoE (см. рис. 4): предложенная модель поддерживает стабильное качество воспроизведения без чрезмерных штрафов за переключения, что подтверждает устойчивость логики переходов автомата.

Эти результаты демонстрируют, что разработанная модель функционирует согласно замыслу и сохраняет предсказуемое поведение при изменении условий сети.

7.2 Анализ чувствительности пороговых параметров

Особое внимание было уделено анализу того, как выбор порогов влияет на поведение алгоритма. Рассматривались изменения следующих параметров:

- q_{hi} – верхний порог пропускной способности, определяющий переход к повторному запросу;
- q_{low} – нижний порог пропускной способности, ведущий к снижению качества видеопотока;
- s_{hi} – верхний порог динамики сцены, выше которого интерполяция становится неустойчивой;
- s^* – средний порог динамики сцены, разделяющий области применения простой и ML-интерполяции.

Чувствительность к q_{hi} . Как показано на рис. 5, изменение q_{hi} в диапазоне 0.6–0.8 приводит к умеренным колебаниям SSIM, подтверждая, что алгоритм адекватно реагирует на этот параметр без появления нестабильного поведения.

Чувствительность к q_{low} . Рис. 6 демонстрирует ожидаемую чувствительность алгоритма к этому параметру: более низкие значения q_{low} откладывают переход к снижению качества видеопотока, тогда как более высокие иницируют его раньше, что соответственно сказывается на показателях качества.

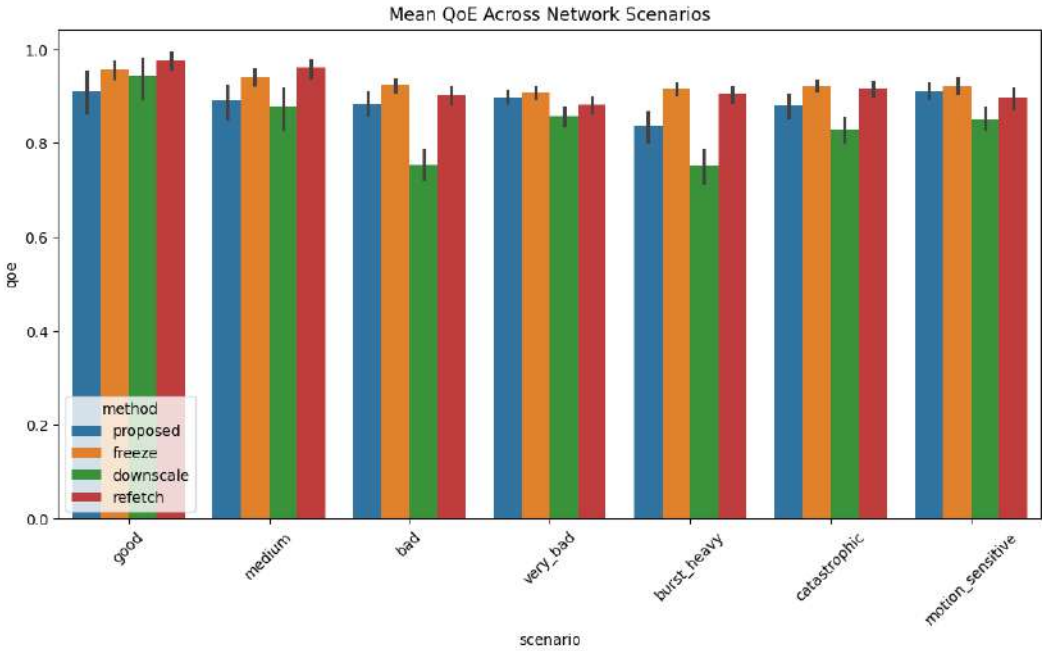


Рис. 4. Средняя QoE (с учетом штрафа за переключения).
Fig. 4. Average QoE (including switching penalties).

Чувствительность к s_{hi} . Варьирование верхнего порога динамики сцены приводит к предсказуемым изменениям в поведении алгоритма. Как иллюстрирует рис. 7, более высокие значения s_{hi} позволяют применять интерполяцию при более динамичных сценах, тогда как более низкие значения ожидаемо ограничивают ее использование, что соответствует задумке модели.

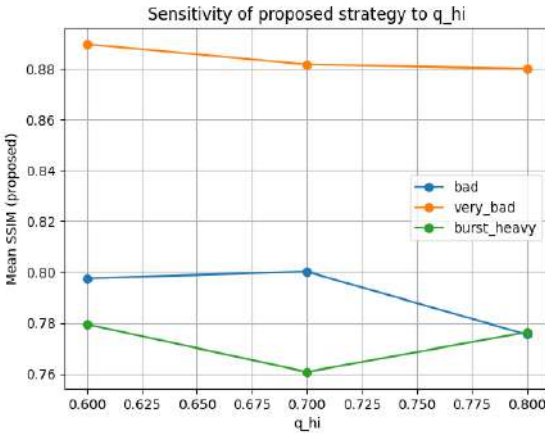


Рис. 5. Чувствительность стратегии к изменению порога q_{hi} .
Fig. 5. Sensitivity of the strategy to changes in the threshold q_{hi} .

Чувствительность к s^* . Средний порог динамики сцены s^* фактически контролирует границу между применением простой и ML-интерполяции. Алгоритм демонстрирует корректные переключения стратегии по мере изменения этого параметра (см. рис. 8), что означает, что переходы между S_1 и S_2 происходят в ожидаемых случаях при различных значениях s^* .

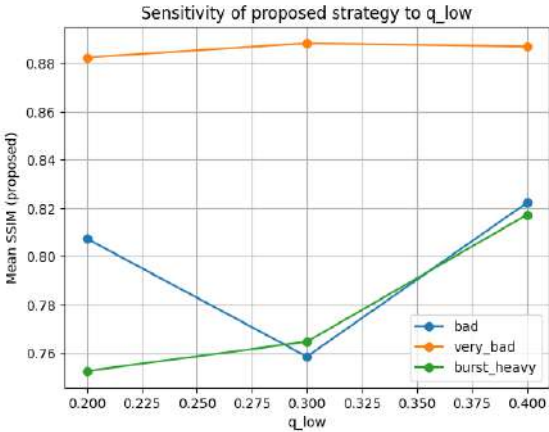


Рис. 6. Чувствительность стратегии к изменению порога q_{low} .
Fig. 6. Sensitivity of the strategy to changes in the threshold q_{low} .

7.3 Ограничения симуляции

Приведенное симуляционное исследование основано на синтетических данных и упрощенных моделях визуального и вычислительного затрат. Его основная цель – проверить внутреннюю согласованность работы конечного автомата и проанализировать чувствительность алгоритма к изменению порогов. Модели D_{vis} и C_{comp} являются аналитическими аппроксимациями и могут не отражать всех нюансов реального видеоконтента и аппаратного обеспечения. Однако они достаточны для проверки логики принятия решений и чувствительности порогов. Полноценная оценка на реальных датасетах и устройствах остается предметом будущей работы.

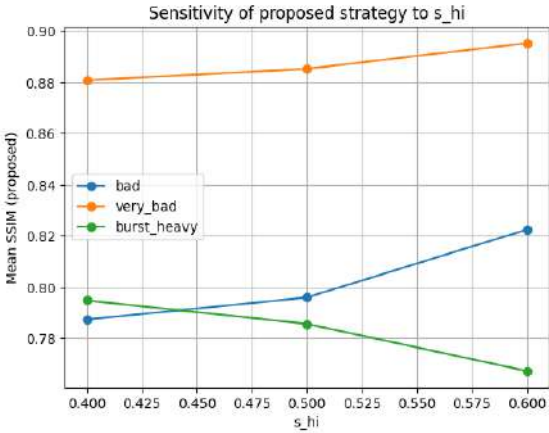


Рис. 7. Чувствительность стратегии к изменению порога s_{hi} .
Fig. 7. Sensitivity of the strategy to changes in the threshold s_{hi} .

7.4 Результаты симуляции и выводы

Симуляционное исследование подтверждает, что предложенная модель работает согласованно и адекватно реагирует на изменения параметров. Анализ чувствительности показывает, что каждый пороговый параметр влияет на поведение алгоритма ожидаемым образом, что служит валидацией внутренней логики конечного автомата. Эти результаты

внушают уверенность в правильности конструкции модели и служат основой для дальнейшей реализации и тестирования в реальных системах.

Детально описанные модели симуляции, представленные в этом разделе, обеспечивают воспроизводимость экспериментов и дают прозрачное представление о процессе принятия решений алгоритма в контролируемых условиях.

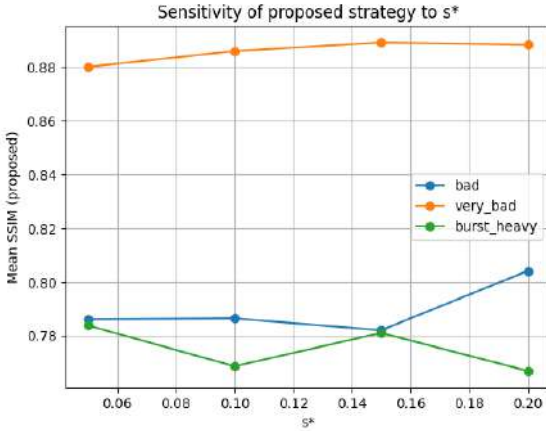


Рис. 8. Чувствительность стратегии к изменению порога s^* .
Fig. 8. Sensitivity of the strategy to changes in the threshold s^* .

8. Заключение

Представлен формальный подход к адаптивному восстановлению видеофрагментов с расширением пространства признаков за счет учета динамики сцены. Предложенная модель интегрирует анализ динамики сцены в процесс принятия решений через архитектуру конечного автомата и обратно совместимое расширение HLS-плейлиста.

Основные результаты и вклад работы:

- формальная модель принятия решений на основе минимизации функции стоимости с явным учетом визуального качества и вычислительных ограничений;
- архитектура конечного автомата, объединяющая сетевые условия, вычислительные возможности устройства и характеристики контента;
- обратно совместимое расширение плейлиста (новый тег), передающее признак динамики сцены и позволяющее учитывать его при выборе метода восстановления.

Проведенная симуляционная оценка не является полноценной экспериментальной валидацией, однако позволяет проанализировать поведение автомата при различных уровнях динамики, качества соединения и вычислительных ограничениях. Результаты показывают, что в текущей реализации признак динамики сцены не приводит к устойчивому повышению SSIM относительно простых стратегий восстановления. Это подтверждает необходимость дальнейшего уточнения моделей визуальной ошибки и вычислительной стоимости, а также более точной параметризации порогов автомата.

Несмотря на ограниченность оценки, предложенная формализация демонстрирует применимость модели принятия решений на основе расширенного набора признаков и дает основу для дальнейшей эмпирической проверки. Перспективные направления развития включают:

- получение реалистичных измерений D_{vis} и C_{comp} на реальных видеодатасетах;
- уточнение модели динамики сцены и исследование альтернативных метрик;

- интеграцию стратегии восстановления с алгоритмами адаптивного управления скоростью передачи данных (ABR);
- анализ влияния улучшений на восприятие пользователями (QoE).

Таким образом, проведенная работа формирует теоретическую основу для построения клиентских систем адаптивного восстановления и определяет ключевые направления их дальнейшей разработки.

Список литературы / References

- [1]. Peroni L., Gorinsky S. An End-to-End Pipeline Perspective on Video Streaming in Best-Effort Networks: A Survey and Tutorial. *ACM Computing Surveys*, vol. 57, no. 12, article 322, 2025, pp. 1-47. DOI: 10.1145/3742472.
- [2]. Qi Y., Dai M. The Effect of Frame Freezing and Frame Skipping on Video Quality. In *Proc. Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, Jakarta, Indonesia, 2006, pp. 423-426. Available at <https://ieeexplore.ieee.org/document/4041752>, accessed 16.06.2025.
- [3]. Spiteri K., Sitaraman R. K., Sparacio D. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 15, no. 2s, 2019, article 15, pp. 1-29. DOI: 10.1145/3336497.
- [4]. Mao H., Netravali R., Alizadeh M., et al. Neural Adaptive Video Streaming with Pensieve. In *Proc. ACM SIGCOMM*, 2017. DOI: 10.1145/3098822.3098843.
- [5]. Sani Y., et al. SMASH: A Supervised Machine Learning Approach to Adaptive Video Streaming over HTTP. In *Proc. QoMEX*, 2020, pp. 1-6. DOI: 10.1109/QoMEX48832.2020.9123139. Available at <https://ieeexplore.ieee.org/document/9123139>, accessed 22.06.2025.
- [6]. Petrangeli S., De Cock J., Van Wallendaal G., et al. A Machine Learning-based Framework for Preventing Video Freezes in HTTP Adaptive Streaming. *Journal of Network and Computer Applications*, vol. 94, 2017, pp. 78-92. DOI: 10.1016/j.jnca.2017.07.009.
- [7]. Wang Y., Zhu Q.-F. Error Control and Concealment for Video Communication: A Review. *Proceedings of the IEEE*, vol. 86, no. 5, 1998, pp. 974-997. Available at <https://ieeexplore.ieee.org/document/664283>, accessed 24.07.2025.
- [8]. Bao W., Lai W.-S., Ma C., et al. Depth-Aware Video Frame Interpolation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3703-3712. DOI: 10.48550/arXiv.1904.00830.
- [9]. Tran Q. N., Yang S.-H. Efficient Video Frame Interpolation Using Generative Adversarial Networks. *Applied Sciences*, vol. 10, no. 18, 2020, article 6245. DOI: 10.3390/app10186245.
- [10]. Lu Y., Bajić I., et al. ABUV: Adaptive bitrate and upsampling for video streaming on mobile devices. *Computer Networks*, 2024, article 110994. DOI: 10.1016/j.comnet.2024.110994.
- [11]. Wang F., et al. Reparo: QoE-Aware Live Video Streaming in Low-Rate Networks by Intelligent Frame Recovery. Preprint (OpenReview), 2025. DOI: 10.1145/3581783.3613441.
- [12]. Ge L., Jiang D. Z., Bao W. 3RE-Net: Joint Loss-Recovery and Super-Resolution Neural Network for Real-Time Video. *Lecture Notes in Computer Science*, 2023, pp. 165-177. DOI: 10.1007/978-981-99-8388-9_14.
- [13]. He Z., et al. Real-Time Neural Video Recovery and Enhancement on Mobile Devices. *arXiv:2307.12152*, 2023. DOI: 10.48550/arXiv.2307.12152.
- [14]. Pantos R., May W. HTTP Live Streaming. RFC 8216, 2017. Available at <https://datatracker.ietf.org/doc/html/rfc8216>, accessed 10.08.2025.
- [15]. Bradner S. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119, 1997. Available at <https://datatracker.ietf.org/doc/html/rfc2119>, accessed 10.08.2025.

Информация об авторах / Information about authors

Никита Алексеевич ДУМКИН – преподаватель Национального исследовательского университета «Высшая школа экономики», департамента программной инженерии факультета компьютерных наук. Область профессиональных и научных интересов включает программную инженерию, потоковую передачу видео, методы машинного обучения, обработку мультимедийных данных и разработку высокопроизводительных клиентских систем.

Nikita Alekseevich DUMKIN – lecturer at the National Research University Higher School of Economics (HSE), Faculty of Computer Science, Department of Software Engineering. His academic and professional interests include software engineering, video streaming technologies, machine learning methods, multimedia data processing, and the development of high-performance client-side systems.

Дмитрий Владимирович АЛЕКСАНДРОВ – доктор технических наук, профессор факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики», департамента программной инженерии. Сфера научных интересов включает программную инженерию, распределённые и высоконагруженные вычислительные системы, архитектуру программного обеспечения, сетевые технологии и методы повышения надёжности и производительности программных систем.

Dmitry Vladimirovich ALEXANDROV – Dr. Sci. (Tech.), Professor at the Faculty of Computer Science, National Research University Higher School of Economics (HSE), Department of Software Engineering. His research interests include software engineering, distributed and high-load computing systems, software architecture, networking technologies, and methods for improving the reliability and performance of software systems.

Михаил Алексеевич ПРОЗОРСКИЙ – студент бакалавриата Национального исследовательского университета «Высшая школа экономики» по направлению «Программная инженерия». Область учебных и профессиональных интересов включает разработку программного обеспечения, основы программной инженерии и современные технологии создания информационных систем.

Mikhail Alekseevich PROZORSKIY an undergraduate student at the National Research University Higher School of Economics (HSE) majoring in Software Engineering. His academic and professional interests include software development, fundamentals of software engineering, and modern technologies for building information systems.

DOI: 10.15514/ISPRAS-2026-38(1)-18



Multi-Record Web Page Information Extraction from News Websites

¹ A.A. Kustenvov, ORCID: 0009-0008-3043-9424 <kustenvovaa@my.msu.ru>

¹² A.K. Yatskov, ORCID: 0000-0002-1312-1675 <yatskov@ispras.ru>

² M.I. Varlamov, ORCID: 0000-0002-1083-6210 <varlamov@ispras.ru>

¹ Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia.

² Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. In this paper, we focused on the problem of extracting information from web pages containing many records, a task of growing importance in the era of massive web data. Recently, the development of neural network methods has improved the quality of information extraction from web pages. Nevertheless, most of the research and datasets are aimed at studying detailed pages. This has left multi-record "list pages" relatively understudied, despite their widespread presence and practical significance. To address this gap, we created a large-scale, open-access dataset specifically designed for list pages. This is the first dataset for this task in the Russian language. Our dataset contains 13,120 web pages with news lists, significantly exceeding existing datasets in both scale and complexity. Our dataset contains attributes of various types, including optional and multi-valued, providing a realistic representation of real-world list pages. These features make our dataset a valuable resource for studying information extraction from pages containing many records. Furthermore, we proposed our own multi-stage information extraction methods. In this work, we explore and demonstrate several strategies for applying MarkupLM to the specific challenges of multi-record web pages. Our experiments validate the advantages of our methods. By releasing our dataset to the public (<https://github.com/ispras/news-page-dataset>), we aim to advance the field of information extraction from multi-record pages.

Keywords: data collection; data extraction; dataset; multirecord extraction; neural network.

For citation: Kustenvov A.A., Yatskov A.K., Varlamov M.I. Multi-Record Web Page Information Extraction from News Websites. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 1, 2026, pp. 275-288. DOI: 10.15514/ISPRAS-2026-38(1)-18.

Извлечение информации из новостных веб-страниц, содержащих множество записей

¹ А.А. Кустенков, ORCID: 0009-0008-3043-9424 <kustenkova@my.msu.ru>

^{1,2} А.К. Яцков, ORCID: 0000-0002-1312-1675 <yatskov@ispras.ru>

² М.И. Варламов, ORCID: 0000-0002-1083-6210 <varlamov@ispras.ru>

¹ Московский государственный университет имени М.В. Ломоносова,
Россия, 119991, Москва, Ленинские горы, д. 1.

² Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. В данной работе мы сосредоточились на задаче извлечения информации с веб-страниц, содержащих множество записей, задаче, приобретающей всё большую актуальность в эпоху больших объёмов веб-данных. В последнее время развитие методов машинного обучения повысило качество извлечения информации с веб-страниц. Тем не менее, большинство исследований и наборов данных направлены на изучение страниц, содержащих одну запись. Это привело к тому, что страницы со списками (они же страницы, содержащие множество записей) изучены недостаточно, несмотря на их широкое распространение и практическую значимость. Чтобы восполнить этот пробел, мы создали новый большой набор данных с открытым доступом, специально разработанный для страниц со списками. Это первый набор данных для этой задачи на русском языке. Наш набор данных содержит 13 120 веб-страниц со списками новостей, что значительно превосходит существующие наборы данных как по масштабу, так и по полноте. Наш набор данных содержит атрибуты различных типов, включая необязательные и многозначные, что обеспечивает реалистичное представление страниц со списками. Эти особенности делают наш набор данных ценным ресурсом для изучения извлечения информации со страниц, содержащих множество записей. Кроме того, мы предложили собственные методы многоэтапного извлечения информации. В этой работе мы исследуем и демонстрируем несколько стратегий применения MarkupLM к решению специфических задач извлечения информации из веб-страниц с несколькими записями. Наши эксперименты подтверждают преимущества наших методов. Публикуя наш набор данных в открытом доступе (<https://github.com/ispras/news-page-dataset>), мы стремимся к развитию области извлечения информации из страниц с несколькими записями.

Ключевые слова: сбор данных; извлечение данных; набор данных; извлечение нескольких записей; машинное обучение.

Для цитирования: Кустенков А.А., Яцков А.К., Варламов М.И. Извлечение информации из новостных веб-страниц, содержащих множество записей. Труды ИСП РАН, том 38, вып. 1, 2026 г., стр. 275–288 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(1)–18.

1. Introduction

In the digital age, a huge amount of information presented on the Internet often exists in complex multi-record formats, especially on news websites. A record is defined as a separate entity on a webpage, for example, for news resources it is one news item, that includes a title, publication date and possibly other characteristics. A multi-record page is a page containing several records. In the news domain, this might be a page of some category, which usually displays several news items. A page with only a single record (news item) is called detailed. Extracting data from multi-record pages poses a significant problem, especially given that traditional methods and datasets are mostly focused on single-record or detailed pages. This paper presents a novel approach to extracting information from multi-record web pages, especially those located on Russian-language news websites. This article aims to improve the accuracy and efficiency of data extraction from such intricate web environments by building a large dataset of over 13,000 pages and applying neural methods such as the MarkupLM model. Despite the numerous approaches to solving the problem of information extraction, this article will focus on methods that do not require a visual representation

of the page (including generating a representation from the HTML code) and instead rely only on HTML code.

2. Related work

There are several datasets for the task of extracting attributes from web pages with many records (see, for example, Table 1). They all have their own unique features, since each was collected to test a specific method of solving this problem. However, there is a certain set of characteristics by which we can compare them:

- Number of pages for each subject area
- Number of sites for each subject area
- Number of records for each subject area
- Number of pages in total
- Year of dataset release

One of the first datasets was announced in the article Simultaneous Record Detection and Attribute Labeling in Web Data Extraction [1] in 2006 and was named LDST – ListDataSeT. In this dataset, the authors included 771 pages which contained lists of records.

The first large dataset aimed at studying the problem of extracting information from pages with many records was proposed in the paper AMBER: Automatic Supervision for Multi-Attribute Extraction [2] in 2012. The authors of the article focused on 150 real sites that contained pages with many records. The resulting dataset contained more than 2,000 pages with a total number of records more than 20,000.

In 2020, a team of researchers from Amazon published the article PLAtE: A Large-scale Dataset for List Page Web Extraction [3] in which they presented a new dataset PLATE – Pages of Lists Attribute Extraction which contains pages of online stores. The authors focused on preparing high-quality data, they conducted a multi-stage preparation of data, which included:

- 1) Filtration of pages that did not contain lists of records.
- 2) Among other pages, the most popular websites were selected based on data published in the Tranco List.
- 3) Also, preference was given to sites that contained as many pages with lists as possible.
- 4) Pages with obscene content were filtered.

Thus, the dataset was based on 43 sites and more than 6,600 pages, which together contains more than 52,000 records. Based on this, main feature of this dataset is the high quality of the data.

One of the problems related to web page extraction is web page segmentation. This task involves finding the boundaries of each record on the page. There are several approaches to the segmentation problem. Some of them assume the presence of visual information. For example, such algorithms are discussed in the article [4]. However, in this work, we assume that visual information is unavailable and we will use other methods.

One of the first solutions of finding record boundaries on the page is the algorithm proposed in the article Web data extraction based on partial tree alignment [5]. The authors developed algorithm called MDR, this algorithm finds similar fragments on the page. It is assumed that "similar" fragments records have the same attributes. After finding these fragments, the process of alignment occurs, this process compares each node of the fragment to the node in the remaining fragments. By this algorithm we can present information in a structured form.

It is also worth mentioning the article Finding and Extracting Data Records from Web Pages [6], in which an algorithm for generating partition candidates and selecting the best one was proposed. In our article we consider using this algorithm from a practical point of view.

Table 1. Dataset comparison.

Dataset	Pages/Vert.	Sites/Vert.	Records/Vert	Total Pages	Year	Language
LDST	771	---	8600	771	2011	English
AMBER	200	150	2000	431	2012	English
PLATE	6694	43	52900	6694	2020	English
Our dataset	13120	278	257595	13120	2023	Russian

According to our research, there are information extraction methods designed to extract from detailed pages, which might be generalized for the task of extracting information from list pages. One of these methods is the MarkupLM, which was designed in 2022 by Microsoft [7]. MarkupLM is BERT-like model, which is used for processing markup-language-based documents such as HTML or XML. The main feature of this model is encoding text and location of node together. The authors propose to use this model as a tool for working with detailed pages. More detailed description of architecture of this model is given in the Fig. 1.

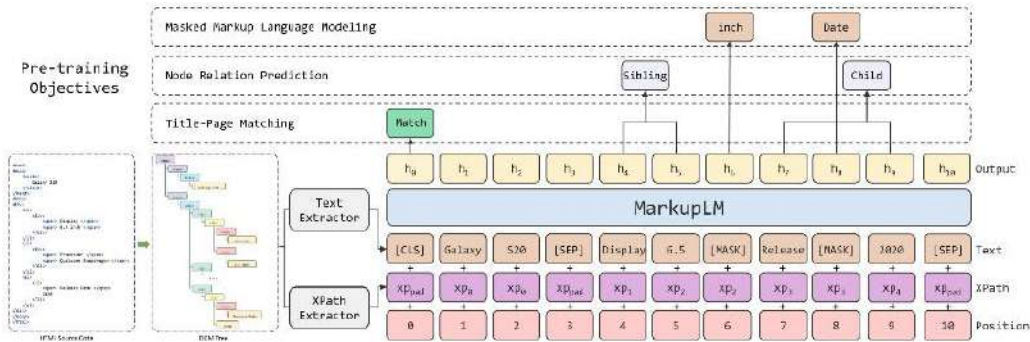


Fig. 1. MarkupLM architecture (This picture is taken from [7]).

3. Problem definition

Given a set W of n websites. Each website W_i , where $1 \leq i \leq n$, is represented by a certain number of pages m_i , a website may consist of just one page, i.e., $m_i = 1$. Each page W_i^j on website W_i , where j is the page number, with $1 \leq j \leq m_i$, is a multi-record page, meaning it contains k records where $2 \leq k$. We defined a record as an entity with a predefined set of characteristics, e.g. in the news domain these characteristics might include date, author, title, tag, etc. So, the set of record's characteristics is a vector $H = \{h_0, h_1, \dots, h_t\}$, where t the number of characteristics in a given record. It is important to note that a record can have multiple characteristics of the same type, such as multiple tags.

We formulate the task of extracting information from multi-record pages as identifying all vectors H within the given set W , with the following constraints:

- 1) The proposed methods must not rely on visual information based on page rendering.
- 2) The proposed methods must be applicable to all multi-record pages in the domain, even if the website was not included in the training dataset.
- 3) The proposed methods should generalize beyond the news domain to other fields.

The output of each method should be a structured representation of the records for each page, for example in JSON format.

4 Dataset construction

We decided to collect our own dataset for the task of extraction information from multi-record web-pages and make it publicly available. In this chapter we describe how the data was collected and annotated, also we provide the final dataset's main characteristics.

4.1 Data collecting

Our dataset contains news web pages collected from Russian-language media. News resources were selected according to the MediaMetrics [8] latest news quotations system, which provides rankings based on the popularity of news resources. Web pages were downloaded between 12/28/2023 and 04/28/2024. Pages were downloaded using a special Python3 script (based on the Scrapy library [9]), which was run daily through prepared sitemaps.

4.2 Sitemaps development

Sitemaps were prepared using the WebScraper [10] browser extension for Chrome. Using sitemaps in special web crawlers allows to download an html page, as well as an answer for this page. Thus, each site requires the development of a unique sitemap. On each page, the boundaries of each record and its attributes were annotated, if they existed. Only the following attributes were noted: *date*, *title*, *tag*, *short_title*, *author*, *time*. The annotation was done manually.

We chose this set of attributes because they are the most popular in the news websites domains.

For each site several categories were annotated, for example, politics, economics, and sports. In this way 312 sitemaps were prepared.

4.3 Dataset preparing

For further use of data in our dataset, it was necessary to preprocess the raw data. The following actions were carried out:

- 1) **Filtering duplicate pages.** Some downloaded pages contained many records scraped before on other pages. Such pages, with more than a quarter repeated records, were filtered.
- 2) **Cleaning HTML.** At this stage blocks, such as blocks of code in JavaScript (i.e. all nodes with the `<script>` tag), that do not affect the algorithm's performance were removed from the HTML code of the page.
- 3) **Translating HTML.** Since our dataset contains pages in Russian, it was necessary to translate them into English to be able to use pre-trained models.
- 4) **Division into training and test parts.** The distribution of attributes and domains of web pages was taken into account. Each domain was placed either to the training or to the test parts (see attributes split in the Fig. 2). The final distribution is shown in the Fig. 3 and Fig. 4. Ratio of parts after splitting was the following: 75% – training, 25% – testing.

4.4 Dataset Statistics

Since the maps were based on CSS selectors, there was a problem with downloading sites that dynamically change the names of the styles on the pages. In other words, when the website changed the name of the CSS style class, the selector specifying the class name stopped working correctly. Therefore, we were able to download only 278 sites.

Thus, a dataset that contained 13120 pages from 278 Internet media was prepared (distribution between pages and entries on them, shown in the Fig. 5). On each page, the corresponding attributes were annotated, and their frequency was presented in the Table 2. All pages presented in the table have UTF-8 encoding.

We researched extraction of 3 attributes from our dataset: title, author and tag. Because they are the most frequent in our dataset.

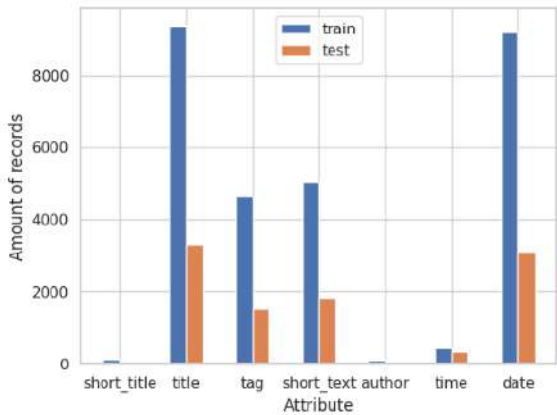


Fig. 2. Attribute distribution between train and test.

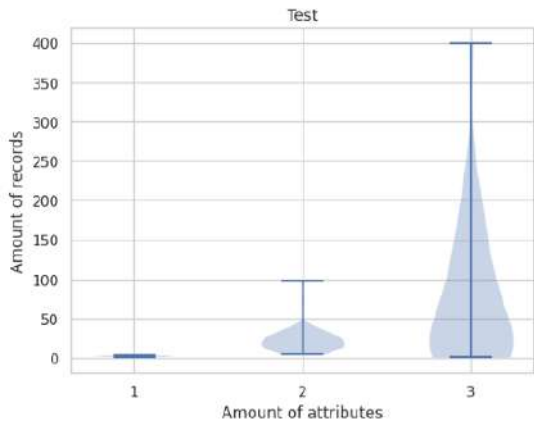


Fig. 3. Distribution of number of attributes at test part.

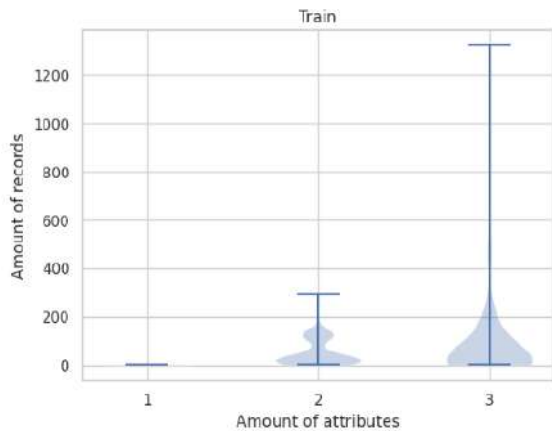


Fig. 4. Distribution of number of attributes at train part.

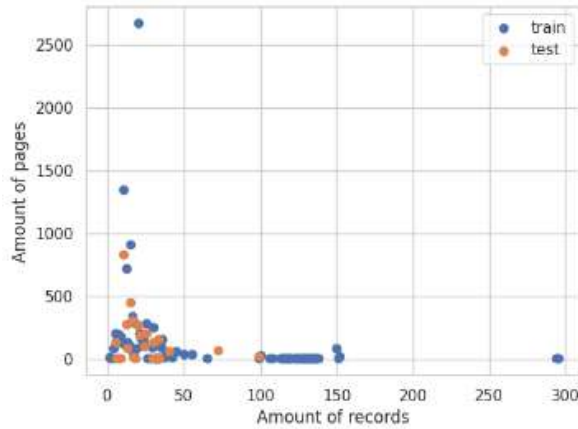


Fig. 5. Distribution between pages and number of records at page at train and test parts.

Table 2. Attribute frequency.

Name	Pages	Records	Sites
title	12679	247262	275
date	12296	241634	251
tag	6165	108400	140
short_text	6855	115983	138
time	730	15809	8
short_title	105	1289	4
author	87	957	1

5 Experimental evaluations

In this chapter we describe our methods of solving the problem and show the results on our dataset.

5.1 Parallel pipeline

The first architecture we tested was Parallel pipeline. Visual scheme of this architecture described in Fig. 6.

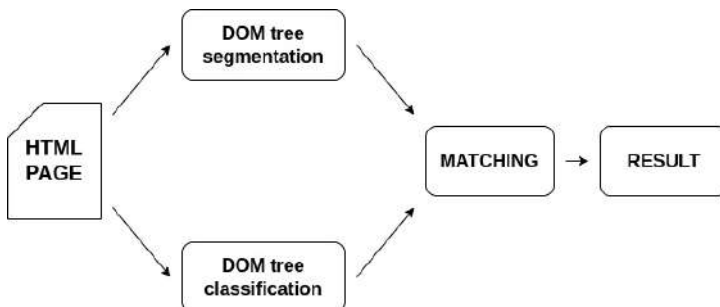


Fig. 6. Parallel pipeline scheme.

- **Segmentation.** At this stage, we find records boundaries that help us split html into records and find corresponding attributes of each record.
- **Classification.** At this stage, we give a corresponding label for each node on html. It is important that the segmentation stage and the classification stage are two independent stages. The results of neither are not shared with the other.
- **Matching of results.** At this stage, the final result of the method is formed. Based on the results of segmentation and classification, the records are matched with their attributes and provided in a structured view.

5.2 Sequential pipeline

We will also test sequential pipeline as described at Fig. 7. In the following we will compare the qualities of both architectures. Quality should vary due to different approaches to the classification stage.

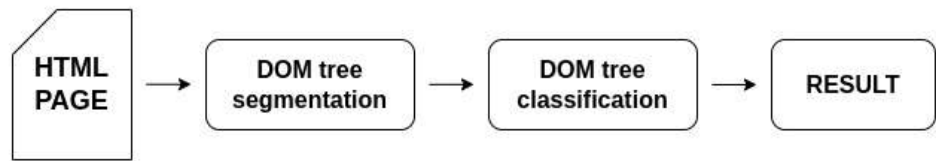


Fig. 7. Sequential pipeline scheme.

- **Segmentation.** In sequential architecture this stage is similar to parallel scheme.
- **Classification.** At this stage, the final result of the method is formed. Based on the segmentation results, the stage of searching for attributes only in the selected area is carried out. Thus, the matching stage is not required.

5.3 Segmentation subtask

We formulate the task of web-pages segmentation as a task of searching for information boundaries, which are first nodes containing text of the DOM subtree and belonging to it. This formulation is the same as that used in the [3].

We test two methods of solving this task:

1. Heuristic method based on classical MDR
2. Neural method based on MarkupLM, which is state-of-the-art model in information extraction from HTML pages

MDR. We tested MDR because it has open-source code. Since this method proposes several candidates for segmentation, ordered by their probability. We choose the segmentation with the highest probability.

MarkupLM. We train the MarkupLM model on this task, having the data previously prepared: for each record on the page, we have marked its first node which contains text with the “BEGIN” label. All other nodes on the page have been marked with the label “OUT”. Thus, the solution of the problem is to predict the corresponding label for each DOM tree node by the model.

Segmentation metrics

The result of the segmentation method can be evaluated by page-weighted metrics: $Precision_{avg}$, $Recall_{avg}$, $F1_{avg}$.

To calculate them, the reference segmentation of the given page on the record is compared with the one obtained by the proposed methods. Based on this comparison, for each segment it is possible to calculate TP_{pageK} – the number of DOM nodes correctly marked as an information boundary, FP_{pageK} – the number of DOM tree nodes that are not an information boundary, but marked with it, FN_{pageK} – the number of DOM tree nodes that are an information boundary, but not marked as it:

$$Precision_{pageK} = \frac{TP_{pageK}}{TP_{pageK} + FP_{pageK}},$$

$$Recall_{pageK} = \frac{TP_{pageK}}{TP_{pageK} + FN_{pageK}},$$

$$F1_{pageK} = 2 \cdot \frac{Precision_{pageK} \cdot Recall_{pageK}}{Precision_{pageK} + Recall_{pageK}},$$

$$Precision_{avg} = \frac{\sum_{i=1}^K Precision_{page i}}{K},$$

$$Recall_{avg} = \frac{\sum_{i=1}^K Recall_{page i}}{K},$$

$$F1_{avg} = \frac{\sum_{i=1}^K F1_{page i}}{K}$$

Also, we will calculate classical *NMI* [11] and *ARI* [12] for segmentation task.

The test results are shown in Table 3. The MarkupLM model shows superior results in all metrics in comparison with the other tool.

Table 3. Results of segmentation experiment.

Method	$Recall_{avg}$	$Precision_{avg}$	$F1_{avg}$	<i>ARI</i>	<i>NMI</i>
MDR	0.473	0.486	0.465	0.437	0.517
MarkupLM	0.908	0.941	0.925	0.802	0.869

5.4 Classification subtask

This subtask includes searching for all potential attributes in the area of interest on the html page (for parallel pipeline – it is the whole page, and for sequential – a part of it).

The MarkupLM model is also chosen as a classification method. We train it on the task of predicting a node's label. Thus, the model predicts the labels: *title*, *tag*, *date*.

Classification metrics

Results are evaluated by classical page-weighted classification metrics: $Precision_{avg}$, $Recall_{avg}$ and $F1_{avg}$.

We trained two models: one for the pipeline with full page context (in such conditions, classification is used in the parallel pipeline), and one with just record context (passing the part of the html related to a specific record directly to the input, so it is used in a sequential pipeline). The comparison of their results is presented in Table 4. So, MarkupLM with record context shows rather high results in this subtask.

Table 4. Results of classification experiment.

Type	Metric	<i>title</i>	<i>tag</i>	<i>date</i>
Record context	<i>Precision_{avg}</i>	0.98	0.79	0.86
	<i>Recall_{avg}</i>	0.99	0.85	0.91
	<i>F1_{avg}</i>	0.99	0.82	0.88
Page context	<i>Precision_{avg}</i>	0.68	0.61	0.64
	<i>Recall_{avg}</i>	0.95	0.76	0.86
	<i>F1_{avg}</i>	0.79	0.89	0.73

5.5 Matching subtask

We propose matching algorithm:

- Let G – set of xpaths (We are considering positional xpath expressions consisting of tags and indices in the DOM tree path from root to the given node) to all DOM-nodes of htmls marked as information boundary at the segmentation stage.
- Let's make $\overline{G}$: for each xpath from G find minimal (by length) node prefix. Each prefix should belong to only one xpath from G .
- Let's enumerate $\overline{G}$.
- Let's refer Xpath of some attribute determined at classification stage as $xpath_{attr}$. Each attribute is matched with record with number N if $\overline{G}[N]$ is prefix for $xpath_{attr}$.

5.6 Final method evaluation metrics

We evaluated the final method using metrics similar to detailed page information extraction task's metrics. We define predicted record as a set of predicted attributes matched to the same record at the Matching stage. Also, we define reference record as a set of ground truth attributes values from a sought information boundary. There are three cases possible when the method is run:

- 1) *We can match the predicted record with the reference record.* TP is the number of correctly extracted attributes (for example if a record contains 4 tags and they are extracted correctly, then TP is 4). FP is the number of extracted attributes but none of them should have been. FN is the number of not extracted attributes but all of them should have been.
- 2) *We cannot match any of the predicted records with the reference record.* Only FN increases for all attributes of the reference record.
- 3) *We cannot match any of the reference records with the predicted record.* In this case FP increases for all attributes of the predicted record.

5.7 Experiments results

We tested both proposed methods: parallel pipeline and sequential pipeline. For these methods, we used the corresponding trained versions of MarkupLM, as we described them in the previous chapters. Comparative results are shown in Table 5, while Table 4 presents the performance of classification model in different contexts – record and page. The results were obtained using the

equipment of the Shared Research Facility “Shared Research Center of the Ivannikov Institute for System Programming of the Russian Academy of Sciences (SRC ISP RAS)”.

Table 5. Results of final method.

Method	Metric	<i>title</i>	<i>tag</i>	<i>Date</i>
Parallel Pipeline	TP	56008	29202	60570
	FP	7021	10804	12728
	FN	8083	4484	12361
	Precision	0.874	0.867	0.831
	Recall	0.889	0.73	0.826
	F1	0.881	0.793	0.828
Sequential Pipeline	TP	55891	28643	58645
	FP	1492	7641	7056
	FN	8801	4899	13659
	Precision	0.864	0.854	0.811
	Recall	0.974	0.789	0.893
	F1	0.916	0.82	0.85

In Table 4, the model using the record context outperforms model using the page context. The tag attribute in the page context performs relatively well, but the overall trend shows that the model is more effective in the record context. We assume that the advantage of page context for tag-attribute extraction is related to the peculiarity of this attribute. Most often, each record has several tags. Information about neighboring records allows making less noisy predictions.

The parallel pipeline demonstrates solid precision, particularly for title. However, it shows a decline in recall. In contrast, the sequential pipeline outperforms the parallel pipeline in recall. The sequential pipeline achieves a balanced performance, making it the preferred method due to its robustness and consistency.

In conclusion, the sequential pipeline is superior to the parallel pipeline. The record context further enhances classification performance, making it the optimal setting for method deployment. The high results of the obtained methods prove their applicability in real life.

6. Conclusion

In this paper, we propose a new large-scale dataset consisting of over 13,000 web pages from Russian news sources, which was specifically designed to address the problem of extracting information from multi-record web pages. This dataset is the first dataset in the Russian language for information extraction from multi-record pages task. It is significantly larger than existing datasets, and includes pages with optional and multi-valued attributes. We make it openly available to provide a valuable resource for researchers of extraction methods.

We have applied and validated multi-stage methods for information extraction. In these methods we fine-tuned a state-of-the-art MarkupLM, which was designed to understand and process markup-language-based documents. Our experiments demonstrated the effectiveness of MarkupLM in handling the multi-record pages. So, we set a robust base for further advancements in this area.

Moreover, our research goes beyond Russian news websites and provides a valuable contribution to development of other forms of semi-structured web content extraction.

The internet continues to expand and the ability to efficiently process multi-record web pages becomes increasingly vital too. Our research provides as a powerful tool for current applications, as a strong basis for future development.

In future work, we are going to study the extraction of additional attributes from our dataset, such as *short_text*, *short_title*, *author*, and *time*. These attributes, while not the focus of the current article, represent further opportunities to enhance the richness and utility of information extracted from multi-record pages. We are also going to expand our dataset with examples from other languages, for example, English.

References

- [1]. J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma, "Simultaneous record detection and attribute labeling in web data extraction," in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 494-503. [Online]. Available at <https://doi.org/10.1145/1150402.1150457>, accessed 31.10.2025.
- [2]. T. Furche, G. Gottlob, G. Grasso, G. Orsi, C. Schallhart, and C. Wang, "Amber: Automatic supervision for multi-attribute extraction," 2012. [Online]. Available at <https://arxiv.org/abs/1210.5984>, accessed 31.10.2025.
- [3]. A. San, Y. Zhuang, J. Bakus, C. Lockard, D. Ciemiewicz, S. Atluri, Y. Ji, K. Small, and H. Elfardy, "Plate: A large-scale dataset for list page web extraction," 2023. [Online]. Available at <https://arxiv.org/abs/2205.12386>, accessed 31.10.2025.
- [4]. J. Kiesel, L. Meyer, F. Kneist, B. Stein, and M. Potthast, "An empirical comparison of web page segmentation algorithms," in European Conference on Information Retrieval. Springer, 2021, pp. 62-74.
- [5]. Y. Zhai and B. Liu, "Web data extraction based on partial tree alignment," in Proceedings of the 14th international conference on World Wide Web, 2005, pp. 76-85.
- [6]. M. Alvarez, A. Pan, J. Raposo, F. Bellas, and F. Casheda, "Finding and extracting data records from web pages," Journal of Signal Processing Systems, vol. 59, no. 1, pp. 123-137, 2010. [Online]. Available at <http://link.springer.com/10.1007/s11265-008-0270-y>, accessed 31.10.2025.
- [7]. J. Li, Y. Xu, L. Cui, and F. Wei, "Markuplm: Pre-training of text and markup language for visually-rich document understanding", arXiv preprint. Available at <https://arxiv.org/abs/2110.08518>, 2021, accessed 31.10.2025.
- [8]. <https://mediametrics.ru/rating/ru/online.html>, accessed 31.10.2025.
- [9]. Scrapy – a fast and powerful scraping and web crawling framework. [Online]. Available at <https://scrapy.org/>, accessed 31.10.2025.
- [10]. <https://github.com/ispras/web-scraper-chrome-extension>, accessed 31.10.2025.
- [11]. A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," Journal of Machine Learning Research, vol. 3, pp. 583-617, 01 2002.
- [12]. L. Hubert and P. Arabie, "Comparing partitions," Journal of Classification, vol. 2, no. 1, p. 193-218, Dec. 1985. [Online]. Available at <http://dx.doi.org/10.1007/BF01908075>, accessed 31.10.2025.

Информация об авторах / Information about authors

Александр Александрович КУСТЕНКОВ – магистрант ВМК МГУ. Сфера научных интересов: сбор данных из веб-ресурсов, автоматизация процесса сбора данных, извлечение информации, машинное обучение.

Alexander Alexandrovich KUSTENKOV – master student of the CMC faculty of Lomonosov Moscow State University. Research interests: data collection from web resources, automation of the data collection process, information extraction, machine learning.

Александр Константинович ЯЦКОВ – младший научный сотрудник Института системного программирования, ведущий программист ВМК МГУ. Сфера научных интересов: сбор

данных из веб-ресурсов, автоматизация процесса сбора данных, извлечение информации, машинное обучение.

Alexander Konstantinovich YATSKOV – junior researcher at the Institute of System Programming of the RAS, leading programmer at the CMC faculty of Lomonosov Moscow State University. Research interests: data collection from web resources, automation of the data collection process, information extraction, machine learning.

Максим Игоревич ВАРЛАМОВ – научный сотрудник Института системного программирования. Сфера научных интересов: сбор данных из веб-ресурсов, автоматизация процесса сбора данных, извлечение информации, машинное обучение.

Maksim Igorevich VARLAMOV – researcher at the Institute of System Programming of the RAS. Research interests: data collection from web resources, automation of the data collection process, information extraction, machine learning.